LEARNING WITH PERSONAL COMPUTERS

Helga A. H. Rowe
LEARNING WITH PERSONAL COMPUTERS:

ISSUES, OBSERVATIONS AND PERSPECTIVES

Helga A.H. Rowe

with the assistance of
Irene Brown and Isabel Lesman

ACER
## Contents

- Preface iv
- Summary vi
- Acknowledgements xiii
- List of Figures xiv
- List of Tables

### PART I THEORETICAL FRAMEWORK

1. Personal Computers in the Pursuit of Educational Excellence 3
2. Cognitive Effects for Students 24

### PART II ACQUIRING COMPUTER LITERACY

3. Curriculum Objectives 57
4. Teaching and Learning 96
5. Assessment and Evaluation 132

### PART III THE EMPIRICAL STUDY

6. Observing Characteristics of Learning with Laptops 155
7. Individual Differences in Attitudes and Learning 197
8. Gender Differences 238

### PART IV ASSISTING THE TEACHER

9. Professional Development for Personal Computer Use in the Classroom 261

References 295
Author Index 316
Subject Index 326
Australia is moving rapidly into the use of computers for learning and teaching in schools. Over a three year period (1984-86) the Commonwealth Government committed 18 million Australian dollars for the development and implementation of a National Computer Education Program. Since then further public and private funds have been raised for the purposes of educational computing by education departments, school systems and individual schools.

In 1984 the Commonwealth Schools Commission appointed a National Advisory Committee on Computers in Schools. The primary task of this committee was to provide advice to the Commonwealth Schools Commission on the development of a rationale and a plan for the use of computers in education.

The principles developed to guide the introduction of computers into schools focused on two issues, namely that the programs be broadly based, and that they would lead to equality of outcome. A broadly based program was defined as one which focuses on teaching about and with computers. Equality of outcome was expected to be achieved by improving the access of all children to computers.

The Commonwealth Schools Commission expected the use of computers in schools to focus on a greater understanding by students of the effects which computers and information technology can have on society. It argued:

An important function of schools is to help students to communicate, to think, to value and to act first and foremost as young people in school, and second in the context of the society in which they live. (Commonwealth Schools Commission, 1984, p. 21)

The use of computers should be seen as an integral part of the teaching and learning process, and should not be regarded... as a potential replacement for teachers, nor should computer aided instruction become necessarily the focus of a school’s use of computers. (Commonwealth Schools Commission, 1984, p. 22)

The Commission produced basic sets of aims and objectives for a National Computer Education Program for students, teachers, principals and administrators. The desired outcomes for students were specified as follows:

Students should learn to

- use computers for inquiry, analysis, information processing, problem solving and recreation;
- make informed and responsible judgments about those aspects of computer use that affect them and others in economic, social, political and physical contexts;
- recognise the sort of problems that are not amenable to computer solutions;
- become conversant with the broad characteristics of the hardware and software with which they are working and acquire the capacity to make a consumer evaluation of such products; and
- undertake a formal study of information processing or appropriate aspects of it. (Commonwealth Schools Commission, 1984, p. 20)

In the years since, computer education has been taken up in all states, and some attempts have been made to meet and refine the Commission’s aims and objectives, but the offerings of schools vary dramatically. Reasons for this include insufficient hardware, inability to obtain suitable software, limitations in teacher education and lack of firm curriculum guidelines. However, the most important obstacle to the successful implementation of computing into schools is probably the lack of knowledge of how to link this technological innovation with teaching and how to integrate it into the day-to-day curriculum. The computer per se cannot produce effective learning, but students can learn effectively with computers.

Convinced of the importance of computing in education for all students, the Queensland Education Department seeks to reduce the lack of knowledge by means of a considerable number of impressive initiatives in the area of educational technology. One of these initiatives was to provide each of 115 Year 6 and Year 7 students with a personal laptop computer for use at school and at home throughout the school year. Though not formally specified, the questions the SUNRISE project took on related to how computers might best be utilised in the classroom, and how their influence can be identified in various social and cognitive contexts. These are practical problems for educational research and development, because it is reasonable to expect that some children might learn better with computers, that most students would develop new thinking strategies which are adaptive to a largely computer-based learning environment, and that individual differences would influence learning with computers differently from learning in the traditional classroom. These are also profound theoretical questions, because they imply that the learning environment is an integral part of students’ cognitive processes and educational and personal development.

The SUNRISE project conducted by the Queensland Education Department at Coombabah Primary School, Gold Coast, Queensland, was the subject of the empirical study of learning and teaching with computers that is described in this book.

As part of its support for the development of such innovative efforts, their dissemination and promotion, the Australian Council for Educational Research contributed funds to this project. We are grateful to the Queensland Department of Education for the provision of access to the SUNRISE classrooms at Coombabah and for the partial funding of the research which resulted in this book.
Summary

This book is about learning and teaching with personal computers. It is aimed at teachers, trainee teachers, those responsible for pre-service and in-service training of teachers, school administrators and parents. The contents of this book should be of particular interest to teachers and school administrators who are planning to introduce computers into classrooms or are teaching students with computers for the first time.

A wide range of practical and theoretical issues is addressed in this volume because of the difficulty teachers and educational administrators are experiencing in obtaining the type of information provided here. The extensive list of references will help those wishing to obtain deeper knowledge in a particular area. An attempt was made to include relevant materials from many sources. The empirical study conducted in the SUNRISE classrooms at Coombabah, Queensland, serves to illustrate the issues discussed and to raise further questions. Although the empirical findings reported here are based on a study in which each student had his/her own laptop computer, readers will find that the results of this study and the deliberations of the book as a whole are equally applicable to classrooms where two or three students share one computer.

The chapters of this book are arranged into four sections. Part I provides a theoretical framework for learning and teaching with computers. Part II deals with issues relating to the acquisition of computer literacy. Part III describes the empirical study conducted with 115 Year 6 and Year 7 students, and Part IV deals with issues relating to the professional development of teachers who teach students with computers and with the evaluation of computer software by teachers.

PART I: THEORETICAL FRAMEWORK (CHAPTERS 1 AND 2)

The promise and impact of personal computers in the pursuit of the goal of turning Australia into a clever country and important issues relating to the evaluation of the effectiveness of computing in schools are addressed in Chapter 1. Some implications of the computer revolution are related to computing in schools. Computing is discussed within a framework of cognitive technologies. Human/machine relationships, more specifically the idea of adapting the technology to one's own purposes versus adapting oneself to the technology, and social interaction between students are discussed. Other topics raised in this chapter include: computing as a cultural component of the learning environment; computer literacy as personal capital; and empowerment through metacognition.

PART II: ACQUIRING COMPUTER LITERACY (CHAPTERS 3 TO 5)

What is being advocated in the three chapters of this section is not that students become programmers in a technical sense, but that they regard computers as a natural and integral part of their lives and that they view what they learn in computing as a fabric of experiences and knowledge which can be woven into many activities in and out of school. Alternative educational philosophies and perceptions of the human/computer relationship will determine the specific uses made of hardware and software in the classroom. The tools themselves are extremely versatile and can support many and quite contradictory educational philosophies and objectives.

Chapter 3 begins with a broad discussion of curriculum objectives and computing policy. How computing will be introduced into a school depends very much on the school's definition of computer literacy. This concept is discussed with respect to other literacies and various types of definitions are provided. The implications of comprehensive and narrow definitions of computer literacy are explained for various components of instructional computing. Suggestions are made and examples are provided of how one might integrate the computer into the existing curriculum. The chapter ends with a discussion of how the technology could best function as a mediator of cognitive development.

Chapter 4 looks at the acquisition of computer knowledge and skills more from the point of view of the teacher and the classroom culture he/she might create. One theme which runs through this chapter is that of control. Learning and teaching with computers provide the opportunity to encourage students to take more responsibility for their own work, motivation and learning. Handling
over, sharing and accepting control can be difficult for both teachers and students. Specific suggestions are made relating to the encouragement of independent and self-regulated learning by students, as they are constructing their own knowledge. The chapter contains a brief discussion of Logo, as this is the language used in the SUNRISE classrooms which were the subject of the empirical study reported in Part IV, and a frequently chosen vehicle for teaching with computers in Australian schools. Most of this chapter addresses quite concrete issues relating to classroom organisation and the interaction with students.

Chapter 5 introduces questions of assessment and evaluation of learning with computers. It deals with expected outcomes and ways of monitoring product and process. Examples of possible assessment procedures are discussed briefly and ways of evaluating the computing efforts of students working in groups is discussed. An initial suggestion is made for the evaluation of student performance on programming related tasks. An example of an objectives wheel is provided which can help integrate computing knowledge and skills, with curriculum units or subjects. At this stage, the area of assessment and evaluation in learning and teaching with computers is probably the most neglected of all the areas addressed in this book.

PART III: THE EMPIRICAL STUDY (CHAPTERS 6 TO 8)

The chapters in this section provide the bulk of the information resulting from the empirical study of 115 students with their own laptop computers, i.e. 56 Year 6 and 59 Year 7 students in the SUNRISE classrooms at Coombabah Primary School, Gold Coast, Queensland. Some results of this study are reported as examples or illustrations in relevant sections of the other chapters of this book.

Two major and severe limitations of the empirical study conducted at Coombabah must be recognised:

1. with increased availability of personal computers, improved software and related curriculum materials, and more professional development activities for teachers, the information presented here might become outdated quite quickly; and

2. a study based on observations made on 115 students from two classes in one non-randomly selected school certainly lacks reliability and thus generalisability.

Nevertheless, in view of the current large knowledge gap in the area of educational computing, information on learning and teaching with computers in Australia must be disseminated even if the findings are timebound. The experiences and reactions of even an unrepresentative sample of students and teachers are not only interesting and informative for teachers and educational administrators who are planning to embark on similar projects, but also raise questions which must be addressed in research and practice. In the absence of similar research information collected in well designed and possibly comparative studies, the attitudes, knowledge, abilities and achievements, learning practices and preferences of students observed in two not too untypical classrooms are expected to provide some useful insights to the reader.

Chapter 6 describes the objectives, design and method of the empirical study. A preliminary conceptual framework, i.e. a system in which the variables under investigation might be related, for such a study is developed. Considerable attention is given to the methods and sources of data which are being collected. The identification of possible attributes of computer use for learning and their measurement are carefully considered. The attributes used in this study are defined and operationalised. An attempt was made to identify possible indicators of effective computer use by students. Though the data of the present study may not have warranted such careful analysis of student profiles of computer use, the reason for this part of the study was to attempt to come up with a way of looking at the attributes of learning in computing which might be used by the authors or other investigators in subsequent studies. In other words, this was an attempt to contribute to the development of methodology in a new area of educational research.

In a very tentative preliminary analysis of student profiles in computing processes, four groups of students were classified on the basis of their patterns of laptop use. The students in group A were labelled orchestrators. These students used the widest variety of learning applications closely linked with teacher and task demands, personal aims and skills, personal learning style and the social demands of the classroom. The attitudes of these students to computing and their uses of their computers reflect a harmony created by flexible and appropriate application. Students in group B were labelled amplifiers, because they tended to use the computer to amplify their existing skills, but viewed it as an adjunct, i.e. a non-essential but at times useful and convenient accessory. These students capitalised on available software and on procedures written by others. Group C students were labelled machinists, because they used their computers mostly as calculators and typewriters. Students in group D were labelled perseverators because of their tendency to use only procedures and programs written by others. They used these over and over again, and spent much time on the same task or activity. They liked drill and practice, particularly in spelling. The groups are described in more detail in the chapter. A small number of students in this study could not be classified on the basis of this scheme.

Other student characteristics reported in this chapter relate to the feelings and attitudes (including anxiety and enthusiasm) of students about computing, self image and confidence, knowledge of computers and computing, learning and problem solving, feelings of control, favourite computing activities and student expectations and perceptions of their teachers. These characteristics were assessed by means of structured interviews, questionnaires, observation and
objective tests. The final part of this chapter reports on an investigation of the students’ own assessment of attitudes to and competence in computing in themselves and their peers. Of particular interest with respect to these findings are the students’ criteria for assessment, i.e. how the students measured success.

In Chapter 6, Year 6 and Year 7 students are compared in their reactions to and knowledge of computing. Chapter 7 analyses the same attitude and knowledge variables but reports on individual differences between students in terms of ability and measured IQ. It also compares students who were judged by both their peers and their teachers as doing particularly well in computing, i.e. the R1 group, with those judged by the teachers as having difficulty in computing, i.e. the R5 group. The final section of Chapter 7 reports on an investigation of the programming habits of the students. Both the production of programs by the students and their understanding of programs written by others were assessed. Three of the tasks given to the students were taken from the published literature. This allowed at least a superficial comparison between the performance of the Australian students and a group of American students, even though the age of the students and the conditions under which they received instruction in computer programming differed in a number of respects. Chapter 7 closes with a brief demonstration of the interaction between students who were engaged in a collaborative programming assignment.

Chapter 8 discusses gender differences in relation to learning with computers. The literature argues that girls are often not given appropriate support and contexts for learning both about and with computers. Some of these findings were supported by our observations at Coombabah. Equal opportunity of boys and girls in access to computers is a problem in many classrooms, but obviously not in Coombabah, where each student has his/her own computer, but differences in participation are not just related to access to computers. We found that in the SUNRISE classrooms boys tended to dominate in discussions about programming procedures and in brainstorming activities. Possible factors contributing to the development of differential interest and achievement in computing, and particularly the development of gender based stereotypes are discussed in the chapter. Of particular interest are gender differences as they relate to computing in particular subject domains. In our empirical study, gender differences were not evident when the students first obtained their laptops; they developed over time and are stronger in the more experienced computer users (i.e. Year 7) than in the less experienced group.

PART IV: ASSISTING THE TEACHER (CHAPTERS 9 AND 10)

The chapters in this section deal with issues relating to the professional development of teachers who are teaching students with computers and make some suggestions relating to the contribution of teachers to the evaluation and development of educational computing software. The empirical study reported in this book was focused on the students learning with computers rather than on their teachers. As only five teachers were involved in the SUNRISE classrooms at Coombabah, the opinions of individuals and groups of teachers outside the project were sought to gather information for Chapters 9 and 10.

Chapter 9 reports on the aims, organisation and contents of staff development activities which could assist teachers with no or limited pre-service training in educational computing. Two likely broad aims for such staff development were identified:

1. to improve the skills and confidence of individual teachers in computer use; and
2. to persuade teachers to explore educational computing and to integrate computing into their teaching practice.

A number of problems relating to existing staff development offerings and the purposes of staff development are discussed. Consideration is given to questions such as whether staff development in educational computing should be voluntary or mandatory, whether incentives are necessary, the duration of courses, and how much training might be required.

Based on an extensive review of the staff development literature in Australia and overseas, published surveys of teachers teaching with computers and some published case studies, a set of conditions for staff development activities in personal computer use in schools was derived. These relate to the appropriateness, variability, incentives, maintenance, objectives, instructor, application and duration of professional development activities. The same literature review yielded a number of content topics and organisational features for professional development activities. A total of 60 Australian teachers, from three states, who teach with computers provided their reaction to the variables which had been derived from the literature review. On the basis of this a number of recommendations were formulated which were presented to the teachers in the SUNRISE classrooms at Coombabah. The teacher recommendations reported in Chapter 9 are thus based on the reactions of a much larger sample than would have been possible without the broader survey. These recommendations are not essentially different from those found in the recent overseas literature. Thus, they may well provide a basis for the planning of such activities in Australia.

The information provided in Chapter 10 is also based on a review of research on computer software and existing guides for its evaluation. The major message of this chapter is that software with improved pedagogical value can result if teachers play an expanded role in its design and development. Together with technical experts in computing and others, teachers should probably be involved in the development of instructional software and provide advice at all stages of development. Teachers are in an excellent position for identifying prerequisites for mastering the concepts and skills to be taught, as well as for deciding on the appropriate means for communicating the subject matter. They can help assure
that the software contains substance as far as its content is concerned, that it is error free, and that it engages appropriate thinking and problem solving skills on the part of the students. Teachers, because they are familiar with curricula in their subject domain, could assist in designing software which can be integrated or at least coordinated with other curriculum materials. The chapter concludes with an exemplary software evaluation procedure, developed by the New South Wales Department of Education (1985), which may be copied and used for educational purposes as long as the source is acknowledged.

Acknowledgements

The greatest single external contribution to this project has been that of Liddy Nevile, my former colleague at ACER, whose desire for a school of the future resulted in the idea of SUNRISE, a vision of making what the powerful tools of technology can provide available to all. She persuaded the Queensland Education Department to set up the Queensland SUNRISE Project and was instrumental in nourishing the slowly developing appreciation of the teachers for teaching and learning with computers.

I express my gratitude to the Queensland Department of Education, especially to Mr Robert Lenahan, Director Resource Management, Mr Laurie Vogler, Mr Jim Tunstall and Mr Ross Lever for allowing the project to go ahead during 1991. Mr Robin Ramsbotham and Mr Glen Finger, the Principal and Vice Principal of Coombabah Primary School and the teachers Jenny Betts, Karen Hallett, Barbara Macfarlane, Dave Mitchell and Zoe Schalch were very accommodating and helpful. I thank them sincerely for their tolerance and support. A big thank you goes to the Year 6 and Year 7 students who participated in the project.

Mr Greg Grimmett helped in many aspects of this project. Above all, he facilitated the communication process between Coombabah and Melbourne. I am most grateful for all his assistance and his valuable advice.

I gratefully acknowledge the assistance of the staff of the ACER library who provided an efficient and speedy service in meeting requests for literature searches and many inter-library loans, and the advice and assistance of Gloria Locock who gave the manuscript its final format.

Finally I express my appreciation to Professor Barry McGaw, Director of the ACER, for allowing me to take time to complete this project.
List of Figures

5.1 Objectives Wheel 146
5.2 Empty Objectives Wheel 147
6.1 Preliminary Conceptual Framework 157
6.2 Results of Year 7 Self-assessment of Competence 187
6.3 Results of Year 6 Self-assessment of Competence 188
6.4 Self-assessment of Competence by the Total Sample 189
7.1 The Series of Figures for Task 3 220
7.2 Performance of Coombabah Students on Task 3 221
7.3 Comparison of Programs of Kurland and Coombabah Samples 226
7.4 Program for Two Flags 228
7.5 Program and Correct Drawing for Robot 229
7.6 The Castle Program 232

List of Tables

2.1 Possible Steps towards Initial Mastery in Programming 40
5.1 Class Record of Informal Comments 144
5.2 Sample Records of Informal Comments on Individual Students 145
5.3 Record of Tool Use 148
6.1 Methods and Sources of Data 159
6.2 Components of Effective Computer Use by Students 164
6.3 Indicators of Effective Laptop Use 165
6.4 Student Measures of Success 191
6.5 Percentage of Students Who Named a Boy Expert or a Girl Expert in Four Subject Areas 192
6.6 Measures of Success of Others 193
7.1 IQ Levels of the Sample 198
7.2 Proportion of Students in Different Intelligence Classifications 199
7.3 Ratings for Student Achievement in Computing 209
7.4 Mean IQs for R1 and R5 Students 209
7.5 Number of Programs Produced in Tasks 1 and 2 217
7.6 Analysis of Programs for Task 1 218
7.7 Analysis of Programs for Task 2 219
7.8 Percentage of Students Choosing Each Figure in Task 3 219
7.9 Percentage of Workable Programs by Shape and Method 223
7.10 Students' Understanding of Parts of the Two Flags Program 228
7.11 Students' Understanding of Parts of the Robot Program 230
7.12 Comparison of Results for Tasks 4 and 5 231
7.13 Students' Understanding of the Parts of the Castle Program 233
7.14 Comparison of Students' Understanding of Tasks 4, 5 and 6 233
7.15 Comparison of Performance on Production and Comprehension Tasks 234
9.1 Conditions for Staff Development in Personal Computer Use in Schools 267
9.2 Topics and Organisational Features Requested for Staff Development 269
PART I

THEORETICAL FRAMEWORK
Personal Computers in the Pursuit of Educational Excellence

The overriding policy question today is no longer whether our schools have slipped into mediocrity, or whether there is a crisis in primary and/or secondary education. Rather, the question is: What can be done to improve the current state of affairs? What can be done to turn Australia into a clever country? An obvious response is to capitalise on technological innovations, in particular on personal computers. However, three interrelated factors currently appear to be restricting the potential contribution of computers as generally accepted tools for learning and teaching from Kindergarten to Year 12. (1) A lack of knowledge about how best to educate teachers in the use of computers in education. (2) A lack of general curriculum objectives and (3) A shortage of suitable instructional software which can be integrated with learning goals and local curricula. All of these factors are addressed in the following chapters.

Herbert Simon, who received the Nobel Memorial Prize in economics in 1978, referred to the invention of the computer as the second industrial revolution. He argues that, like the invention of the steam engine in the industrial revolution, the computer promises to dramatically increase the number and kinds of things we can do and to equally dramatically change the ways in which we do things. He reminds us, however, that it took the steam engine 150 years to have a pervasive influence on society, but computers have been around for less than 50 years. Simon also notes that the full impact of even these landmark inventions is tempered by and conditional upon other events, such as related inventions and accumulating experience resulting from their use.

Computing in Schools

For the computer to bring about a revolution in our schools and education more generally, this technological innovation must be accompanied by improvements in our understanding of the processes of learning and teaching and their implications for cognitive development, and by changes in the organisational structure of classrooms, schools, the curriculum and the broader learning contexts.
Computers have certainly transformed our society. As the products resulting from the industrial revolution amplified and boosted the physical power of humans, the computer revolution has the potential of increasing the power of the mind.

The first generation of computers began to transform society but not the schools. Problems of cost, size, and lack of appropriate software led to restricted access to computers. The invention of the personal computer has changed this. Despite some resistance from unprepared school personnel who had understandable doubts and were thus reluctant to accept the new technology, the easy availability of relatively low priced personal computers has brought about a second computer revolution during the past ten years, and this time schools are likely to be transformed.

Curiously, when politicians and people in the media speak of the computer revolution and its role in Australia's quest for educational excellence, learning and teaching are rarely mentioned, except perhaps to note that teachers might not be adequately prepared for the revolution and need to be provided with professional development. The focus tends to be on different kinds of hardware and its availability, on financial allocations by governments and on selected educational programs available at certain schools. From these news items one might gain the impression that the computer revolution has arrived and that a learning environment now exists in schools in which teachers and students make good use of computers, be it for drill and practice, for the development or remediation of basic skills, learning enrichment for the gifted, or computer literacy.

Unfortunately this impression does not accurately represent most environments in which teaching and learning with personal computers take place. The extent to which progress towards computer literacy and learning with computers is met actually depends on how well students and teachers are able to adapt, not to learning environments, which are close to ideal, but to reality, i.e. situations which contain relatively few resources and less than optimal curriculum materials.

Attempts to make the use of computers in schools more effective, therefore, must begin with the recognition that what is going on in educational computing in most schools currently is less than optimal. Progress can be made, however, for example by assuming that despite the limitations some classrooms in which personal computers are currently used for learning and teaching have produced highly successful students, and that the knowledge gained about learning and teaching decisions and practices, and recommendations for curriculum objectives, staff development and curriculum materials can help fill the current knowledge gap. Any attempts to fill this gap place us in a better position to identify what teachers and students need to know to use personal computers successfully in learning and instruction, what changes in classroom organisation and what improvements in resources, curriculum materials and material adaptation and development are required.

In the current debate regarding feasible roles of technology in education, and in particular advantages and disadvantages of using computers in the classroom, the machine itself is too often the starting point. Treated as the newest delivery system for teaching and learning, computer hardware and software, and other information technologies, become the main topic for discussion. A more productive debate of the uses of technology in education might result if the focus were moved to the processes of education, and the learner, rather than the physical components of the new technology.

We need to be more concerned about the different ways students can learn when using the computer, the pressures on students now and in the future when they use computers, the learning processes underlying present and future software, the power of various kinds of feedback, and the psychology of the interaction between students and computers. (Hattie, 1988, pp. 1-2)

As educators we are concerned with improving both our basic understanding of children's learning and thinking, and the processes of education which can be expected to develop them. It is expected that this book will play at least a small part in improving this understanding with respect to the use of computers in the classroom.

A key component of learning and teaching is the processing of information. Students and teachers, like most other members of our society, collect, interpret and make decisions with respect to a vast array of information, some of which was gathered by earlier generations or by people in distant locations. Obviously, information processing has always been an aspect of learning, but the quantity of information now available to us is increasing exponentially. This is due largely to the availability of new technologies. At the same time we can make use of technology to help us handle all this new information. There is a long tradition of using technological aids such as printed matter, paper and pencil, an abacus or calculator, as tools to supplement and amplify human mental powers.

The computer is the latest addition to this range of tools. It is argued that by using the computer as a tool it becomes both an amplifier of human capabilities and a catalyst to intellectual development. This view of computing leads to more productive outcomes in the classroom than do those uses which turn the computer into a surrogate teacher. Obviously, the more effective uses of computers in education will require new patterns of interaction between students and teachers, changes in the social organisation of the classroom, the adaptation of curricula and alternative purposes and modes of student evaluation. All of these issues are addressed in this book.

COGNITIVE TECHNOLOGIES

Long before the arrival of computers, remarkable extensions of human intelligence were accomplished through the use of technical instruments. It is
taken as axiomatic that intelligence is not merely a quality of the mind, but a product of the relationship between mental structures and tools of the intellect provided by the environment, and more generally the culture (Bruner, 1966; Cole & Griffin, 1980; Luria, 1976, 1979; Olson, 1976, 1985; Olson & Bruner, 1974; Vygotsky, 1962, 1978). I shall refer to these tools as cognitive technologies. A cognitive technology is provided by any means that help transcend the limitations of the mind, such as restrictions in short-term memory, in activities such as thinking, learning and problem solving. The technologies which have tended to receive most attention in this respect are writing systems (e.g. Goody, 1977; Greenfield, 1984; Olson, 1977; Ong, 1982; Scribner & Cole, 1981) and systems of mathematical notation such as algebra or calculus.

Let us reflect on computers as cognitive technologies. Computers can store and dynamically manipulate symbols. It so happens that symbols appear also to serve as the primitives of human thought. Capable of real time interactions with human users, computer programming may well provide the most extraordinary cognitive technologies ever to be devised. Past experience with non-computer cognitive technologies may well help to inform and guide our definition of priorities for future uses of computers as cognitive technologies in education.

Cognitive technologies, such as written languages, are commonly thought of as cultural amplifiers of the intellect, to use Jerome Bruner’s (1966, p. xii) influential phrase. They are viewed as cultural means for empowering human cognitive capacities. Greenfield (1966) observed that cultures with technologies such as written language will push cognitive growth better, earlier and longer than others (p. 654). As discussed in Chapter 2, we find similar predictions for computer technologies based on a widespread belief that computers will inevitably and profoundly amplify human mental power, and alter both what we do and how we do it.

The amplification metaphor for cognitive technologies has led to many research programs, particularly in relation to the cognitive consequences of literacy and schooling in the decades since Bruner and his colleagues published studies in cognitive growth (e.g. Bruner, 1964a, 1964b; Bruner & Tajfel, 1961; Greenfield, 1984; Olson, 1976; Scribner & Cole, 1981). This metaphor continued, for example in work on electronic technologies such as the prototype software systems for writing and mathematics in the form of idea amplifiers, notebooks, etc. and their uses for learning with computers, which are discussed in detail in later chapters of this volume.

While quantitative measures such as the efficiency and speed of problem solving, decision making and learning may truly describe changes that occur as a result of working with electronic tools, it can be shown that more profound changes -- as will be described in later chapters -- can be missed if we confine ourselves to the amplification perspective. Computing can do more than increase the efficiency, speed, reliability and comprehensiveness of our mental efforts: it can actually change the tasks problem solvers are faced with and thus alter the cognitive processes involved in solving these tasks. Thus, computers can bring about change in the forms of thought itself.

Another tradition in the study of cognitive technologies can be characterised as cultural-historical. Influenced by the writings of Vico, Spinoza and Hegel, Marx and Engels developed a theory of society now described as historical or dialectical materialism. In this view, human nature rather than being a product of environmental forces, is of the person’s own making and continually becoming. Humans are shaped through a dialectic of reciprocal influences: our productive activities change the world, thereby changing the ways in which the world can change us. By shaping nature and how our interactions with it are mediated, we change ourselves. As the biologist Stephen Jay Gould observes (1980), such cultural evolution, in contrast to Darwinian biological evolution, is defined by transmission of skills, knowledge and behaviour through learning across generations and has been our nature-transcendent innovation as a species.

From this cultural-historical perspective, labour is seen as the connecting link, i.e. the mediating factor, between humans and nature. By creating and using physical instruments and machinery which mediate in less and less direct ways our interactions with nature, we come to reshape human nature. Note how a change in the instruments of work (e.g. a plough rather than the hand) changes the functional organisation or system characteristics of people’s fundamental relationship to work. Not only do humans accomplish their work faster, but what they do, i.e. their task, changes.

In efforts to integrate accounts of individual and cultural changes, the Soviet theorists Vygotsky (e.g. 1962, 1978) and Luria (e.g. 1976, 1979) generalised the historical materialism developed by Marx and Engels for physical instruments, and applied it to a historical analysis of symbolic tools such as written language which serve as instruments for redefining culture and human nature. Vygotsky (1978) recognised that

the signs [symbols of language] act as an instrument of psychological activity in a manner analogous to the role of a tool in labour. (p. 52)

Using a Vygotskian perspective, which stresses the functional reorganisation of cognition with the use of symbolic technologies, Cole & Griffin (1980) argued that the amplification metaphor has important shortcomings. Specifically they discussed how symbolic technologies qualitatively change the structure of the functional system for such mental activities as problem solving or memory. These fundamental changes are likely to go without being noticed if one thinks about cognitive technologies only with the amplification metaphor. Cole & Griffin highlighted how Luria enriched the term function for psychology. We are often inclined to assume one-to-one correspondences between functions and structures (e.g. planning is a function of the frontal cortex). In contrast, Luria speaks of the function of respiration not as the function of particular tissue, but as an entire functional system consisting of many components, such as the motor,
sensory and autonomic nervous systems. For Luria functional systems are distinguished not only by the complexity of their structure, but also by the flexibility of the roles played by constituents (Cole & Griffin, 1980, pp. 347-8).

In similar fashion Vygotsky saw shifts in functional systems of thinking as the sine qua non of developmental change:

I have attempted to demonstrate that the course of child development is characterised by a radical alteration in the very structure of behaviour; at each new stage the child changes not only her response but carries out the response in new ways, drawing on new instruments of behaviour and replacing one psychological function by another. (Vygotsky, 1978, pp. 72-73)

By contrast, Cole & Griffin (1980) note how use of the term amplify means to make more powerful, and to amplify in the scientific sense. They suggest that amplification

refers rather specifically to the intensification of a signal (acoustic, electronic) which does not undergo change in its basic structure. (p. 349)

As such, the amplification metaphor leads one to unidimensional, quantitative theorising about the effects of cognitive technologies. For example, the use of paper and pencil can be thought of as amplifying the power of a student’s memory for a long list of words when only the outcome of the list length is considered. But it would be incorrect to go on to say that the mental process of remembering the words in the list, which led to a particular outcome, was amplified by the use of paper and pencil, because remembering in the two instances refers to two qualitatively different activities. The pencil does not amplify memory capacity, but it restructures the functional system for remembering, and thereby leads to a more powerful outcome (at least for the purpose of remembering more items).

Olson (1976), Ong (1982) and others argue in a similar way about restructurings of thinking process created through written language. For example, logical analysis of arguments for consistency/contradiction becomes possible because memory limitations for oral language are mitigated, and print (rather than oral narrative) provides a means to store and communicate cultural knowledge. It is important to remember that the possible restructurings of cognitive technologies are an empirical rather than a definitional matter. Cognitive restructurings are rarely predictable. They have emergent properties which come to be discovered only through their use. In this sense, as Dilthey (1976) urged, human history like evolution is a postdictive rather than a predictive discipline.

UNDERSTANDING THE IMPACT OF COMPUTERS

Society itself is rapidly and fundamentally changing in its structure and activities. Many of the changes are rooted in new ways of generating, storing, communicating and using information. We are shifting from the industrial age to the information age. In part this means that an increasing number of people are spending more and more of their time handling information.

Computers and related communication technologies are the visible signs of the information revolution. Hence, it has become important for all educated members of society to acquire basic computer literacy. This concept is discussed in the following chapters. Increasingly, parents, teachers and students believe that by learning about computers and by being able to use them, the students will be better prepared to survive and to enjoy economic well-being in the changing world. Educational computing has become the means for adapting schools to the new age.

What is it about computers that makes us so optimistic about their beneficial effects in the pursuit of educational excellence? The characteristics of the computer which most immediately account for its growing popularity and ubiquity are its ability to employ a wide range of symbols and to operate on symbolic expressions in powerful ways. In fact, these are the capabilities of computers which most closely correspond to human information processing. But computers not only employ a range of symbol systems, they differ from other media in the ways they can be used to structure and apply this information. The computer can be used to connect information based on the ways ideas are related to each other. Words can be linked to their definitions or to referent pictures. Concepts can be connected to examples for them. Theoretical principles can be linked to animated programs or video demonstrations. The computer’s processing capability can be used to create procedural systems in which the information provided by the user determines what happens next. Such explicit representations of the relationships among information and symbolic expressions can serve as models for how knowledge can be related, structured, and used.

The emergence of the personal computer as an instructional tool has been surrounded by enormous publicity and speculation, which has tended to obscure many of the substantive issues surrounding its real and potential uses. Schools, parents and others in technologically advanced societies have shown themselves to be highly susceptible to the promise that the introduction of computers will provide the definitive answer to teaching and learning. In part, this susceptibility reflects one facet of our age, namely a fascination with machines and the belief that they might create progress and new frontiers. Every school strives for the latest and best in educational technology. However, the tendency to focus solely on the computer, rather than on the multitude of concurrent interactions with the learning environment, is a problem that faces teachers, curriculum planners and administrators alike. There is a growing realisation that new technologies will not
be easily integrated without a more definitive understanding of the interactional context into which they are being introduced.

Neither the features inherent in the machine nor the characteristics of the software will determine the influence of computers and computing in education. Rather, it is what people do with the machine that determines the influence of personal computers in any area of society. Hence the questions addressed in the study reported in this volume do not focus on the effects of computers on student learning, classroom structure, etc. The focus of the study is on what students do with the computer. How they adapt the technology for themselves and how they use it, rather than how the students themselves adapt to the technology.

HUMAN/MACHINE RELATIONSHIPS: ADAPTING THE TECHNOLOGY VERSUS ADAPTING TO IT

How could we view our relationship with the computer? What we will achieve with this technology will be determined by the uses which we can imagine for it. If we view the computer as we view a pencil, e.g. as a tool with which to produce a piece of writing, then we will obtain different results than if we view it in the way we do a wristwatch. This is the distinction between machines which work for us (e.g. motor cars, washing machines and other engines, watches, lights, and also computer programs designed for drill and practice) and those with which we work, i.e. tools (e.g. pencils, scissors, garden tools and word processors). We adapt to the machines which work for us and we adapt the machines and instruments which are our tools so that they best serve our purposes.

Human/machine relationships can also be thought of in terms of their degree of transparency, defined by the extent to which the machine becomes an extension of the human user or remains as separate, in psychological terms a significant other. Whereas a pencil or a spade is an extension to oneself, i.e. an addition which has been made to make it possible for a person to extend certain powers, to improve or make possible certain outcomes, other tools are not extensions as much as separate objects with their own purposes. They are agents rather than parts of the user.

Classroom computers have a primary effect which is transparent, or at least translucent, but it remains to be seen what the secondary effects will be. There will be motivational and social effects of educational computing, and effects on educational philosophy, but the most potent effect of computing is expected to relate to the development of students’ powers of thinking.

Papert, one of the creators of the computer language Logo, and during the 1980s probably the leading exponent of the use of computer programming to expand students’ intellectual power, proposed that computer programming environments in which children can deal with concepts, formerly regarded as too abstract for their developmental level, stimulate the development of important intellectual processes and can create conditions under which intellectual processes may take root. Computer programming can make the abstract concrete and personal, and thus help children learn more effectively by making their thinking processes conscious. By programming the computer to do what the student wants it to do, students are forced to reflect on how they might deal with the task, and therefore on how they themselves think. Computer programming thus holds the promise of being an effective device for cognitive process instruction, i.e. teaching how rather than what to think.

There are many unanswered questions regarding the effects of computer programming experience on students’ cognitive and personal development. Exploratory studies by the developers of Logo and others (e.g. Papert, 1980; Papert, Watt, diSessa & Weir, 1979; Clements & Gullo, 1984; Gorman & Bourne, 1983) indicate that even quite young students can learn to program and that they seem to profit intellectually. There is some evidence that programming can improve problem solving ability (e.g. Billings, 1983; Milner, 1973; Soloway, Lochhead & Clement, 1982). Other studies report considerable variability in skill levels attained by individual children, and that students’ programming ability is often limited to specific contexts (e.g. Pea, Hawkins & Shingold, 1983).

Neither pencils nor computers can be regarded as an independent variable which is introduced into a classroom, the effects of which can then be observed. The computer does not cause better or worse learning in mathematics or social studies. It does not cause more social interaction or less. The computer is not an agent but something which has become part of the learning environment and the total social environment in many different ways. No wonder that computers have been used in support of the most diametrically opposed theoretical approaches to learning and teaching. Computing can be used to make highly structured learning even more structured, and it can be used to make open classrooms more open.

Above all, it can be used to increase the learner’s self-directed exploration of learning tasks and problems. It can help students become better learners because it can provide individuals with explicit knowledge about their own learning and thinking processes.

Computing can provide a more personal relationship with many aspects of knowledge and thinking, because it is such a rich source of reference points for so many otherwise abstract ideas. Obviously the computer plays a role in learning and teaching even when it is not physically present. Papert points out:

The computer in the head can often be a more effective aid to instruction than the computer on the desk. (Papert, 1987a, p. 182)

A popular argument by the early 1980s was that learning to program computers is more empowering for students than computer assisted instruction (CAI) or computer managed instruction (CMI), because in programming the user, rather than the computer, directs the interaction and is thus in control. In CAI and CMI the computer is in control.
Papert’s book *Mindstorms: Children, computers and powerful ideas* ushered in a wave of enthusiasm about the powerful, cognitive benefits of teaching children to program computers. Another early advocate of computer literacy, Luehnman (1980) also argued that students should be taught how to use and control computers through programming them rather than being controlled by them as in CAI and CMI.

Computing constitutes a new and fundamental intellectual resource. To use that resource as a mere delivery system for instruction, but not give a student computer instruction in how he might use the resource himself, has been the chief failure of the CAI or CMI efforts. What a loss of opportunity if the skills of computing were to be harnessed for the purpose of turning out masses of students who were unable to use computers. (p. 133-4)

Both Papert and Luehnman view students as active learners. They envisaged the computer as a *tutee* to be programmed or a *tool* to be used by technically knowledgeable students to serve their own needs, terms popularised by Taylor (1980, pp. 1-4) to distinguish these computer uses from uses that envisage the computer as a mere delivery system.

If we want our students to be decision makers in the future, not passive recipients of technological accomplishments, we need to make sure that they develop empowering knowledge. We need to structure opportunities for them to question as well as to learn to use and understand technology. It is the development of the expectation and responsibility to question and inform policies, coupled with technical knowledge, which promises to be more empowering than technical knowledge alone. Watt (1982) describes this type of knowledge as the ability to understand the growing economic, social, and psychological impact of computers on individuals and groups within our society and on society as a whole.

This includes the recognition that the computer applications embody particular social values and can have different kinds of impacts on different individuals and different segments of society. (Wan, 1982, p. 58)

Recognising that personal computers are dynamic components of a larger social system enables us to see the relationship between classroom culture, appreciation of computers, and learning and teaching as a mutually defining (rather than a unidirectional) relationship.

### SOCIAL INTERACTION

Critics of the policy of providing students with personal computers often believe that computers will negatively affect social interaction in the classroom. Some parents have expressed fears that having unrestricted access to their own computers may result in children *spending many hours alone, sitting in front of* the computer, and will lead to isolation of the individual from his/her family and peers. An answer to this criticism is that there are *loners* of all ages who tend to withdraw from interaction with other people and focus their energies on themselves or their hobbies, e.g. their computer, their music, or their television set.

Studies on social interactions in classrooms containing computers indicate that computer presence actually increases the amount of interaction among the students (e.g. Emihovich & Miller, 1988a; Papert, 1987a) and our empirical findings in the SUNRISE classrooms at Coombabah strongly support these findings. Computing allows for dialogue between students, and projects that encourage cooperation. It allows students to create pieces of work or other outcomes about which they are excited and which they want to discuss with others. It facilitates new means of communication by exchanging and cooperating on files, sending messages by computer mail to recipients within and outside the classroom. In fact, one of the aims of the designers of Logo was to encourage communication between users.

Logo programs are modular so that they can be borrowed and shared. Logo is also designed to make it as easy as possible to talk about how you made your program work -- what the bugs were, what the difficulties were, and how you solved them. Thus the content of actual computer work, even on what might seem like a very technical level such as designing a computer language, is a factor that can make for greater socialisation or greater isolation. (Papert, 1987a, p. 185)

Even in the rare classrooms where each student has his/her own computer, group work is likely to be the norm. Interaction between students during computer-based activities is regarded as beneficial, and teachers in such classrooms actually encourage pupils to talk, to reach group decisions and help each other with new procedures. For example, it has been shown that working in pairs or small groups at the word processor, even at the error correcting stage, can be highly beneficial. Students learn quickly from one another.

Lepper (1985) has commented on motivating aspects of computer use, and has pointed to the opportunities for an education that capitalises upon them. Papert refers to the sense of mastery over one’s environment as a powerful motivator when children program in Logo, and Underwood & Underwood (1990) have found the same growth in self esteem, with consequent improvements in cognitive development when children work on databases. Self esteem may develop because children feel at the centre of their worlds. Motivation in such situations is due to the sense of positive power which comes from the ownership of a skill or knowledge, and is closely related to self esteem.
COMPUTING AS A CULTURAL COMPONENT IN THE LEARNING ENVIRONMENT

Papert (1987b) claimed that researchers who studied the effects of Logo on educational outcomes and cognitive processes were victims of what he called technocentric thinking. He defines this term as thinking which makes the properties of the machine the paramount feature of concern. Papert rejected the use of questions such as *What is the effect of the computer on cognitive development?* and *Does Logo work?* and suggested that these kinds of questions revealed a lack of understanding that 'the context for human development is always a culture, never an isolated technology' (p. 23).

The following quotations from Papert (1987b) provide some sense of his general argument:

> Developing a discourse is at the heart of developing a culture, and a more textured and knowledgeable discourse about Logo contributes to the *Logo culture*, the *computer culture*, and to the *learning culture* in its broadest sense. It sets the cultural context for personal learning. (p. 23)

> However, the finding as stated has no force whatsoever if you see Logo not as a treatment but as a cultural element -- something that can be powerful when it is integrated into a culture but is simply isolated technical knowledge when it is not. (p. 24)

> In a parallel way, I have sought to decenter the perception of the Logo experience. We are not looking at the effects of a technological object on an individual child, we are looking at the workings of a cultural process. (p. 29)

The second thrust of Papert's argument relates to methodological issues of how evidence should be collected to support the point that computer usage is embedded within a cultural context. Papert revealed a strong preference for anecdotes, single classroom case studies, and ethnographic research methods. He derided the use of experimental design as having little value because he believes the very nature of the design to be based on a one-to-one cause-effect relationship, which is ill-suited to revealing the multiplicity of factors that affect children's performance when using computers in the classroom.

Much of the debate about the effects of learning and teaching with computers in Australia and overseas assumes that a personal computer with a particular kind of software will have a specifiable and generalisable impact on students and teachers. Computing is regarded as a single factor of change introduced into a classroom which otherwise remains the same. In other words, the computer is perceived as an independent variable the effect of which can be controlled and quantified. In reality, computers in the classroom are far more than a treatment.

They become inextricably intertwined not only with the way in which students might go about tasks, but with the whole context of learning and teaching.

Computers in the classroom alter the collaborative interaction and shared dialogue between students, and between students and teachers. The introduction of computers changes the classroom culture. A fundamental feature of any attempts to evaluate the impact of this technology must thus be a focus on the dynamic interplay between learning processes, students, teachers and the learning context (cf. Rowe, 1991b). As noted above, it is not the features inherent in the machine but what students and teachers do with it that determines the effects of computing in schools.

Papert stresses the fact that children have access to programming and other computer applications in a multitude of ways, not just in school, and certainly not just as a treatment created by researchers who set out to compare performances between groups in controlled experiments. Although Papert may not encourage the use of Logo in experiments, he did imply that when it is used by teachers, positive outcomes are achieved. As evidence he stated that children who have been through Logo training obtained higher reading and attendance scores than children of similar background, not commenting on the causal relationship between differential use of Logo and different outcomes. These findings have been supported in more recent research.

Papert favours the use of Logo in natural experiments, while attributing the negative outcomes achieved in a controlled experimental study to the research designs used. While he did acknowledge that other factors could have played a role (e.g. parental motivation) in obtaining higher reading and attendance scores for the Logo groups, he uses this point to suggest that 'factors of this kind simply don't work one by one; they work as a web of mutually supporting, interacting processes' (Papert, 1987b, p. 26). In other words, he believes that controlled experiments cannot capture the web; they focus only on one factor at a time and miss the overall process of how the web evolved.

Of the three critics who responded publicly to Papert's (1987b) comments, only Pea (1987) addresses the cultural argument, though in a limited way. The other two (Becker, 1987; Walker, 1987) focused primarily on methodological issues which will be raised in later chapters of this volume. Pea emphasised that the research he and his colleagues conducted at the Bank Street College of Education, New York, was not limited to experimental studies. For example, the comprehensive, two-year case study conducted by Hawkins (1987), referred to in later chapters of this book, documented the experiences of three teachers who implemented Logo in their classrooms. The study revealed that Papert's claim of the value of Logo needed to be substantiated within the realities of classroom life. Even if Logo is conceived of as building a new culture of learning, it is still necessary to have more precise information about how it might affect children's performance in various domains, to determine differences in learning styles which children bring to computer usage and to consider these differences in designing instruction. Several other researchers besides Pea and his colleagues (e.g. Kull, 1986; Leron, 1985) have found that discovery learning or messing about with the computer, as Papert terms it, leaves many children unable to grasp the conceptual power of Logo to produce deep mathematical thinking in the way Papert originally envisaged it.
My view is that both sides have explained only a piece of the puzzle, because neither side has the appropriate conceptual starting point from which to begin the analysis of computer use in schools. Although Papert is on the right track in emphasising the cultural aspects of computer use, his framework is just too diffuse. It is almost impossible to move from his sweeping assertions to the level of students in the classroom, and then to locate classroom computer use within a wider social context. In contrast, Pea and his colleagues began their work by focusing on individual cognitive processes, and while these should be considered, a fuller account is needed of how cognitive processes are themselves embedded in social practice, perhaps along the lines suggested by Rogoff & Lave (1986). If computing in the classroom is a cultural component and a social practice, rather than a technological one, the crucial ingredient is people's experience with computing and not any inherent features of the hardware or software.

The development of computer literacy, for example, is a much more complex issue than simply learning how to use word processing, spreadsheets, databases and other programming capabilities of the computer. Several chapters of this book have been devoted to this topic. For the students of the 1990s and beyond who are preparing to live in our increasingly technological world, computer literacy makes up a large part of their personal capital. Obviously, this view raises issues of equity and access to computers, the same issues that are generic to the development of literacies in other areas such as reading, writing or mathematics.

COMPUTER LITERACY AS PERSONAL CAPITAL

Although there is little consensus over the term computer literacy, many of the definitions correspond to Papert's concept of technocentrism in that the effectiveness of the machine is considered primarily in terms of its use as a tool, either for personal efficiency or for carrying out instructions (given by others) more efficiently. Most definitions of computer literacy focus on the need for children to acquire expertise as programmers, for adults to use the computer as a tool to accomplish other goals, such as word processing, or for teachers to use the computer as a delivery system for CAI or CMI. In that none of these definitions view the computer as located within a complex social system like the classroom, they are analogous to traditional definitions of literacy as the simple acquisition of reading and writing skills.

As will be discussed in later chapters, the aims of computer literacy curricula range from those designed to rise general awareness of computers through skill in their use to the possession of broader understandings of the personal, educational, social, economic and political contexts and consequences of computer technology in society. The type and amount of knowledge one has about computers determines the potential of that knowledge to be socially and politically empowering.

Computer awareness is intended to provide the student with some general information about computer hardware, software, vocabulary, uses, history, and social impact. Students may or may not actually see or use a computer, but if they do, the use is usually limited to demonstrations. In their teaching of computer awareness, teachers rely on a variety of sources of information about computing, such as books, films and videos. Technical information and noncontroversial commercial applications are emphasised. Much of the information is produced and provided by the computer industry.

Another form of computer literacy emphasises the ability to use computers, where programming and applications dominate the curriculum. The hands-on approach is stressed, because the aim is to train students to control the computer. Programming, word processing, databases and spreadsheets are introduced in the elementary grades and more systematically elaborated in the middle school and high school mathematics and business education departments.

A third type of computer literacy is one which aims to have students make the computer part of themselves, and their personal work, as well as social environments. This type of computer literacy, which allows students to work with computers as malleable tools, rather than have computers work for them, is potentially most empowering, yet it appears to be the least discussed and experienced in schools. It introduces computing skills and knowledge within a broader social context, stresses the implications of computer technology and the empowering effects of such knowledge. The hands-on approach to teaching computing enables students to develop considerable computer specific skills, including graphic skills, and word processing. Students become confident in their interaction with the computer. But demystifying what is generally a user-friendly personal computer might contribute little to understanding the importance and potential of the technology for the individual. In fact, a narrow focus on the technical skills of using computers may even lead the students to a false sense of empowerment.

The technical focus shifts attention away from social questions and portrays computers as something to learn rather than as something to think about... The computer is portrayed as friendly and accessible... and the user is encouraged to think that all computers, even those in large systems, are friendly and accessible. In this manner, computers are further mystified in the very act of demystification. (Noble, 1985, p. 72)

Students' technical knowledge needs to be accompanied by understanding and knowledge of who controls the direction of computing, for what purposes, for whose benefit and whose loss.

In the information age, the ability to access and use knowledge becomes a form of capital. It becomes a resource that allows those who have it to succeed in a society that depends increasingly on the manipulation of information rather than on just the production of goods and services. In the classic Marxist sense, the goods and services produced by human labour was the capital of society, but recently sociologists and economists have broadened that definition to include...
other facets such as intellectual products as a form of capital as well. From this perspective, computer literacy can be described as personal as well as cultural capital. Treating computer literacy as a form of personal capital raises issues relating to opportunity of participation and equity of outcomes.

The personal capital of children who receive only remedial instruction in the use of computers, or are restricted to CAI and CMI, is severely reduced, since they have little opportunity to acquire a broad set of competencies applicable to a wide range of computing and more general problem solving and learning situations. In effect, they are deskillled (Apple, 1982), competent only in managing computer related tasks in which all the steps are prescribed. Just as the student’s educational options are limited with only limited reading and writing skills, the same limitations occur with computer use where students can only follow directions and cannot adapt the computer for their own purposes. There is a large difference between students who experience the computer in terms of what it tells them to do and children who learn to view the computer as an interactive partner.

It is not too difficult to imagine a counter response to the argument presented above along the lines of What’s wrong with teaching students basic reading and writing skills on the computer? No one would argue that reading and writing skills are not important. The key is that teachers can be teaching reading, writing and arithmetic at the same time as giving the students access to a sense of the potential and power of the technology that can fundamentally alter how they perceive themselves. In the best educational environments, computers will become an extension of the mind and they can allow students and teachers to see possibilities which they had previously hardly imagined.

One of the changes which has been apparent in classrooms where students are using computers as tools has been the subtle shift in the social order and power structure. The autonomy of the students increases. Students and teachers share experiences and become partners in learning. This can bring about

an encounter between thought and reality, between desire and possibility, that takes places in the symbolic realm, and thereby vastly multiplies human capacity to process, analyse, critique and re-invent experience. (Easton, 1989, p. 429)

This view supports Papert’s (1980) original intent for developing Logo, a medium through which whole microworlds can be created on a computer and children can experiment with multiple environments in endless ways. This sense of empowerment is especially important to children whose previous experience with school has consisted of one failure after another. As children become empowered in their quest for learning, the dynamics of the social learning environment change as well.

EMPOWERMENT THROUGH METACOGNITION

The term empowerment is used here in the sense of students’ recognising that they have control over their ideas and thought processes, that they can access what they are thinking and then describe it to others. The psychological literature refers to this source of empowerment with the term metacognition, which means thinking about one’s own thinking. It also refers to an individual’s awareness and regulation of his/her cognitive processes and strategies (e.g. Brown, Bransford, Ferrara & Campione, 1983; Flavell, 1979, Rowe, 1988).

Metacognitive skills help students to monitor their strategy use during any cognitive enterprise, in accord with changing circumstances. The importance of metacognitive processes is that without them students find solving higher order problems almost impossible. These skills can be considered as another form of personal capital, one which many minority groups and/or children from deprived home backgrounds have been less successful in acquiring either in or out of school. The point to be stressed here is not that these students lack the cognitive capacity to acquire these skills, but that they have not been provided with the types of experiences that would help develop them.

In a series of studies investigating whether learning computer programming would affect the development of children’s metacognitive abilities Emihovich and colleagues (Emihovich, 1989; Emihovich & Miller, 1988a, 1988b, 1988c; Miller & Emihovich, 1986) wanted to learn whether teaching children (ranging from preschool to grade 4) in urban and rural schools how to program a computer would help them become more aware of their thinking processes, in terms of how they planned and executed solutions to problems they generated themselves. Logo was used because it enabled students to be involved in programming activities. To make the turtle (i.e. the cursor) move, the students type in a set of commands indicating direction, along with a number of spaces they want the turtle to move. For example, forward 50 or (FD 50) would result in a line on the screen from one spot to another. In short, what the students see on the screen would be a graphic representation of the image of the move they had planned in their minds.

Most of the children the above investigators worked with had obtained low scores on standardised psychometric and achievement tests. They were representative of the children who are typically denied experiences to stretch their minds, because of a concern by teachers over their lack of basic skills. Despite these perceived deficiencies, however, these children succeeded over time in learning to combine commands to create very complex designs. The investigators believe that this programming experience became an empowering one for the children for two reasons:

1 It provided the children with an opportunity to excel in a task which was cognitively demanding. They worked on tasks which, as a result of their low test scores, these students would not have been considered competent enough
to attempt in most schools. Although in one study the investigators were able to show a relationship between Logo programming and improved performance on a standardised mathematics test, that was not the major purpose of the research. The investigators wanted to prove that certain children should not be denied access to computer programming and computer literacy simply on the basis of previous test scores. Access to computer literacy, as demonstrated in these experiments, is a strong source of empowerment.

2 The investigators did not view computer literacy in the sense of students becoming expert programmers, but in the sense of children becoming aware of the fact that their ways of thinking, speaking, and writing could be mapped onto a powerful piece of technology which allows them to have access to the content of their thoughts. Using the turtle allowed the children to see the connection between what they had envisaged in their minds and what they actually drew on the screen. As described by a 6th grade student in the SUNRISE classroom at Coombabah: It's like drawing things straight from my mind. Like, whatever you think you draw.

Olson (1985) has suggested that to be intelligent in the society of computer users is to be skilled in making one's meaning explicit (p. 7). Through programming students become more aware of the need to be explicit about the way they communicate with the turtle, i.e. exactly how they go about tasks. This awareness will lead our students to become more independent and self-directed in their learning, and to take control of their own motivational, problem solving and learning efforts.

THE ROLE OF TEACHERS

As a result of the new educational technologies the work of teaching and the role of teachers are likely to change with respect to curriculum content, classroom management and student assessment. The computer is a powerful tool and tool box. It is an instrument which facilitates the acquisition of knowledge and skills in active ways. The student is provided with an environment which is conducive to exploratory learning. Learning through exploration puts high cognitive demands on students. At times this may result in inefficient and ineffective learning strategies, where learners flounder and do not use the opportunities the classroom environment offers. This is why support is needed if learning with computers is to be effective. Most of this support should be given by the human teacher, although some can be derived directly from the computer software.

In describing children’s experiences with computing we must not lose sight of the fact that successful acquisition of computer literacy depends not only on changes in individual cognitive processes but to a large extent on the social practices surrounding how the instruction is presented. The way computing is taught in the SUNRISE classrooms at Coombabah, while not intentionally based on the work of Vygotsky (1978), could certainly be reconciled with his theory. Vygotsky emphasised the role of social interaction in the modification and development of metacognitive skills in children. Vygotsky’s concept of the zone of proximal development was defined as

the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers. (p. 68)

In the above noted studies by Emihovich and her colleagues, adult guidance was provided by tutors who helped the children understand the processes involved in programming in order to bridge the gap between the child’s understanding of the task, what was accomplished and the outcomes which were expected in the classroom. This role is played by the teachers. Skilled and sensitive teachers have always known that they must provide the scaffold to initially assist children to understand what is required of them. A more critical aspect of this concept, however, is that by providing a particular type of instruction, which might be called mediated instruction, and which is often termed mediated learning, it is the teacher who is able to help students to extend their performances beyond the levels they could reach independently. In short, only teachers (and in some cases more experienced peers) can help students cross the zone between what they know and what teachers know they could do with assistance.

Mediated instruction does not mean the teacher is there to direct children to execute specified tasks. Rather the teacher serves as a facilitator in helping the child reflect and think about how he/she arrived at an answer, or how they planned and carried out their ideas using the turtle. In this framework the role of the teacher is critical, especially with young children, because the teacher is the one who initially helps the student construct meaning out of an activity.

The need for collaborative interaction between students and teachers, i.e. the need for a shared dialogue about learning, is one piece that was missing in Papert’s early writings about the use of Logo. He relied too much on a stand alone model of human/computer interaction derived from the ideas of artificial intelligence (AI). In the AI models, children learn all they need to know from highly interactive machines simply by engaging in the creation of microworlds.

Although the quotations cited earlier from Papert’s (1987b) article suggest that he recognised that Logo must be seen as a cultural process, his precise interpretation of the meaning of this is not clear. In contrast, it is clear that he saw the teacher as having a peripheral role, if any at all. Papert is not alone in this view. After reviewing the literature, Olson (1988) noted that most writing about computers in the classroom is dominated by the AI view that teachers are a hindrance to technological progress, and that the mindlessness of traditional teaching will be replaced by students engaged in stimulating problem solving activities alone at the computer. Olson disagrees with this proposition:
Talk of thinking skills, of autonomy, of discipline, does not consider how school subjects become meaningful for children. Is not the teacher the very resource students need to construct meaning, because teachers are intelligent? Teachers really can talk back. Furthermore, school subjects provide teachers with tools for creating meaningful contexts for learning. Is talk of decoupling learning from teachers simply to talk about making learning less meaningful? These are the issues that emerge from thinking that teachers can be supplanted and school subjects transcended. (Olson, 1988, p. 9)

The computer is no substitute for the individual, experienced teacher. Rather, it offers a number of new teaching opportunities. In order to be effective, learning with computers requires the presence of a teacher to monitor the performance of individual students and provide both directive and non-directive support.

Approaches to professional training which view the teacher solely as a bystander, as a technician or as a mere consumer of curricula that others design, are likely to be completely inadequate for the preparation of teachers for classrooms in which students learn with computers. More importantly, such approaches are unlikely to provide teachers with a significant professional role in shaping the future technological transformations of schools, which will be adopted by future teachers only in so far as they are meaningful and integral to their teaching situations. Teachers must be encouraged to become partners in the creative enterprise of curriculum (including software) development.

As is stressed in later chapters of this book, ideally, teachers will be the central participants in and builders of the future of technology in education, not solely the recipients of decisions made by others, either in the area of training or tool design. Teachers must be supported and encouraged to adapt educational computing to their own and their students’ purposes, to explore the ways in which technologies can alter what happens in the classroom, and to share what they do and what works with other teachers. Their influence should be felt in what is produced and marketed for schools during the process of software and curriculum development, not after.

Professional development programs must support teachers to shape and engage in experiments with technology in education. Some of the most imaginative and successful uses of computers in classrooms today have come from teachers who were willing to redesign learning activities to take advantage of the technology, or who have discovered new dimensions in the technology that could be shaped and revised for use in the classroom.

CONCLUSION

The use of personal computers in the pursuit of educational excellence in schools is a new territory, which means that there is too little background in the literature (both empirical and theoretical) to give answers to all the issues and questions raised here and in the body of this book. This should not discourage the reader, but rather become a compelling motive for deep reflection and consideration of the philosophical and pedagogical assumptions, educational goals and classroom practices in the use of personal computers in schools.

As will become evident, the experiment conducted in the SUNRISE classrooms at Coombabah Primary School has shown that personal computers can be successfully integrated into classrooms. The observations made during the 1991 school year give some powerful insights into what can occur when teachers teach students who have their own computers. This book discusses theoretical and practical issues which arise when computers are introduced, and opportunities offered to students and teachers through this technological innovation.

However, predicting the long-term outcomes of new developments is difficult in any field. During the past five years we have certainly witnessed important changes in the use of computers in schools, but what does this mean in terms of the potential implications of educational computing? We recognise that the computer is an enabling tool which facilitates the execution of many routine and non-routine processes in creative production, problem solving and learning. Computers can facilitate the achievement of many valued learning goals, but their role is not a simple one. It is not sufficient to buy a personal computer and software, place the student in front of the computer and have the computer work its magic. Their is no magic in computing: rather, because of its interactive aspects the computer allows the student a more active part in constructing knowledge than paper and pencil could. Understanding the impact of personal computers in classrooms means understanding the complex system of interactive relationships between people, situations, tasks, social and cultural processes, and the learning context of which the computer is an integral part.

As Simon (1987) pointed out, the success of a major innovation is dependent on and subject to the occurrence of a variety of concomitant events and conditions. Consequently, the impact of the personal computer on classroom learning and teaching cannot be assessed, or even considered, in isolation. Computing in the classroom cannot be disentangled from the cognitive, social and personal demands of the curriculum goals and instructional tasks which teachers set for their students, or from the interests, motivations, skills, knowledge, abilities and difficulties which students bring to the learning situation. Understanding the impact of computers in education means understanding this complex dynamic system of variables as a whole.
Cognitive Effects for Students

The use of computers in classrooms in general, and projects such as the one conducted in the SUNRISE classrooms at Coombabah in particular, raise issues of the following kind: What exactly does computer technology offer the processes of education? What is unique about the function of the computer as a tool of the intellect? How does or should the technology, as used in society, influence what is done in schools with computers? How might information technology redefine the very possibilities of learning and teaching? How can our practices and our research contribute to the development of students and to effective curriculum design? Or just simply: What is the use of computers in the classroom? Some of these questions are analysed in this chapter and the next.

Computers are influencing, in a very fundamental way, the traditional organisation and definition of many jobs and workplaces in industry, business and the public sector. Workplaces are undergoing physical transformations as well as changes in job specifications and job category distribution (e.g. Noyelle, 1984; Cyert & Mowery, 1987), and new social interaction patterns are brought about by the electronic environment. Specific performance demands are also shifting, for example from manual to sensory discrimination, and towards the processing of text, suggesting that technology is affecting basic psychological activity (e.g. Martin & Scribner, 1991). The same is true for the classroom, i.e. the predominant work environment for students and teachers.

A general view is that the availability and use of computers increases people’s productivity and efficiency. Computers can extend, supplement and thus boost human performance. An alternative implication of computing is that it can provide a box of tools which are actually capable of changing the characteristics of problems and learning tasks, and hence somehow lead to a restructuring of the processes of learning and problem solving more generally. This latter attribute of computing might well contribute new visions of the potential cognitive benefits of the technology.

The question arises: Might the learning of computing have positive effects on students’ problem solving and reasoning? To answer this question, two important and related questions need to be investigated: Which components of instruction in computing could result in improvements in problem solving and reasoning skills? Which sequential steps of mastery of computing skills might lead to such improvements? Some reflections on the relationship of problem solving and computer programming, and a discussion of an initial set of steps towards an elementary level of mastery in programming, described in this chapter, are a preliminary attempt to deal with these questions.

Early Research

Two major influences appear to have contributed to the belief that programming may spontaneously discipline thinking. The first is from artificial intelligence, where constructing programs that model the complexities of human cognition is viewed as a way of understanding these complexities of human behaviour. The contention is that in explicitly teaching someone (or the computer) something, one learns more about one’s own thinking. Papert (1972a) postulates that through programming students would learn about problem solving processes by means of the necessarily explicit nature of programming, i.e. as they articulate assumptions and precisely specify steps in their problem solving during programming. The second influence is the widespread assimilation by educationists of constructivist epistemologies of learning, most familiar through the work of Piaget (e.g. 1970, 1972, 1973). Papert (1972a, 1980) has been a strong advocate of the Piagetian theory of knowledge acquisition through self-guided problem solving experiences, and has extensively influenced views relating to the benefits of learning to program through learning without curriculum in a process that takes place without deliberate or organised teaching (Papert, 1980, pp. 27-28). It should be noted here that Piaget is not advocating the elimination of organised teaching in schools.

For a long time there appeared to be a considerable gap between the rhetoric about the overt and latent benefits of educational computing and classroom reality. Grand claims have been made about the potentially positive implications of educational computing as a tool of the intellect. Some of these claims are supported by evidence gained from small scale studies conducted under almost ideal circumstances, such as teaching by enthusiastic experts who have generous resources. More substantial claims for the effects of learning programming on thinking are exemplified in the writings of Papert & Feurzeig (e.g. Feurzeig, Horwitz & Nickerson, 1981; Feurzeig, Papert, Bloom, Grant & Solomon, 1969; Goldstein & Papert, 1977; Papert, 1972a, 1972b, 1980; Papert, Watt, diSessa & Weir 1979) concerning the Logo programming language, although such claims are not unique to Logo (cf. Minsky, 1970; Nickerson, 1982).

Ross & Howe (1981) have translated Feurzeig, Papert, Bloom, Grant & Solomon’s (1969) four claims for the expected cognitive benefits of the development of mathematical thought to the learning of computer programming and proposed:
that programming provides some justification for and illustration of, formal mathematical rigor;
2 that programming encourages children to study mathematics through exploratory activity;
3 that programming gives key insights into certain mathematical concepts; and
4 that programming provides a context for problem solving, and a language with which the pupils may describe their own problem solving. (Ross & Howe, 1981, p. 143)

Papert (1972b) argued for claims (2) and (4) by noting that writing programs of turtle geometry is

a new piece of mathematics with the property that it allows clear discussion and simple models of heuristics [such as debugging] that are foggy and confusing for beginners when presented in the context of more traditional elementary mathematics. (p. 252)

He provides anecdotes of children making spontaneous discoveries relating to phenomena such as the effects of varying numerical inputs to a procedure for drawing a spiral on the shape of the spiral. He concludes that learning to make these small discoveries brings the child closer to mathematical thinking than being taught new mathematical concepts. Papert (1980) discusses the pedagogy surrounding Logo and he argues that cognitive benefits will emerge from taking powerful ideas inherent in programming, such as recursion and the concept of a variable, in mind-size bites.

Feurzeig, Horwitz & Nickerson (1981) provide an extensive set of cognitive outcomes expected from learning to program. They argue that the teaching of concepts related to programming can be used to provide a natural foundation for the teaching of mathematics, and indeed for logical and rigorous thinking more generally.

More Recent Findings

There is a considerable body of opinion which focuses on the real contributions computers can make to education, tempered with an awareness of the barriers to educational change which are met by all educational innovations, and the likely slow rate of progress.

In a synthesis of the results of thirteen quantitative reviews of research into the benefits of computer-based instruction. Niemiec & Walberg (1991) covered more than 250 individual research studies and showed the typical and average effect of computer-based instruction to be that it raises learning outcomes by .42 of a standard deviation. Although differential effects were noted by the authors of this analysis, they found that the overall effect of computer-based instruction placed the students in computer-based instruction at approximately the 66th percentile of the control group distribution.

Other relevant research projects, many of them conducted at Bank Street College, New York, include studies of the development of problem solving and planning skills in Logo programming (e.g. Kurland & Pea, 1985; Pea & Kurland, 1984; Pea, Kurland & Hawkins, 1985), concern with cognitive demands and consequences of learning programming (e.g. Clement, 1984; Kurland, 1984; Kurland, Clement, Mawby & Pea, 1987), classroom uses of software such as database management systems and word processors (e.g. Freeman, Hawkins & Char, 1984), investigations into how teachers' interpretive frameworks for software are linked to how they reorganise classroom learning with new technologies (e.g. Hawkins, 1983; Hawkins & Sheingold, 1987; Sheingold, Hawkins & Char, 1984), and formative research aimed at the creation of computer or multi-media instructional packages of an open-ended nature for student learning in mathematics, science, languages and technology (e.g. Char, Hawkins, Wooten, Sheingold & Roberts, 1983). In these studies the term problem solving has a broader and deeper meaning than its often restrictive educational association with mathematics implies. Computer environments make it possible for students to experience some of the deeper ideas that underlie a correct understanding of what human problem solving entails.

Some of the claims made in the literature suggesting that learning to program can be expected to bring about fundamental changes in thought are summarised below. Expected improvements include:

* rigorous thinking, the development of precise expression, and a recognised need to make assumptions explicit (because programs operate on the basis of specific algorithms);
* an understanding of general concepts such as a formal procedure, variable function, and transformation (as these are used in programming);
* the general idea that one can invent small procedures as building blocks for the gradual construction of solutions to large problems (as programs are composed of procedures, templates, etc.)
* greater facility with heuristics, explicit approaches to problems useful for solving problems in any domain, such as planning, finding a related problem, solving the problem by decomposing it into parts, etc. (because programming provides highly motivating models for the use of heuristic concepts);
* the general idea that debugging of errors is a constructive planning activity applicable to any kind of problem solving (because it is so integral to the interactive nature of the task of getting programs to run as intended);
* generally enhanced literacy and metacognitive awareness with respect to the processes involved in solving problems (due to the practice of discussing the process of problem solving in programming by means of the language of programming concepts);
* enhanced recognition for domains beyond programming that there is rarely a single best way to achieve a goal, and an understanding that different ways have comparative costs and benefits with respect to specific goals (learning to distinguish between process and product).
Clements & Gullo (1984) advance the following exploratory hypotheses in relation to possible cognitive benefits of computer programming:

1. In Logo programming children invent, construct and modify their own projects; therefore, Logo programming might facilitate divergent thinking. Because Logo is designed to encourage children to reflect on how they think, programming should lead them to develop metacognitive abilities, especially the ability to realize when they do and do not understand instructions.

2. Similarly, Logo programming may develop reflectivity in children as they think about their errors and how to correct them.

3. If computer programming can allow children to master ideas formerly thought too abstract for their developmental level, it may accelerate cognitive development, including operational competence.

4. Finally, because Logo programming involves giving explicit spatial commands, it should increase children’s ability to describe directions from their own and others’ perspectives. (Clements & Gullo, 1984, p. 1052)

It is possible, of course, that any benefits derived from computer programming can be attributed to the interactive nature of computing, rather than to the programming activities per se. It would therefore be necessary to provide a control group with computer experience not involving computer programming. Such experience might consist of using software containing word processing, ready made databases and spreadsheets, or they might consist of computer assisted instruction (CAI). Clements & Gullo (1984) set up such a study.

CAI has its roots in programmed learning and thus has a strong connection to the behaviourist tradition. Emerging from three themes of learning theory, i.e. individualisation, behavioural objectives and educational technology, many CAI programs employ the approach of programmed learning. Thus, they share the following characteristics: (a) they store a sequenced series of steps, often providing alternative learning paths for individuals, (b) they offer independent pacing for individuals, (c) they provide students with controlled contingent reinforcement, and (d) they can evaluate performance quickly and accurately to provide feedback on the degree of mastery.

Clements & Gullo (1984) compared a Logo group and a CAI (control) group of students and the effects of each on 3rd grade children’s cognitive style (including reflectivity, divergent thinking, etc.), metacognitive ability, operational competence, and overall cognitive development. The two groups did not differ significantly prior to treatment in the language and cognitive domains as measured by the Peabody Picture Vocabulary Test-Revised Edition, PPVT-R (Dunn & Dunn, 1981). The study revealed significant pre- to post-test differences on the Tests of Creative Thinking (Torrance, 1974) for the Logo group on fluency and originality, as well as on the overall divergent thinking score, while no significant differences were found for the CAI group. They also found that in the Logo group the latency time increased and the number of errors decreased. The Logo group significantly outperformed the CAI group on two metacognitive tasks. The ability to monitor one’s own thinking and realise when one does not understand are likely to be influenced by programming environments in which problems and solution processes are brought to an explicit level of awareness, monitoring and subsequent modification. Through consistent feedback in the form of visual representation of the procedures and sequences of their own thinking processes, the Logo students may have learnt how to monitor these processes.

The scores on a test of describing directions were similarly affected. The Logo group, being face to face with the turtle, practised orienting themselves to the turtle’s visual perspective. This skill is a prerequisite to the successful completion of the Describing Directions Test in the Kaufman Assessment Battery for Children, K-ABC (Kaufman & Kaufman, 1983). The directions of this test involve left-to-right and top-to-bottom reversals. Research has shown that with practice children improve on visual perspective taking (Flavell, 1977; Donaldson, 1978). No difference was found between the groups in two areas of cognitive development -- operational competence (classification and seriation) and other aspects measured by the McCarthy Screening Test (McCarthy, 1978).

Linn & Dalbey (1985) showed in a study involving over 500 pre-college students in 17 classes that the form of instruction, the access to computers, and the ability of the student influence outcomes from programming instruction. Specifically, exemplary instruction was found to move students further towards mastery of component skills than do less effective or as described by Linn & Dalbey, typical methods of instruction. Furthermore, both access to computers and the general ability of the students were found to be related to progress in typical classrooms. In exemplary classrooms, for medium and high ability students, neither ability nor computer access outside of school were found to be related to programming performance.

As noted previously, there is a widely held belief that computers will influence how effectively we accomplish traditional tasks, supplementing or extending and thus boosting human cognitive capabilities. In the assumption that the tasks stay fundamentally the same. The central point made in the context of the Coombabah classrooms and other projects where children learn with computers, rather than about computers, is quite different. Here, the primary role of computing is seen as one of changing the tasks and what teachers and students do, thus reorganising or restructuring tasks and possibly mental functioning, not only by extending and supplementing it. The predominant use of computers in schools today is with software that aims to make long-familiar drill and practice activities, especially in mathematics and language, more efficient and effective. Why not focus instead on using software as tools to support and restructure the student’s thinking?

**COMPUTER PROGRAMMING AND PROBLEM SOLVING**

At the core of computer programming is that set of activities, involved in the development of a re-usable product, consisting of a series of instructions, which
make the computer accomplish a given task. As is the case in more general theories of problem solving, cognitive studies of programming reveal a set of distinctive mental activities that occur as computer programs are developed. Some of these activities are involved throughout program development, irrespective of whether the programmer is an expert or a novice, because they constitute recursive phases of the problem solving process in any context and theory of problem solving (e.g. Heller & Greeno, 1979; Newell & Simon, 1972; Polya, 1957; Raahem, 1974; Rowe, 1985, 1991a). They may be summarised as follows:

1. understanding and defining the programming problem,
2. planning or designing a programming solution,
3. writing the programming code that implements the plan,
4. comprehension of the written program, and
5. program debugging.

Planning

One of the claims made about the positive effects of programming on thinking has been in the area of planning. The assumption is that programming experience will result in greater facility with heuristics, explicit approaches to problems useful for solving problems in any domain. It is possible, however, that students who can think logically, plan, and have acquired reasonable problem solving heuristics develop programming skills.

One may raise the objection that it is possible to bypass planning in program development, i.e. one might first make an initial reading of the problem and then compose code at the keyboard to accomplish the task. Although planning as one proceeds is certainly possible in the production of some programs, it seems likely that such attempts might create problems for the inexperienced programmer. While expert programmers can draw on their knowledge of a vast range of plans when creating a new program, the novice programmer has neither the sophisticated understanding of programming code nor the experience of devising the successful programming schemas which are necessary for engaging in planning as they proceed.

Planning can be characterised as a process of revision. As a consequence of considering alternatives, effective planners revise their plans. They alternate between top-down planning strategies, which create a plan from successively refining the goal into a sequence of subgoals for achievement in sequence and bottom-up planning strategies, which note the emerging properties of the plan or the planning environment and add data-driven decisions to the plan throughout its creation (e.g. Hayes-Roth & Hayes-Roth, 1979; Pea, 1982).

In the Coombabah project, most children appeared to do little planning in their programming work. Planning before writing an essay was insisted upon by the teachers, pre-planning before programming was mentioned by some teachers but not insisted upon, and explicit pre-planning aids (e.g. worksheets) were not provided. Students appeared to write and revise their code in terms of the immediate effects that commands and sequences of commands produced.

Pea, Kurland & Hawkins (1987) found that students who had spent a year programming in Logo did not differ on various developmental comparisons of the effectiveness of their plans and their planning processes from students who had not learnt to program computers. They concluded that learning thinking skills and how to plan well are not intrinsically guaranteed by the Logo programming environment. Rather, the development of planning skills must be supported by teachers who, tacitly or explicitly, know how to foster the growth of such skills through judicious use of examples, student projects and direct instruction. This is in contrast to the Logo instructional environment which Papert (1980) offers to educators, which is devoid of curriculum, and lacks an account of how the technology can be used as a tool to stimulate students' thinking about such powerful ideas as planning and problem decomposition.

Teachers are told not to teach, but are not told what to substitute for teaching. Thinking skills curricula are beginning to appear, but teachers cannot be expected to induce lessons about the power of planning methods from self-generated product-oriented programming projects.

(Pea, Kurland & Hawkins, 1987, p. 196)

Cohen & Feigenbaum (1982) define problem solving as a process in which a sequence of actions is developed to achieve some goal. These authors understand planning to mean deciding upon a course of actions prior to initiating the actions. A plan may consist of an unordered list of goals or an ordered set of goals (first do this, then do that, etc.). In a sense most of the research in artificial intelligence can be subsumed under this broad concept of problem solving research.

Basic Skills and Their Application

In the development of human minds two broad classes of activity are of particular importance: (1) activities which serve to equip children with a toolkit of basic mental skills, and (2) activities which require the application of those skills in generalised problem solving. Whereas the toolkit of the educated student will include specific arithmetic and language skills as well as more general cognitive skills, such as the ability to question and to categorise, the more widely based activity of problem solving requires the manipulation of information through the use of combinations of these and other skills. The overall aim is to provide students with a possibility of improving their thinking abilities. This is achieved by providing them with the basic skills, i.e. the cognitive toolkit, and with experiences in the use of different combinations of the components of their toolkit in problem solving exercises.
Because programming involves solving problems, it is generally assumed that learning to program, at least superficially, leads to improved problem solving skills in students. Hence, in most situations where programming is taught, teachers expect at the same time to improve students' problem solving and reasoning skills. In his seminal work *Mindstorms*, Papert writes:

> In my vision, the child programs the computer and, in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building. (Papert, 1980, p. 5)

In contrast, outcomes from actual programming instruction often fall short of these expectations (e.g. Pea & Kurland, 1983; Dalbey & Linn, 1985). Furthermore, cognitive research has shown that problem solving abilities appear to be much more subject or discipline specific than had first been thought (e.g. Larkin, McDermott, Simon & Simon, 1980; Linn, 1985a; Resnick, 1983). Thus, the mechanisms that might lead to generalisation of problem solving ability from one discipline to another require specification.

Programming a computer is a form of problem solving. Superficially, at least, students who learn to program acquire reasoning skills and learn about problem solving. Furthermore, certain features of the computer learning environment demand rather complete and accurate solutions to problems. For example, computers require precise input because they only respond to a limited set of commands. Programmers must decompose complex problems into subproblems and then generate sets of step by step instructions to solve the subproblems. In addition, the learning environment of computer programming is interactive in that by testing the problem solution, the problem solver can receive feedback about how effective the solution is and can use this information to modify the solution. Thus, writing a program requires the student to explicitly use some potentially powerful problem solving skills.

Programming is also compelling as a vehicle for teaching problem solving because many students are highly motivated to use the computer (e.g. Lepper, 1985). Programming sessions frequently allow students to try out procedures, even when students have little idea of how they will use them. Lepper notes that, among other things, the precision and interaction of the environment as well as the potential for challenge and fantasy appeal to students.

Computer programming, therefore, seems ideal for encouraging problem solving. But will the problem solving skills generalise to other areas of learning? Cognitive research has shown that learning is much more discipline-specific than had first been thought (e.g. Larkin, McDermott, Simon & Simon, 1980; Resnick, 1983; Scribner & Cole, 1981). When students learn to solve physics problems they learn about physics but not necessarily about problem solving in general. Thus, when students learn to solve programming problems they may not automatically or necessarily learn to solve other problems.

### Acquiring a Toolkit of Knowledge and Skills

The perception of a shrinking world is brought about largely by technological change. Although it is not a new phenomenon, the exponential increase in information and the associated accelerating rate of change in the latter half of the twentieth century has increased the feeling of a shrinking world. Caught in an information deluge from around the world, we must look for new tools to help us cope. We need bigger memory stores, faster and more accurate retrieval systems and, above all, the ability to sift and sort the information before we drown in it.

The computer is a tool designed to extend our overstretched human capabilities. It can be argued that our future economic success depends on the degree to which children are taught to be sufficiently flexible and adaptable in their thinking and actions in order to handle the pace of change brought about by the information age. Gagne (1970) has argued that the most important things learnt in schools are intellectual skills, not pieces of verifiable knowledge. This is the distinction between declarative knowledge, i.e. to *know what*, and procedural knowledge, i.e. to *know how*, both of which are central to the idea of a generalised cognitive toolkit. However, it is the acquisition of procedural knowledge which we should be looking for in classrooms that are dedicated to flexible problem solving, even though Longworth (1981) paints quite a gloomy picture. He considers that what today's children learn at school, at best, has a useful life of half a generation, while at worst it is obsolete as it is being taught. Competence with a variety of new technologies and their applications will become more useful as we enter the *hole in the wall* society in which not only cash but also information will be dispensed only to those who know how to gain access to it.

There is strong evidence that the current emphasis in classrooms is upon factual knowledge combined with a very limited structural organisation of information, rather than upon information processing skills. In the area of computer-based learning, drill and practice programs have so far tended to dominate. Students and teachers are being tyrannised by curricula which fossilise information into facts to be known, rather than into material to be manipulated and thought about. Even powerful, open-ended tools such as word processors can be used in a fossilised education if they are viewed as neat typewriters rather than tools for text manipulation and information handling which allow us to explore our own thoughts. A major aim of teaching computing in schools should be to help our students to acquire a toolkit of cognitive skills.

The value of the cognitive toolkit is that it frees the mind of its user for more general problem solving activities. Children have little difficulty in acquiring the subskills which form part of the toolkit, although the integration of relevant skills may be another matter. The educator has the choice of which subskills to emphasise, of course, and how to present them. Reciting multiplication tables, or writing row after row of well formed letters of the alphabet presents the child with subskills for his or her toolkit. Many students are not motivated by this form
of learning. Also, it is questionable whether these subskills should receive such strong emphasis. They contribute little if anything at all to the development of flexible problem solving skills. They are part of an educational philosophy which emphasises knowing what rather than knowing how, i.e. they assist the development of declarative rather than procedural knowledge. Papert (1980) points out that personal computers can be used to simulate environments which provide children with the conditions necessary for the reorganisation of their understanding of phenomena concepts, etc., and he argues that children can acquire powerful problem solving tools very effectively by learning to program. By programming the computer to create graphic displays, abstract ideas can be made concrete, and the means of manipulating the world can be made personal and apparent.

Computer-based activities which can be used to facilitate the development of the toolkit include the creation and interrogation of databases, the use of word processing packages and Logo programming. The feature common to these tools is that they are open-ended. In the sense that pencils, pens and typewriters are open-ended. I would suggest that the most useful pieces of educational software are open-ended tools in that they do not provide right or wrong answers but opportunities for the development and exploration of ideas. The aim of these activities is not an end in itself in most cases, but to provide general skills which can be used in the solution of other problems. Component subskills of problem solving activities are performed without much effort and attention because they are highly practised. These over-practised skills are valuable, and computer programs which facilitate the development of a cognitive toolkit of subskills are to be welcomed, but they must be chosen with care. Practice does not necessitate drill and practice, and subskills can be acquired in the context of problem solving activities such as simulation games and Logo programming.

The claim made for having children learn to program is that, through interaction with the physical microworld, they will acquire a toolkit of general problem solving subskills. Programming is a specific form of problem solving -- e.g. the problem might be one of how to get the computer to draw a spiral, or a row of houses -- and the act of programming itself requires the use of a set of subskills which are independent of any body of facts about the programming language itself. Arguments supporting the view that certain subskills, which might be developed by learning to program, will improve cognitive processing and performance more generally include the following:

- Programming requires creative and critical thinking to be made explicit. There is no magic button when it comes to giving instructions about the movement of the cursor: it moves exactly as instructed and makes no assumptions of its own. Imprecise instructions are not recognised, and reformulation of imprecise commands is necessary.
- Programming provides an environment in which general concepts, such as variable, function, transformation, and recursion, can be learnt and their consequences demonstrated. These powerful abstract concepts can be seen in operation through programs. Logo programs are particularly valuable because of their graphical output which can be said to make thinking visible.
- With problem solving through programming, it is possible to appreciate the usefulness of heuristic approaches to a solution. These are the general problem solving skills involved in planning the route to a solution, solving problems by breaking them into smaller parts, and solving problems by analogy. In particular, programming can foster the idea that problem solving can be organised by parts. Small procedures can be seen as the building blocks from which large solutions are derived.
- The interactive process of getting a program to run as intended gives an appreciation of debugging a less than desired solution. Errors become important and helpful aids to learning. They can be informative as a diagnostic means of locating the source of a student's difficulty with a task. The strategy of using errors as a starting point for remediation and improvement can be generalised to other problem solving tasks.
- The acquisition of the vocabulary of programming made necessary not only as a tool for thinking but also as a tool for communication with others -- i.e. to openly discuss the process of problem solving during programming -- leads to stronger awareness of the processes of problem solving. Reflection becomes a means of selecting and giving strength to the control processes which are necessary in the choices between alternative routes to a solution, and in the reviewing of resources necessary for problem solving. This reflectiveness and awareness of process has been labelled metacognition, i.e. knowledge about one's own personal cognitive processes and others', their limitations and their applicability. Awareness of our ability to handle problems includes the knowledge of the ways in which we know ourselves to be capable of solving tasks and knowledge of the kinds of activities in which we must engage in order to find a solution. An important part of this awareness of problem solving strategies is the recognition that individual problems call for individual solutions. The selection of the most appropriate solution will depend upon cost/benefit analysis of the alternatives by the individual in a particular context.

Such educational effects of learning to program are the powerful ideas to which Papert (1980) refers. They are powerful tools because once they have been acquired they can be used over and over again. They are powerful in the same way as being able to read is a powerful tool, because it allows the individual to read a variety of materials for a variety of purposes and with important consequences. They are also powerful in another sense, because by making the suggested effects so explicit they have been made testable, i.e., we can evaluate them.

The essence of the claim made here is that programming trains people in problem solving. The programmer must be able to formulate goals and routes to
the goals precisely, to be able to apply a number of heuristics which apply to many types of problems, and to appreciate the usefulness of diagnostic errors. Once these powerful ideas have been acquired through programming experience, they can then be applied to other kinds of problems. They are generalisable tools which can be applied to a variety of situations with and without computers.

Few of our students will need, as adults, to be able to program anything more sophisticated than a washing machine or a video recorder, and there are very few advocates of programming as an educational discipline in its own right. The benefit of learning computer programming lies in the experience it provides of a range of concepts which are a useful addition to our students’ cognitive toolkits.

Papert (1980) goes further than this, however, and points to Logo programming experience as a simulation of the kinds of interaction with the world which can foster cognitive development. He suggests that the reorganisation of young minds occurs through the discovery of regularities in the world and through the testing of hypotheses about its structure. By exploring a computer-based microworld, students can discover the regularities of that environment, and will discover the effects of their own actions upon the microworld. These interactions will produce successive reorganisations of the child’s understanding of how the world works. Programming could, thus, accelerate a child’s normal course of cognitive development.

Chandler (1984) is particularly concerned that children should not only be aware of, and have access to, national and international databases, but that they also need to be contributors to the store of knowledge they contain. He argues that guided tours of someone else’s frame of reference are not enough. One of the most effective ways of understanding any body of knowledge is to reconstruct it. In some cases this corresponds to the building of a physical model, or of a computer model with, for example, Logo graphics, and in other cases this may be the redescription or adaptation of a body of knowledge with a personal database. Children need to create their own systems for communicating information with one another. otherwise they will become passive, if not alienated, consumers of the knowledge of others. Papert defined this as the power principle. The learner must be empowered to perform personally meaningful projects. The premise that the child’s cognitive performance will improve over a wide range of measures, if the educational experience builds upon the child’s own experiences, is one which finds ready acceptance among practising teachers and theorists alike. Papert (1980) described this as the continuity principle.

**Major Toolkit Applications**

One might focus on three applications of problem solving in which the toolkit provided by computing is of particular relevance. In the first application the use of computers provides a vehicle for direct problem solving. The second application arises when the computer is used as a tool for creating an object, idea, system, etc. The third application involves problem solving processes in order to find new uses of computers by students. The three areas of application are discussed in further detail in this section.

1 **Direct Problem Solving** Some examples of the use of computers for direct problem solving are the use of Logo, adventure games and using the computer to control and monitor external devices. The identification and use of strategies in computing can make it clear when students have engaged in different levels of problem solving activity. It can also help students to invent their own problems and adventures. A useful kind of activity for this is Lego-Logo.

Using Lego-Logo, students in the Coombabah project have built, for example, cars which can go backwards and forwards, a barrier that goes up and down, traffic lights which change, a drink machine that gives change, etc. The motor for each model was controlled by the computer through relatively simple Logo programming. For example, students worked towards achieving a sequence where the car moved up to the barrier, which would then rise, allow the car through, and then lower again. The students had to work through four major stages: (1) to comprehend the requirements of the task, (2) to identify and use the combinations of switch commands needed to operate the car and barrier, (3) to convert these into flexible subroutines, and (4) to relate these subroutines to analyses of the distance, times, etc., needed to solve the original problem.

The students structured each of these stages using processes which involved activities ranging from looking at their brief user guides, through trying out possible combinations of switches, to deciding on a sequence of ordered experiments. After solving the original problem, the students were able to provide observers with a detailed verbal description of the stages they had worked through. The feelings of one student are reflected in the following personal report:

> I think this project was about how to make things work [move]. It was interesting and fun. It was hard at first, because we had to find what every bit did. Then we had to think about what we were going to do and make a decision. Then we did lots of test runs and calculations. In the end we ended up with the car moving forward, a gate lifting, the car going under and the gate going down. (Personal communication of a Year 6 student)

The use of programming processes and an appreciation of levels or stages are very apparent in this student’s reflection on the problem solving to which he contributed.

2 **Creating Something** When the computer is used to create some object, idea, process, etc., it is important to alert students (and other users) to the constraints imposed by hardware, software or both. The use of any type of technology adds its own specific constraints to the realisation of particular outcomes. The properties of the machine and/or software have to be analysed as a part of the creative activity. The use of problem solving processes for this analysis has led to
interesting developments. In most instances, the students themselves have been able to articulate the way in which the use of the technology affected the outcomes of their activities. Some students have also been able to specify the concepts involved and what they have learnt.

The following example might illustrate this application. A group of students decided to produce a school newspaper by using a simple desktop publishing package. They learnt about the structuring of the production process by means of activities such as gathering information, analysing and evaluating it, deciding on what information to use and how to present it. However, these processes also led them to new and important knowledge and understanding about ways of using language. Two insights, in particular, were the direct result of the consideration of constraints related to the technology. One such constraint was lack of space, as the software was suitable only for articles up to the length of 60 words. This forced the students to consider the style of writing which would be suitable. They decided that they should use as few words as possible to convey ideas. One student coined the phrase punchy style to describe what she perceived to be required.

Considering the constraints of the technology led to the consideration of other constraints such as the need to sell copies of the finished product. This meant that the text had to be readable and easily understood by students at different year levels. Another constraint was that headlines were compulsory, so these had to be invented and designed to catch the eye of potential readers.

These constraints led to the students having to edit the work of other students who had submitted articles. The idea of students editing and rewriting the work submitted by peers would not have come up if it had not been for the constraints of the situation. Working in a problem-solving climate enabled the students to make decisions naturally, and also to articulate how they reached them. Upon task completion, the students were asked to reflect on their experience of producing the newspaper and what they thought they had learnt. Many of the comments related to setting out and other production components of the task, but a number of students reported that they felt that they had learnt to work more systematically, to plan and to think more clearly.

3 New Uses of Computers A third application involves using problem solving processes in order to provide an alternative approach to the use of computers by students. At present the introduction of computers into classrooms is often highly structured and prescriptive, with an emphasis on training rather than understanding. Many guides for the use of computers in schools, such as published user guides, provide step-by-step instructions without any explanation and discussion of the principles. The result can be a lack of flexibility for the learner because of a lack of basic understanding.

In a problem-solving climate, the use of new technology is seen as a situation to be dealt with by usual methods of finding information, thinking, making decisions, monitoring and evaluation. Students learning to use new technology expect to gain information both from the teacher and peers, and to a lesser degree from books and user guides. They expect to use trial and error methods for some details, but tend to look for general principles. The natural processes of enquiry and learning are thus perfectly appropriate, and can be expected to relate easily to using the features offered by the new technology in the classroom context.

One aspect of cognitive and personal development which will be of increasing importance in the future is the ability to be flexible in a rapidly changing society. One way of encouraging such flexibility is to place more emphasis on the use of higher order analytical thinking by students. The Coombabah SUNRISE Project and other opportunities for learning with computers are certainly helping to achieve this by fostering the bond between problem solving and computers in the classroom.

STEPS TOWARDS INITIAL MASTERY OF PROGRAMMING

An arbitrarily selected set of steps towards initial mastery of basic aspects of computer programming may actually suggest how links between problem solving in programming and problem solving in other domains may arise.

Such a set of steps for programming instruction might offer guidance for understanding what might constitute appropriate student experiences and also provide a standard against which to measure instructional methods. These steps are not intended to be an exhaustive list of all possible activities, but rather to identify some of the activities involving considerable cognitive skill. The steps describe a direction which instruction might take, but it must be remembered that gains in problem-solving ability occur slowly. Introductory instruction can start students off in the right direction, by making explicit the aspects of programming that are likely to generalise to other domains. Table 2.1 summarises and describes some possible steps.

Language Features

In order to be able to use the programming language being studied, it is important to understand many of the language features or non-decomposable elements of the language. In programming courses teachers typically introduce language features, explain how they work, and have students use them. The teachers in the SUNRISE classrooms at Coombabah explain the language features to the group as a whole, then demonstrate how they work to a small group who become experts. Rather than giving students formal practice in using the features, they are introduced to them and then encouraged to explore them for themselves. When difficulties arise, the students consult an expert and/or the teacher.

Students' knowledge of language features could be assessed by comprehension items which require the student to predict how programs using certain
Table 2.1 Possible Steps towards Initial Mastery in Programming

Step 1: Command of the language.

The features of the computer language are the primitives or non-decomposable elements of programming. For BASIC they include IF ... THEN, GOTO, PRINT, etc. For LOGO they include MAKE, SET, REPEAT, PRINT, TO, IF, etc.

Step 2: Skills to design programs.

This requires the acquisition of a repertoire of templates and procedural skills. Templates are fixed patterns of code (prototypes) using more than a single feature of the language. They are employed in programs to perform commonly encountered tasks. Templates can perform complex activities such as sorting or searching for words or numbers, and can be used in many different situations. For example, rather than inventing a solution each time sorting is required, a sort template is applied to each sorting problem. Procedural skills are used to combine templates or language features to solve a problem. They include planning a solution using available templates and language features, testing the plan to ascertain whether it accomplishes the objectives and reformulating the plan until it succeeds.

Step 3: Problem solving skills which are transferable.

What has been learnt could transfer to a new programming language, a database management system, a computer controlled device, or even a subject matter like motion in physics. One aim is to develop a repertoire of generalisable templates suitable for adaptation to other tasks and contexts. Another aim is to identify generalisable procedural skills for planning, testing and reformulating problems in a variety of situations. This involves identifying the isomorphism between procedural skills used for several types of problems.

Design Skills

Design skills are the group of techniques used to combine language features to form a program that solves a particular problem. These include templates and procedural skills. Design skills are essential in order for students to write computer programs.

Templates

Templates are fixed patterns of code that use more than a single feature of the language. Templates perform complex functions such as sorting alphabetically and/or numerically, counting the number of words in a text, executing basic arithmetical functions, etc. Templates can be stored, called up and used each time a given task is encountered. They perform a function similar to schemata, and more specifically weak schemata in the theoretical formulation of Anderson (1984) and to plans in the work of Soloway & Ehrlich (1984).

When students have a repertoire of templates, they have a set of flexible and powerful techniques which allow them to complete many tasks without inventing new code. Well chosen templates facilitate good programming, because they can help to reduce the cognitive demands of programming by providing obvious ways to decompose a task. For example, in Logo and in a number of other languages, students can form a type of looping template that combines a decision based on the IF ... THEN language feature with the GOTO feature. A specified action would be performed as long as the decision is affirmative. Examples of problems which could be solved by using this type of template are as follows:

- read names if the last one has not been located.
- read scores which are smaller than 50 (< 50).
- add water if the pool has less than 10,000 gallons in it.

This type of template offers a good way to manage flow-of-control, i.e. the order in which the computer executes statements in a program. Programs with sequential and organised flow-of-control are easy to understand and revise. Students can decompose tasks into pieces of the size of their available templates.

The theory is that if one has a strong schema, comprehension is principle driven and predictions can be thought of as derived. With a weak schema comprehension is precedent driven. Predictions are not so much derived as looked up, and generalisations are local in scope and treated with caution. Anderson's hypothesis is that the notion of a weak schema gives the best account of the thinking of ordinary people in ordinary circumstances dealing with ordinary matters of knowledge.
Then they can complete the task, by solving the problem using their templates. Students who have a repertoire of templates can write more complicated programs than those without such a repertoire.

Experts structure their knowledge of programming in templates (e.g. Atwood & Ramsey, 1978; Kurland, Mawby & Cahir, 1984). As research in cognitive psychology suggests (e.g. Chi, Feltovich & Glaser, 1980), the way knowledge is structured and organised determines how it will be used subsequently. If programmers develop a repertoire of effective templates they are likely to program more effectively, and to identify and re-use algorithms for problem solutions rather than re-invent solution paths over and over again. Students need to recognise the importance of templates and to collect a powerful set of them.

Templates can be learnt from direct instruction. Some computing textbooks emphasise their usefulness as a mode of acquiring and and organising programming knowledge. However, only a small number of the students in the Coombabah project look for books on computing in the library. They do not make use of templates as frequently as they could. Although this group of students may not be typical of other students learning with computers, it must be emphasised that teachers should motivate and encourage students to look not only at one another's programming but at books containing new ideas. Many expert programmers report that they learn new templates by reading programs written by others.

Students can often employ templates which they could not invent themselves. Most of the Coombabah students know this. By using these templates they can profit from the work of expert programmers and themselves solve more interesting and complex tasks than would be possible if they had to invent all their own templates. Even if they do not fully understand a template initially, the experience of using a well designed one will help students comprehend the technique in the template and the role of templates more generally. Just as experts learn templates from others, so can novices, thus increasing their template repertoires.

**Procedural Skills**

Procedural skills are used to combine templates and language features, which are available to the programmer, in order to solve a new problem. Procedural skills include planning the solution path, testing the plan and reformulating the plan if any of the tests fail. Reformulating, previously mentioned as a technique for testing knowledge of language features, can also be used to modify longer sequences of code.

Programmers need a plan for combining language features and templates to solve a programming problem. They decompose the problem into component parts and plan how best to combine those parts. Once a plan is implemented, the programmer needs to test the plan in order to ascertain its correctness. Testing involves determining whether a program meets specifications by deciding what data or other conditions might cause difficulty and then running the program under those conditions to see whether it operates correctly. When the testing of a program reveals problems, the programmer decides whether it requires refinement. Programmers reformulate programs to make them more adequate.

**Planning**

Planning is required for the solving of complex tasks. Novices rarely work on programs complex enough to demand planning. Programming assignments which involve only linear combinations of single language features often fail to illustrate the advantages of planning. Thus, programming instruction must be carefully designed in order to ensure that students understand the importance of planning and have the opportunity to practise it. Only then can they gain knowledge about the conditions under which a template will function. Planning is an important component of the behaviour of expert programmers. In some studies, experts spend much of their time engaged in planning. In contrast, planning is not an aspect of novice behaviour (Dalbey, Tournaire & Linn, 1986). Similar differences in the time spent planning solution paths are reported for experts and novices solving non-computer problems in physics (e.g. Larkin, McDermott, Simon & Simon, 1980).

**Testing**

Testing is an important component skill of programming that can be enhanced by asking students to find out whether programs perform as expected or intended. Experts and novices differ in this skill. Experts not only recognise the advantages of testing their programs but are good at devising tests to reveal possible problems. For example, experts tend to test the boundary conditions, to ensure that no division by zero is possible, and to consider difficulties resulting from interactions between parts of their programs. In addition, programs written by experts tend to have built-in tests for potential confusions, such as tests to be sure that the input data meet the problem specifications. In contrast, novice programmers in Years 6 and 7 test only the obvious or usual forms of input and may fail to test all of the code.

**Reformulating**

Reformulating is required when students are required to modify a program plan. It is another skill that differentiates experts and novices. Experts are likely to respond to the results of tests by considering large scale as well as minor reformulations of their programs. In contrast, novices tend to seek localised remediation for their programs, perhaps never learning how to revise larger programs. These efforts of novices often result in what experts have called spaghetti code.

The acquisition of design skills can be assessed by asking students to write programs to solve tasks. To require planning, testing, and major reformulations, such problems must be cognitively demanding for the students. This means that the tasks must be reasonably complex, challenging and, where appropriate, have multiple solutions. To measure template acquisition, problems must require commonly learnt templates. Once a problem is solved, the program that solved it can become a new template. As noted above, expert programmers use these skills effectively, whereas novices often fail to plan their problem solutions, fail to test their programs and are unable to successfully alter the programs that do not perform the desired task.
Problem Solving Skills

The major potential gain in these steps towards initial mastery of programming consists of problem solving skills which could also be useful for the acquisition of new knowledge and transfer of prior learning to new contexts. These problem solving skills include both generalisable templates and generalisable procedural skills, i.e. templates and procedures which are common to many formal systems. They are generalisable in the sense that the characteristics applicable from one formal system to another are made explicit and can be applied to new systems. Learners who represent for themselves their programming knowledge and experience in such a way that the elements common to several programming languages are separable from those that are idiosyncratic to one particular language will be more likely to acquire generalisable skills.

For example, templates such as sorting in one programming language can often be used when programming in a new language. Generalising a sort template from one programming language to another may require substituting one type of looping for another. The generalised template is represented in such a way that the user knows that the looping structure needs to conform to the conventions of the new language.

Generalised procedural skills consist of techniques of planning, testing and reformulating which can be applied to several tasks and contexts. For example, there are many similarities between planning a solution to a computer program, an algebra word problem, and a geometric proof. Instructions which make these similarities explicit and provide opportunities to use these skills in more than one formal system may well facilitate the acquisition of general problem solving skills. As yet few programming courses offer opportunities to examine this possibility. Studies of experts suggest that transfers of knowledge and skill are achievable. The procedural skills of planning, testing and reformulating are applicable both in learning new programming languages and in learning to use other systems such as database management software, spreadsheets and word processors.

General problem solving skills may be acquired when students attempt to apply templates or procedural skills learnt in one context to a new context (e.g. Dalbeys & Linn, 1986). Also, students may identify aspects of templates or procedural skills that are central to their effectiveness as well as aspects that are peripheral to their effectiveness. This knowledge then becomes general enough to be used for problems in other programming languages and for non-programming problems. The acquisition of problem solving skills can be assessed by asking students to solve problems using an unfamiliar system such as a new programming language. The set of steps of cognitive accomplishments that culminate in technological expertise is a long one. Fairly complex problems are required before students can be expected to use cognitively demanding skills such as planning. Experience with several formal systems may be needed before students actually acquire transferable problem solving skills.

Making Learning with Computers Cognitively Demanding

Jerome Bruner (1966) argued that positive attitudes towards learning are encouraged by the complexity and challenge of the task in hand. Complexity encourages curiosity, and perceptual curiosity in turn generates a state of high arousal or excitement which is relieved by the exploration of the stimulus. Bruner’s challenge to educators is for them to find ways and means of fostering the drive to achieve competence at a task, and to create in the student the need to have mastery over the environment through at least some understanding of its complexity.

There is a general demand for higher order cognitive outcomes from schooling. Expressions of the desire to transform Australia into a clever country call for the new basics of the 21st century, i.e. thinking skills that allow individuals to cope with rapid technological, social and scientific advances, and with the accompanying philosophical, economic, cultural and personal changes. What is being emphasised in countless public statements and reports is the need for teaching and learning which will foster problem solving and can prepare students not only to deal with new technologies as they become available, but to make these technologies part of the student’s personal repertoire of intellectual tools. Learning with personal computers can help students towards these aims.

It is easy to mistake sophisticated technology for sophisticated learning, and it would be a mistake to assume that productive outcomes will necessarily emerge whenever people use computers. There are many examples of complex technology being used to achieve low level educational goals, and some uses of drill and practice programs are a case in point here. The computer is a versatile piece of equipment which can be used to promote sophisticated learning strategies in which the machine, the student, or both, take a more active part in the learning experience. It is this very versatility which raises fundamental issues about future directions of educational computing. The computer may be used to great effect as a calculator, a teaching machine a processor of complex information, a creator of microworlds or to control complex systems. In essence, this means that the computer can support a full range of educational philosophies, e.g. acting as a tutor for those who believe we should return to a traditional basic skills curriculum or as a key factor in stimulating the dynamic processes of creative writing, complex problem solving and other higher order cognitive activities.

All too often we find students whose main motive for preferring to learn with a computer is the ever-patient and generally non-judgmental response computing appears to present to their mistakes. The practice of discrete skills is not inherently wrong, in fact, practice is vital if skills are to reach the level of automaticity necessary to allow the individual to focus attention on higher level problems. If we must concentrate on the spellings of words and on the formation of letters with a pencil, then we have less time available to think about the
meanings of the sentences being composed. If the lower order skills are
automatised, the students' minds become free to plan, create and review. Tedium
routine activities then take care of themselves, but only when they have been
practised and overpractised. In the first instance, drill and practice serves the
purpose of releasing our minds.

Dede (1986) suggested that instructional control strategies for the use of
computers in education form a continuum based on the balance between varying
levels of passivity of the computer and the child. At one end lies the directed
learning in which students are passive recipients of wisdom and unable to explore
the material themselves. At the other end lie the open-ended computing tools
such as Logo, data bases and word processing. These problem solving tools give
control to the learner but no longer provide built-in guidance when the student
has difficulty, although they inform the user that an error has been made or an
inappropriate action has been taken. One end of this continuum will appeal to
those teachers who believe that there is a critical, generally agreed upon body of
existing prerequisite knowledge which all students need to be taught. The other
extreme will appeal to the teachers who believe in the need for the student to
discover his or her own truths, and to build up their personal knowledge with
varying degrees of support from the teacher.

Characteristics of Cognitively Demanding Activities

Several features of learning with computers are likely to increase the quantity and
quality of cognitively demanding activities offered to students. For example, the
Assessing the Cognitive Consequences of Computer Environments for Learning
(ACCCEL) project (Dabney, Tourniaire & Linn, 1986) has identified six features of
learning with computers which result in the capacity of such environments to
provide cognitively demanding activities. Three of these are characteristic of
many school environments, the other three somewhat unique to learning with
computers.

The first feature common to some traditional and some computer learning
environments is complexity. Computers can help students solve complex tasks.
For example, students can solve problems which require the management of large
amounts of information, such as plotting graphs or computing compound interest.
The second feature is challenge. The computer can challenge the student to solve
problems such as figuring out the best move in a game or determining the most
efficient path through a maze. The third feature is the provision of multiple
solutions to a question or problem. Students can write and compare several
programs which accomplish the same end. These three features are common to
computer learning environments but they are also characteristic of some
traditional learning contexts.

Three additional features of the computer environment are less characteristic
of other classroom learning: (1) the computer environment is interactive at all
times. The computer can respond immediately and informatively to the learner's
specific request or need. Thus students can try several approaches to
reframing their computer program and determine whether each of the
approaches is successful. In contrast, it can take days or weeks for students to
receive responses to their homework or class assignments. (2) Computer
feedback is precise. It reacts to all input. Without computers, students frequently
receive rather imprecise feedback, e.g. an A, B- or a C. (3) Computer learning
environments are consistent. The same response is received for all identical input.
Moreover, for identical input the same response is received by all learners. In
contrast, teachers do not necessarily respond identically to the same student
reactions, either because they are rushed or distracted, or because they are
tailoring their responses to the perceived needs of the individual student. When
teachers behave as good tutors, their tailored responses to the student provide
advantages not available in most computer-based learning. On the other hand, if
teachers are distracted, their inconsistent responses can be less than desirable than
those characteristic of computers.

Currently the most cognitively demanding activity readily available for
students on the computer is programming. This situation is changing as new
software becomes available. Eventually, software which demands higher
cognitive skills, but is free from some of the drawbacks of programming, may
well become preferable to programming for fostering higher cognitive skills.
Currently, however, school students who have cognitively demanding
interactions with computers are usually engaged in programming. In spite of this
situation, much current programming instruction, including that provided in the
SUNRISE classrooms at Coombabah, lacks a conceptual framework, and is not
necessarily geared to fostering higher cognitive skills (cf. Linn, 1984). Apart
from the software manual, teachers in Australia and other Western countries tend
to have textbooks for programming instruction: instead they often amass
materials somewhat haphazardly. Very limited funds have been available for the
much needed professional development of teachers in educational computing and
other areas of computer education, as many schools and regions have followed a
buy hardware now, plan for its use later approach. Before programming
instruction can achieve reasonable goals, this situation must be rectified. A
preliminary approach might be to identify some of the characteristic behaviours of
expert programmers, and to attempt to develop such behaviours in our novice
students.

Characteristics of Programming Experts

The contrast between good professional computer programmers and children
learning to program in school suggests a need for materials which can build up
knowledge and skills which might culminate in programming expertise. As
matters stand now, some staff teaching computing in tertiary institutions
complain that what students have learnt in school actually interferes with the ability of students to profit from tertiary courses in computing. Analyses of the nature of expertise, the characteristics of current instruction in computing in schools and tertiary institutions, and the potential of the environment in which programming instruction takes place, need to be conducted before the usefulness and relevance of school versus university and college computing can be established.

The behaviour of programming experts contrasts sharply with the characteristics of current instruction in educational computing, as revealed by several studies of experts (e.g. Pea & Kurland, 1984; Linn, 1984; Jeffries, Turner, Polson & Atwood, 1981). To develop the knowledge and skills which might be prerequisites to programming expertise, students must learn the skills experts use every day. Current instruction may not provide this opportunity.

One component of expertise is an extensive repertoire of programming templates or procedures. Templates can apply to a whole program as exemplified in an input-process-output template. As noted previously, templates can also apply to a specific function of a program. Research in the USA and at Coombabah has shown that both expert adult programmers and quite young students who are experts in some aspects of programming can articulate their templates, recognise the relationships between their templates and new templates, and actively seek to create new templates or procedures.

Professional programmers use a variety of procedural skills. As noted above, these are among the skills referred to as the new basics of schooling in many recent reports. They are part of the set of thinking or problem solving skills which individuals need to survive in our society. Important components of these skills are planning and the ability to determine an appropriate sequence of available procedures. In the past, investigations of expert performance in formal systems, such as solving mechanics problems in physics, have proved themselves to be informative for educators designing programs to foster these skills in novices and to make the learning of programming more cognitively demanding (e.g. diSessa, 1982, 1986, 1988a, 1988b; Larkin, McDermott, Simon & Simon, 1980).

The literature dealing with the planning of solutions to programming problems by experts indicates that experts engage in two complementary techniques: top-down design and stepwise refinement (e.g. Brooks, 1980; Atwood & Ramsey, 1978; Jeffries, Turner, Polson & Atwood, 1981). Top-down design is an approach which decomposes a complex problem into subproblems. Experts can do this effectively, we surmise, because they have a large repertoire of program templates. Experts use their knowledge of templates to guide the decomposition process. Top-down design is somewhat iterative in nature. After the initial decomposition, each resulting subproblem may require further decomposition until the task reaches a manageable degree of complexity. Experts proceed with top-down design by selecting appropriate templates for each problem.

**Stepwise refinement** experts engage in successive restatements of the problem specification, with each step coming closer to machine level notation. The original problem specification describes, in natural language, a process the computer is to perform. Stepwise refinement means to translate the process description, through incremental stages, into language, i.e. code, which the machine understands. Experts can do this well because they are very familiar with the language the machine uses. Experts know the degree of precision and the degree of clarity needed to describe the process for a machine solution. Ultimately they generate unambiguous statements of their program design.

**Characteristics of Students of Programming**

Students who are just beginning to learn a programming language usually differ dramatically from one another. There are many reasons for this which relate to differences in motivation and interest, previous experience, ability and other variables contributing to individual differences between students. Another important source of variation in learning outcomes relates to the kind of instruction which the students have received (e.g. Soloway, Ehrlich Bonar & Greenspan, 1982). For example, Dalbey, Tourniaire & Linn (1986) observed 30 junior high school BASIC programming classes. Most were found to have offered teaching which emphasised features of the programming language, and often failed to provide instruction in how to combine the language features into larger algorithms.

In this and other research it was found that students are introduced to a language feature such as the **PRINT** statement, and then write programs using that statement. Their understanding of the program is basically at the level of a single line. They type in a line and get feedback about their use of the **PRINT** statement. Students respond by typing in a different line which hopefully corrects the mistake they have made initially. These students are engaged in drill and practice on a language feature. Instruction rarely emphasises the templates which experts use for solving programming problems. Students therefore fail to acquire procedures which help them decompose problems and plan problem solutions.

Novice programmers are characterised by a **rush to the computer**. They frequently attempt to go from a statement of the problem directly to trial and error of program code without any consideration of how to design the code. Novices appear to lack the tools necessary for constructing intermediate states between the problem specification and the program program code. They rarely receive an opportunity to observe their teachers or expert programmers model the use of planning.

The expert process of **stepwise refinement** appears not to be required or really necessary for most assignments novices receive in programming instruction. It appears that many students fail to grasp the notion that programs are detailed process descriptions which can be refined out of natural language description.
They presume that programs are assembled by piecing the language features together. They fail to understand that the natural language problem description is less precise and more ambiguous than a problem description in machine terminology, i.e. code. They do not engage in the activities required for refining the natural language statement of the problem into a statement which can be decomposed and coded into a problem solution. As a result, when they are asked to solve problems which are more complex than simple translations of known language features, their solutions are often poorly organised, inefficient or even incorrect. The top-down designs and stepwise refinement which experts use to write programs are not taught or modelled by the teacher. As discussed in Chapters 6 and 7, a conservative estimate is that 25% to 35% of the students in the SUNRISE classrooms at Coombabah appear to be lacking the tools necessary for constructing intermediate states between the problem specification and the program code.

**Explicit Intervention to Foster Higher Cognitive Skills**

One method of intervention to counteract the above described lack of programming sophistication in novices is to provide students with some abstract templates which they could use for stepwise refinement of the problem specifications. This would equip students with a mechanism for constructing a problem solution that might be more detailed than the available problem specification but less detailed than the actual language statements. Students would thus be encouraged to consider an intermediate state between the problem specifications and the program code.

As noted above, most students move directly from the problem specification to the keyboard. This could lead to frustration and inefficient trial and error solutions. Coombabah students are being given tasks that can usually be solved or partially solved. In some areas they ask each other and keep trying. Lack of planning and inefficient solutions were certainly evident. The majority of students did not appear to become frustrated initially. However, as will be discussed further in later chapters, there was an indication of increasing frustration with more experience, with some Year 7 students becoming quite alienated.

It should be noted that students basically appear to be very happy when working at the terminal or on their laptops. They fail to associate their difficulties in achieving a solution with their lack of planning. A general belief among students is that more computing time is needed, rather than that they need to plan, hypothesise, evaluate, etc., to solve problems effectively.

Because of the nature of most of the instruction in computing, it is impossible to emphasise planning until the students have acquired at least a reasonable subset of the features of the computer language being taught. At Coombabah, the teachers have tried to emphasise planning but they have not explicitly taught it.

They did not provide abstract procedures which students could use to practise refining the problem specifications, nor did they provide students with exercises which required the translation of problem specifications into the provided procedures or templates. Students need practice in coding their solutions from templates as well as in mapping problem specifications onto templates.

The literature in educational computing shows that current modes of instruction frequently fail to communicate the value of planning in programming. This lack of appreciation of planning stems in part from certain characteristics of instruction. (1) Students' initial programming experiences do not require planning, therefore the advantages of planning are not apparent to them. (2) Students find the computing, i.e. interaction with the computer, very motivating and they prefer to be interacting with the computer, even if they are not making progress in solving the problems. Unfortunately, the interactive nature of the computer learning environment is not being well channelled towards the development of the higher cognitive skills such as planning.

One important reason why students fail to appreciate planning is that many of them can solve even the most difficult problems assigned without planning. Many students could recognise how to solve the problems which were presented to them without spending any time in planning.

Our experience at Coombabah suggests two directions which teachers might wish to consider in their endeavour to make instruction more cognitively demanding for their students:

Rather than beginning programming instruction with drill and practice in the language features it would seem quite appropriate to begin instruction with comprehension of program code. Students could be given reasonably sized programs (10-15 lines of code) and could be encouraged to come to understand those programs. Those programs would demonstrate how planning is used in programming. Students could see how experts use planning to write a big program. Thus students would have a better understanding of the role of planning in programming.

2 Structure diagrams could be used to help students comprehend a larger or more complex program. Such a program could be represented using structure diagrams. Comprehension of the program could be facilitated by using structure diagrams to illustrate the templates or procedures used by the programmer to construct the program. Instruction could then proceed by demonstrating the top-down design and the stepwise refinements as used by the expert programmer in the construction of a program.

Programming instruction has the potential of fostering the higher cognitive skills asked for by many recent reports on the state of educational practice but, so far, their potential is not being achieved. Instruction which builds sequences of computing knowledge and skills culminating in the planning skills used by expert programmers requires early and consistent emphasis on these skills. Teachers are
needed who can demonstrate and model good programming. Texts are needed which delineate the steps between problem specification and program code. Research is needed to understand more clearly the sequences of activities which will facilitate the desired learning outcomes.

**Metacognitive Outcomes**

The impact of working with computers on the problem solving skills of students may be increased if students had a greater awareness of the problem solving procedures or strategies embedded in their work. Brown, Campione & Day (1981) distinguish three types of training programs implemented by educators:

a. **Blind training** in which the learner is induced to use a strategy without concurrently understanding its significance.

b. **Informed training** in which the learner is persuaded to use a strategy at the same time provided with some information as to the significance of the activity.

c. **Self-controlled training** in which the learners are not only persuaded to use a strategy, but are also explicitly instructed how to employ, monitor, and evaluate the strategy.

Of these three training methods, the self-controlled method is by far the most successful in terms of enhanced performance and transfer. It has long been known that problem solving skills are likely to transfer to novel situations only if the principles on which they are based are made explicit to the learner (e.g., Lochhead, 1985; Simon, 1980). The self-controlled method may thus also be more effective in learning with computers.

One way to persuade students to monitor and evaluate their strategies while working with or without computers is to encourage them to reflect on their actions, for example by *thinking aloud* and by verbalising their ideas and strategies. Verbalisation helps learners to externalise ideas and strategies, to reflect on them and to elaborate them. Students could work in pairs or small groups on a task and take turns in verbalising while the others listen and ask questions. This forces both problem solvers and listeners to evaluate the strategies used and to monitor plans and solutions closely.

Computer-based learning environments are particularly conducive to enhancing cognitive and metacognitive skills as they provide the learners with many opportunities to practise these skills and receive immediate feedback.

**CONCLUSION**

It is possible to list some benefits which can be associated with learning with computers. In some circumstances, none are guaranteed to automatically flow from computer use, but many can be achieved through good teaching and the modeling of effective learning with and without the computer. There is an expectation that there will be: student access to learning resources and sources of information which do not depend upon the teacher; an increase in the variety of styles of teaching and learning; greater metacognitive awareness in students; more student planning and implementing of their own work programs; greater ability among students to use computers; and an improvement of skills such as self-monitoring, generalising, theory building and verifying information, concepts and ideas.

A number of commentators have offered the view (cf. Ridgway & Passey, 1991) that the experiences which students are offered in schools often conform to the school culture, and not to the intellectual tradition the educators purport to be teaching about. So students can learn how to perform a set of mathematical procedures, but not to function like a mathematician (e.g., Schoenfeld, 1985), or learn scientific facts, but not how to function like scientists (e.g., Edwards & Mercer, 1987). Computers in classrooms can actually offer the opportunity for students to learn how to be mathematicians and scientists. The students will also gain a deeper appreciation and understanding about the products of work in those disciplines.

The real implications of computing in education are yet to evolve. Harry Simon, in drawing an analogy between the introduction of the computer and the invention of the steam engine and the motor car (Simon, 1987), points out that both the steam engine and the motor car were responsible for further technological and scientific developments, but above all they opened up new social worlds, in which people had to function, and for which they were required to develop appropriate skills. Many of the social changes now associated with the motor car, even mediated by it were impossible to predict at the time of its invention: for example, the suburban sprawl, environmental damage and huge employment opportunities.

Major educational and social changes can be expected from the introduction of computers, and these changes will be even larger as a result of the use of computers in learning and teaching. We have to live with the fact that we do not know what the real and best uses of computers in education will be, and adopt a style of research and development which allows us to capitalise upon its as yet undetermined potential in an opportunistic fashion.

Because the cognitive technologies which we invent serve as instruments of cultural redefinition (shaping who we are by changing, and not just expanding and supplementing what we do), defining educational values becomes a foreground issue. The demands of an information society make an explicit emphasis on general cognitive skills a priority. The urgency of updating...
education's goals and methods can only be met by an activist research paradigm which simultaneously creates and studies changes in processes and outcomes of human learning with new cognitive and technological tools.