Introducing Programming: A Balanced Approach

Christine Prasad
Kay Fielden

UNITEC
Auckland, New Zealand

cprasad@unitec.ac.nz

Proceedings of the 15th Annual NACCQ, Hamilton New Zealand July, 2002 www.naccq.ac.nz

ABSTRACT

Teaching computer programming at an introductory level is a challenging task with a number of associated problems. The current approach taken towards research into teaching computer programming at an introductory level can easily be classified into four main categories: Delivery, Theory, Cognitive Style and Social/ Behavioural. Previous research done in each of the four categories mentioned above find no clear solution to the problems. This paper reports on a comprehensive investigation into past and current research in the area of introductory computer programming, and suggests that a balanced approach be taken towards teaching programming at an introductory level. This approach would incorporate a multiplicity of cognitive styles, regardless of delivery modes and theoretical content, and hence, form the basis for future research.

Keywords: Computer education, delivery mode, theory, cognitive style, introductory programming.

1. INTRODUCTION

In the current day and age, when computers are part of domestic and office "furniture", when household appliances are programmable, when databases are the norm rather than the exception, when computerised searches and simple office applications require the use of logical operations, one would expect that computer programming is not a subject which people would shy away from. However, many people are put off by the supposedly abstract and "hi-tech" nature of the discipline.

This research is seeded by the belief that programming should be for everyone and that learning how to program should somewhat be like learning how to drive a car. This belief is supported by many writers like Appleman, (1994), Fellesien *et al.* (2000) and Knuth, (1973). As a starting point, programming at an introductory level at a tertiary education level is being investigated.

Despite having being around for at least the last 40 years, teaching computer programming at an introductory level is still a challenging task. This is seen in high failure rates in first year programming courses, low student retention numbers in programming streams and, for many students who successfully proceed to higher level programming courses, a lack of understanding of basic programming concepts (Carbone *et al.* 2001, Jenkins,

2001). Some of the factors attributing to this are the abstract nature of programming, the method of delivery of programming courses, experience and skill of teachers, and the belief that programming is suitable for people with certain learning styles and attributes (Byrne and Lyons, 2001).

While it is true that computer programming teaching has evolved drastically from the early days with theory now being heavily supported with practice, and innovative teaching methods and delivery modes being put into the curriculum, the concept that programming is for a 'certain type' of people is still paramount. There is still a common belief that programming is a different way of thinking (Turkle and Papert, 1990), and that only people who are analytical, logical thinkers and planners have a real chance of learning how to program successfully.

When dissected, a number of disciplines are incorporated in computer programming. Firstly, since its roots are in the mathematical sciences, programming requires the same skills as mathematics, i.e. problem solving, abstraction, and deduction. (Fellesien *et al.* 2000). Secondly, programming requires skills needed to learn a language i.e. comprehension, reading, writing. Thirdly, programming requires creativity and composition skills as in arts. Fourthly, since most programs in the real world are written to support business, an understanding of the business sector is also necessary. Also, programming relates to the field of psychology and communication given the 'user' end of the discipline.

In this paper, the word 'programming' refers to the process of understanding and solving a problem, designing its solution, coding the solution in a programming language, and testing it to completion.

This paper reports on the four main categories in which introductory programming is currently being researched. These categories were classified on the basis of the quantity of literature present and form the four following main sections of this paper:

1.1 DELIVERY

This category addresses issues relating to the delivery of introductory programming classes, i.e. the modes, tools and techniques used to pass the subject content to the learner, or, 'how' information is delivered.

The section on this category discusses the two fundamental theoretical models of teaching: objectivism and constructivism. It also reports on how teaching styles have evolved over the years, from the traditional lecture format to more tutorial and practical classes, and delivery modes from traditional paper and textbooks to the current approaches being internet, online and web delivery. Also discussed are tools using graphics and multimedia.

1.2 THEORY

This category addresses the theory covered in introductory programming classes, that is, the concepts conveyed to students, or, 'what' information is delivered.

This section further describes how the theory taught has evolved over the years, and the reasons for this evolution. It also covers the evolution of teaching language and methodologies.

1.3 COGNITIVE STYLE

This category addresses cognitive style, i.e. the learning styles of students believed to be prevalent when learning programming.

The section on this category looks at a number of bipolar cognitive styles that have been examined in available literature and one popular instrument for measuring cognitive style.

1.4 SOCIAL/ BEHAVIOURAL

This category addresses all the 'non-academic' issues that may affect introductory programming students like age, gender, motivation, attitude, previous computing experience, and mathematical background.

The final section of this paper links the above categories, then describes the direction for further research.

2. DELIVERY

Significant research has been carried out in the area of delivering introductory programming classes, that is, the modes and methods by which information is carried across to learners.

The two theoretical models on which classroom teaching is based is objectivism and constructivism where the objectivist model is teacher-centred and

teacher-controlled while the constructivist model is student-centred (Van Gorp and Grissom, 2001).

The traditional teaching style in most tertiary institutions has been wholly objectivist, with the teacher appearing in front of the class to deliver lectures either verbally, and/or by the aid of presentation tools. However, as Carter and Boyle (2002) quite correctly point out, in a practical subject like IT, students need to be active in the learning process by "interacting with the concepts, testing out theories, being allowed the space to make mistakes". Around the late 1980's, the constructivist model was enforced and most learning institutes coupled teacher-centred lectures with separate, practical or laboratory classes where students were required to construct knowledge based upon meaningful activities and collaboration with their peers.

Continuing research into Computer Education has come up with other means to enforce the constructivist model:

- Discussion classes: Hagan and Sheard (1998) reported on using discussion classes aside from the lectures and practical sessions which gave students the opportunity to construct ideas and collaborate amongst peers and teachers away from computers.
- Group based classes: Van Gorp and Grissom (2001) used a number of group based collaborative activities like code walkthrough's, code writing, debugging and lecture note reconstruction
- Visualisation Tools: Smith and Webb (2000) reported on using a low-level program visualisation tool called Bradman which gave the students an opportunity to see how a program executes, and conclude that such tools "can be beneficial in teaching novice programmers". Satratezemi *et al.* (2001) reported on using AnimPascal, a program animator that enables students to visualise program execution. Rowe and Thorburn (2000) reported on using an on-line tutorial tool for teaching introductory programming called VINCE.
- Problem-Based Learning (PBL): Barg *et al.* (2000) reported on an even more interesting approach where they used Problem-Based Learning for their foundation level Computer Science course. This uses a large, real-world problem where students work in groups and traditional lectures are replaced by additional tutorial and laboratory time. Stein (1998) described an alternative approach of a first year programming course in Java where the student learned "computation by interaction".
- Graphics: McKay (1999) reported on enhancing text-based instructional materials with Graphics and compared the materials with the learning style of students and saw an improvement in performance when students were presented with the graphically enhanced materials.
- WWW/Internet/Distance Learning: Hagan, Sheard and MacDonald (1997), Hagan and Lowder (1996) reported on using the World Wide Web (WWW) to interest students, primarily by putting information on the Web and using discussion boards. Karsten and Kaparthi (1998) reported on using visual explanations of programming constructs on the WWW to supplement conventional instructional methods and materials for introductory programming courses. They used graphics and animations to enhance visualisation.
- Virtual Labs: The PEARL project (http://kmi.open.ac.uk/projects/pearl/) currently being carried out at the University of Cambridge, is researching how laboratories can be carried out over the Internet. This will be a breakthrough in setting an interactive, web-based delivery mode for teaching practical subjects like Computer Programming.

All the projects mentioned above, except the PEARL project, which has not published any quantitative analysis, observed improvements in the performance of their students. Some found marginal improvement while others, like the PBL and discussion classes found major improvements.

Despite such a variety of delivery modes available, many institutions still continue with the typical lecture/tutorial/practical approach. The main reason behind this would be that first year classes are usually quite large, so implementing new delivery modes would be at a considerable cost and hence are quite often not implemented.

3. THEORY

Only as far back as 12 years ago, an introductory programming class would cover huge chunks of computational theory and mathematics. However, with the constructivist approach being taken, a lot of

this changed. Also, it was realised that, although the basis of computer programming is Mathematics, students do not need to know in detail, the mathematical background in order to be programmers (Knuth, 1973)

Currently, the content covered in programming lectures usually start off with an introduction to the Program Development Lifecycle, then moves on to problem solving, algorithms and their representation, and then the translation of the algorithm to a programming language, with the language syntax and concepts being taught in parallel. Taking a more pragmatic and real-world approach, the programming lectures are delivered, with shorter theoretical lectures supplemented with laboratory and practical sessions.

The bulk of literature in this category covers the choice of teaching language and methodology. In the early days of computing, machine language was the main language for teaching, this was when hardware and processors were in the limelight. Around the late 70's and 80's, COBOL became the prime choice of teaching language due to its business orientation and commercial popularity. This was succeeded by more general purpose, 3rd generation procedural languages like Pascal, Basic and C, which made it easier for students to understand the logical intricacies of a program (Augustine and Surynt, 1990).

Now, in support of the change of programming methodology from structured/ modular/ procedural/ process-driven to object-oriented/ data-driven, the trend is moving towards more object-oriented languages like C++ and Java. A number of researchers have reported on the change of first year Computer Science curriculum from structured concepts and languages to object-oriented: Kölling and Rosenberg (2001) have reported on introducing Object Oriented concepts using a development environment called BlueJ; Sheard and Hagan (1997), Conner, Niguidula and Van Dam (1994), Connolly (1996), Stein (1998) have also reported on similar case studies.

Bruce *et al.* (2001) reported on the using event-driven programming techniques in a first year programming course that uses Java, with the help of a custom-made library. Fagin (2000) has suggested the use of Ada-based Robotics to teach introductory computer programming courses, with the incentive that students will see the visual results of the amount of work they put into writing a program.

Augustine and Surynt (1990) have suggested the use of a fourth generation approach to an introductory programming course using a fourth generation language (4GL) and 4th Generation Programming Tool (dBaseIII Plus), since "they incorporate interactive, integrated, multi-functional programming environments".

However, it should be noted that the choice of a teaching language is usually dependent on current market trends and suitability to teaching, especially given the large numbers and wide array of computing backgrounds of first year programming students. Also, the argument still remains that programming languages are merely tools to illustrate the concepts of programming, and that a good programmer should be able to adapt to any programming language despite the language s/he was taught in.

4. COGNITIVE STYLES

This category addresses the less technical and more psychological issues of teaching programming. Cognitive style has a number of definitions (Riding and Cheema, 1991). In this paper, we use the term cognitive style as a broad umbrella term for all psychological elements relating to learning, and this ranges from aptitude to personality types.

Research in this category suggests that the biggest indicator of cognitive style is personality type (Bishop-Clark and Wheeler, 1994). The most suitable instrument for studying cognitive styles in computer related disciplines is suggested to be the Myers-Briggs Type Indicator (MBTI) (Jones, 1994).

As described by Bishop-Clark and Wheeler (1994), the MBTI stems from Carl Jung's theory of personality types and measures preference on four bipolar dimensions. The two dimensions of the MBTI considered to be the cognitive, i.e. learning dimensions, are the perceiving (sensing/intuitive) and the judging (thinking/feeling) dimensions. Most researchers hypothesise that sensors (those who depend on their senses rather than intuition), and thinkers (those who analyse rather than feel) are more likely to perform better in computer use and programming.

Out of all the available literature in this area, perhaps the most relevant is that of Bishop-Clark and Wheeler (1994), who did a study of students enrolled in an introductory programming class based on the hypotheses above. The findings indicated that

personality may have little to do with overall achievement in computer programming classes.

However, after further research, Bishop-Clark (1995), also stated that "most of the studies relating computer programming achievement to cognitive styles or personality traits treat computer programming as if it were a single activity.." and "it may be that a particular cognitive style influences problem representation and planning but has little effect on the coding and debugging stages of computer programming." This is an important investigation due to the multi-disciplinary nature of programming.

Van Merrienboar (1990) based his research on the reflection-impulsivity cognitive style in relation to computer programming where reflectives tend to plan their activities which impulsives are spontaneous. He hypothesised that reflectives would prefer to learn programming by completing existing programs while impulsives would prefer to learn by generating programs. The results of his findings did not support his hypothesis fully.

Yet another measure of cognitive dimensions is the Wholist-Analytic and Verbal-Imagery (Riding and Cheema, 1991) where the wholist learner is able to perceive whole concepts while the analytic learner analyses material into parts without seeing the whole concept, and the verbal learner prefers verbal tasks and represents information verbally or in mental pictures during thinking while imagers prefer concrete, descriptive and imaginable tasks.

McKay (1999) carried out a study to compare performance of the 4 different cognitive style learners in programming tasks with text-based instructional materials versus graphics enhanced materials. Contrary to her hypothesis, verbalisers performed better with the graphics enhanced materials than imagers.

Another approach to "typing" learners is seen in Turkle and Papert (1990), who discuss pluralism in the approach to computer programming knowledge. They describe two "type" of people – bricoleurs, who gain knowledge from concrete evidence, are creative, artistic and spontaneous, and planners, who prefer abstract, planned, structured and traditional approaches to learning programming. The way the two groups approach the same problem is almost opposite, but the end results they get are the same.

While there is much lesser literature in the area of cognitive styles and computer programming when compared to the previous two categories, there appears to be a lot more promise in further studying this category in regards to finding the solutions to the problems of teaching introductory programming.

5. SOCIAL AND BEHAVIOURAL

The remaining literature relate to "external" factors affecting learning of programming i.e. social and behavioural. Some of these are gender, motivation, attitude, age, mathematics, programming, and computing background and experience.

A lot of observations have been made on the lower number of female students in computer programming courses but researchers have not yet found an answer to why this trend exists (Byrne and Lyons, 2001; Wilson, 2002). Also, as expected, lower motivation levels and negative attitudes will have a negative impact towards learning any subject so Jenkins (2001) suggests that instructors ought to motivate students better.

Prior computing background works both as an advantage and as a disadvantage in different cases of introductory programming. McKay(1999) determined that students with little or no programming background performed better using structured instructional formats while those with prior programming knowledge were happy with the discovery instructional format, i.e. those which enable the learner to construct and draw knowledge as they prefer to. Further research has concluded a "clear link between programming ability to existing aptitude in mathematics and science subject" (Byrne and Lyons, 2001) and that "experience with a programming language would, at least initially, benefit students in introductory programming courses" (Hagan and Markham, 2000).

These areas of research in Introductory Programming are still new and, in most cases, it is difficult to draw conclusive results from data.

6. WHERE TO FROM HERE?

The primary purpose of this investigation was to understand the factors affecting teaching computer programming at an introductory level, and to find out if any solutions have been found for the problems associated with teaching.

Having examined a substantial amount of literature on previous and current research, it is evident that the problems have not been totally solved. However, a number of innovative delivery methods have been created and attempted, and these have increased student performances to some extent. The introduction of new languages and methodologies, have not resulted in a major improvement in performance.

It is also seen that, due to its multidisciplinary nature, any one particular cognitive style or social and behavioural factor cannot be solely responsible for the effective learning of programming.

Given that every student is different and has a different learning style, which may fall either at the extremes or anywhere in between the wide spectrum of bipolar cognitive scales, it seems apparent that just changing the delivery mode or programming language will have minimum effect on overall learning, as each has a different effect on different "types" of people.

What we need then is a balance of delivery modes and theory to cater for a range of learning types. In other words, if current programming courses tend to favour the "planner" types of people, it must be restructured to cater for the "non-planner" types as well.

The direction, then, for our future research, is to get this balance into the introductory programming curriculum that will cater for all "types" of individuals.

7. CONCLUSION

Despite the number and variety of research in this area, the problems associated with the teaching and learning of computer programming at introductory levels still exist. In this paper, we have presented the findings of previous research on introductory programming in the categories of Delivery, Theory, Cognitive Style, and Social/ Behavioural and conclude that no one category can be held responsible for solving the problems. Thus, we have suggested a balanced approach to teaching introductory programming which will span over multiple cognitive styles. It is hoped that this approach will enable people of all "types" to adjust to learning programming and will erase the conceptions of programming being a different kind of thinking for a certain "type" of people.

REFERENCES

Appleman, D. & Ishida, S.(1994). How computer programming works. Ziff-Davis Press, Emeryville, CA, USA.

Augustine, Dr. F. K. Jr. & Surynt, Dr. T. J. (1990). A Fourth Generation Approach to the Introductory Programming Course. Journal of Information Systems Education. Vol 2, No 2, 8p.

Barg, M., Fekete, A., Greening T., Hollands, O., Kay, J., Kingston, H. & Crawford, K.(2000). Problem-Based Learning for Foundation Computer Science Courses. Computer Science Education. Vol 10, No 2, pp109–128.

Bishop-Clark, C.(1995). Cognitive style and its effect on the stages of programming. Journal of Research on Computing in Education, Summer95, Vol. 27 Issue 4, pp373-378.

Bishop-Clark, C. & Wheeler, D. D.(1994). The Myers-Briggs Personality Type and its Relationship to Computer Programming. Journal of Research on Computing in Education. Vol 26, No 3, pp358–369.

Bruce, K. B., Danyluk, A. P. & Murtagh, T. P.(2001). Event-driven Programming is Simple Enough for CS1. Proceedings of the 6th Annual SIGCSE/ SIGCUE Conference on Innovation and Technology in Computer Science Education ITiCSE 2001, pp1–4.

Byrne, P & Lyons, G.(2001). The Effect of Student Attributes on Success in Programming. Proceedings of the 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education ITiCSE 2001, pp49–52.

Carbone, A., Hurst, J., Mitchell, I. & Gunstone, D.(2001). **Characteristices of Programming Exercises t**hat lead to Poor Learning Tendencies: Part II. Proceedings of the 6th Annual SIGCSE/ SIGCUE Conference on Innovation and Technology in Computer Science Education ITiCSE 2001, pp93–96.

Carland, J. A. & Carland, J. W.(1990). Cognitive Styles and the Education of Computer Information Systems Students. Journal of Research on Computing in Education. Vol 23, Issue 1, pp114–127.

Carter, J. & Boyle, R.(2002). Teaching Delivery Issues – Lessons from Computer Science. Journal of Information Technology Education. Vol 1, No 2, pp77– 89.

Conner, D. B., Niguidula, D., and Van Dam, A.(1994). Object-Oriented Programming: Getting it Right at the Start. Educator's Symposium at the 9th Annual conference of Object Oriented Programming Languages, Systems and Applications, Portland, Oregon.

Connolly, M. V.(1996). Starting Computer Science Using C++ with Objects: A Workable Approach. Association of Small Computer Users in Education (ASCUE) Summer Conference Proceedings (29th, North Myrtle Beach, SC).

Fagin, B. S. (2000). Using Ada-Based Robotics to Teach Computer Science. Proceedings of the 5th Annual Conference on Innovation and Technology in Computer Science Education, July 2000, Helsinki Finland.

Felleisen, M., Findler, R. B., Flatt, M., & Krishnamurthi, S.(2001). The MIT Press, Cambridge, Massachusetts, USA.

Hagan, D. & Lowder, J.(1996). Use of world wide web in introductory computer programming. 13th Annual conference of the Australian Society For Computers In Learning In Tertiary Education, University of South Australia, Adelaide, pp247-257.

Hagan, D. & Markham, S.(2000). Does it Help to Have Some Programming Experience Before Beginning a Computing Degree Program? Proceedings of 5th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITICSE 2000, pp25-28.

Hagan, D. & Sheard, J.(1998). The Value of Discussion Class for Teaching Introductory Programming. Proceeding of 3rd Annual Conference on Integrating Technology into Computer Science Education, ITICSE '98, Dublin, Ireland, pp108–111.

Hagan, D., Sheard, J. & MacDonald, I.(1997). Designing and developing a new learning environment for teaching introductory programming: change for effect. World Multiconference on Systemics, Cybernetics and Informatics, Internet Institute of Informatics and Systemics, Caracas Venezuela, pp206-209.

Jenkins, T.(2001). The Motivation of Students of Programming. Proceedings of the 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education ITiCSE 2001, pp53–56.

Jones, P. W.(1994). Computer Use and Cognitive Style. Journal of Research on Computing in Education. Summer94. Vol 26, Issue 4, pp514 – 523.

Karsten, R. & Kaparthi, S.(1998). Using Dynamic Explanations To Enhance Novice Programmer Instruction via the WWW. Computers & Education. Vol 30, No 3-4, pp195-201.

Kölling, M. & Rosenberg, J.(2001). Guidelines for Teaching Object Orientation with Java. Proceedings of the 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education ITiCSE 2001, pp33-40.

Knuth, D.E.(1973). The art of computer programming: Fundamental algorithms. Second Edition. Addison-Wesley, USA.

McKay, E.(1999). An investigation of text-based instructional materials enhanced with graphics. Educational Psychology. Vol 19, no 3, pp323-335.

PEARL. Effective Learning through Remote Experiments. Accessed March 12th, 2002. <http://kmi.open.ac.uk/projects/pearl/>

Riding, R. & Cheema, I.(1991). Cognitive Styles – An Overview and Integration. Educational Psychology, Vol 11, Issue 3/ 4, p193.

Rowe, G. and Thorburn, G.(2000). VINCE – an on-line tutorial tool for teaching introductory programming. British Journal of Education Technology. Vol 31, Issue 4, p359.

Satratzemi, M, Dagdilelis, V & Evagelidis, G.(2001). A system for program visualisation and problem-solving path assessment of novice programmers. Proceedings of the 6th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education ITiCSE 2001, pp137-140.

Sheard, J. & Hagan, D.(1997). Experiences with teaching object-oriented concepts to introductory programming students using C++ Tools. Asia'97 Conference in Beijing, China.

Smith, P. A. & Webb, G. I. (2000). The Efficacy of a Low-Level Program Visualisation Tool for Teaching Programming Concepts to Novice C Programmers. Journal of Educational Computing Research. Vol 22(2), pp187–215.

Stein, L. A.(1998). What We Swept Under the Rug: Radically Rethinking CS1. Computer Science Education. Vol 8, Issue 2, p118, 12p.

Turkle,S. & Papert,S.(1990). Epistemological Pluralism: Styles and Voices within the Computer Culture. E & L Memo, no 3 MIT Media Laboratory, http://lcs.www.medi.mit.edu/groyups/el/Papers/memos/memo3/3.EpistPlural.html

Van Gorp, M. J. & Grissom, S.(2001). An Empirical Evaluation of Using Constructive Classroom Activities to Teach Introductory Programming. Computer Science Education. Vol 11, No 3, pp247-260.

Van Merrienboer, J. J. G.(1990). Instructional strategies for teaching computer programming: Interactions with the cognitive style reflection-impulsivity. Journal of Research on Computing in Eduation. Fall90, Vol 23, Issue 1, pp45-54.

Wilson, B. C.(2002). A Study of Factors Promoting Success in Computer Science including Gender Differences. Computer Science Education. Vol 12 (Numbers 1-2), pp141–164.