Ways of experiencing the act of learning to program:

A phenomenographic study of introductory programming students at university

Christine Bruce
Camille McMahon
Lawrence Buckingham
John Hynd
Mike Roggenkamp

Feb 2003

Abstract

The research reported here investigates variation in first year university students' early experiences of learning to program, with a particular focus on revealing differences in how they go about learning to program. A phenomenographic research approach was used to reveal variation in relation to the act of learning to program. Semi-structured interviews were conducted with students who had either completed, or were recently completing a university level introductory programming subject. The analysis process revealed five different ways in which students go about learning to program in introductory university level units. These are captured in categories of description which capture the critical dimensions of what students learn as well as how they go about learning. Students may go about learning to program by:

- Following – where learning to program is experienced as 'getting through' the unit.
- Coding – where learning to program is experienced as learning to code
- Understanding and integrating – where learning to program is experienced as learning to write a program through understanding and integrating concepts
- Problem solving – where learning to program is experienced as learning to do what it takes to solve a problem
- Participating or enculturation – where learning to program is experienced as discovering what it means to become a programmer

The mapping of the variation constitutes a framework within which one aspect of the teaching and learning of introductory programming, how students go about it, may be understood. Implications for teaching and learning in introductory university curricula associated with each category are discussed. Recommendations for further research are made.

Table of Contents

1 Introduction

An expected outcome of many students' computer science and engineering education is programming skill (McCracken et al., 2001). However, much of the literature on computer programming education shows high failure rates as a major problem; it is common for students to approach their final year project work determined to avoid programming at all costs, presumably because they either cannot program or believe that they can not (Carter and Jenkins, 1999, p.1). The literature also provides many examples of the implementation of new teaching approaches in programming education, usually in an attempt to improve poor rates of student success (e.g. Fincher, 1999; Barg et al., 2000; Stein, 1999; Gottfried, 1997; Williams and Kessler, 2000; Hagan et al., 1997; Ellis et al., 1999; Carbone and Sheard, 2002). Our examination of the literature reveals (Bruce and McMahon, 2002) a 'trial and error' approach to redesigning introductory programming curricula largely based on the application of constructivist principles of active and collaborative learning. The outcomes of this work have produced limited insights into how students learn to program and how to help them learn to program.

Since the mid 1970s the constitutionalist, or relational, research agenda has provided new and important understandings of the character of teaching and learning in higher education (Marton and Booth, 1997; Bowden and Marton, 1998). A wide range of studies in Australia, Sweden, UK, Canada, China and elsewhere have revealed repeatedly that it is possible to make visible variation in the outcome of learning, and in the act of learning amongst groups of students; and that achieving this makes it possible to design learning experiences which bring about the kinds of learning outcomes that are desirable. (Marton and Tsui, in press; Watkins and Biggs, 2001) The outcomes of this body of research, and its associated theory of teaching and learning, has had a far reaching impact on the higher education sector and development programs for university teachers internationally (Ramsden, 2003; Trigwell and Prosser, 1997). This agenda has already been applied to aspects of IT education, resulting in the early formation of relational frameworks for information literacy (Bruce, 1997), information systems (Cope, 2000) and programming (Booth 1993) education. Studies are currently being undertaken investigating students' understanding of networking protocols (Berglund, 2002), and algorithmic thinking (McDonald, 2002).

Despite our limited understanding of students' experiences of learning to program, the idea that improvements to teaching can be made by investigating and understanding variation in learning to program has been clearly established. In her research with engineering students, Booth (1992; 1993) investigated variation in the outcomes of learning to program in which she established students' ways of experiencing or constituting programs and programming. Subsequent research by Booth (e.g. 1997; 2001) and Carter (e.g. Carter, 2001; Carter and Jenkins, 1999) has further illuminated the variation in experiences that potentially affect students' learning of programming. Carter and Jenkins (1999) investigated attitudes and approaches to the learning of programming and found differences in the way students approached programming based on differences in the way they are conditioned to study according to gender. Booth (2001) is presently engaged in research investigating the idea of learning to program as an induction to a datalogical culture, part of the professionalisation of more experienced students. Further research investigations of students' ways of experiencing learning to program, particularly the experience of novices, are required.

This project is part of an emerging international agenda seeking to investigate the practice of teaching and learning programming based on constitutionalist learning theory. Our investigation complements the earlier work of Booth (1992). It seeks to illuminate variation in IT students' early experience of learning to program at university, focussing particularly on the act of learning in introductory programming units.

2 Aims and significance

2.1 *Aim*

Our research aims to investigate variation in students' early experiences of learning to program. The specific focus of the project reported here is in relation to the act of learning. We have addressed the question: What are the different ways in which foundation year students go about learning to program?

2.2 Significance of the study

The significance of this study resides in its ability to illuminate one aspect of how the experience of learning to program is constituted in the foundation year learning community. Understanding the act of learning to program from the learners' perspective complements the work of Booth (1992) which uncovered variation in aspects of the outcomes of learning to program from the learners' perspective. The study also opens the territory of understanding learning to program from the perspective of university students undertaking introductory courses.

The study affirms the value of the phenomenographic approach for investigating students' experience of learning to program and offers results which have clear implications for teaching. The outcomes suggest therefore that the study has implications for curriculum design and teaching.

3 Method

3.1 Theoretical Framework and research approach

Our project is grounded in the view of learning espoused by Bowden and Marton (1998). In this view, learning is about broadening one's ways of experiencing some aspect of the world, and teaching is about giving learners access to the different experiences and bringing into focus the critical dimensions of these experiences. The collective consciousness of the learning community refers to the complementarities in teachers' and learners' ways of seeing the experience of learning to program. These complementarities (or conceptions) may be described in terms of the differing ways in which learning to program is constituted, the differing ways in which teachers and learners are aware of learning to program and the differing ways in which learning to program appears to them. These differences may be further described in terms of the different dimensions of learning to program which become figural in awareness in a particular way of seeing, and those dimensions which recede to the ground. In this investigation it is the ways of constituting learning to program amongst early learners, which is our research object. This research object places our investigation within the terrain of the phenomenographic research project.

Phenomenography is an interpretive research approach that seeks to describe phenomena in the world as others see them, the object of the research being variation in ways of experiencing the phenomenon of interest (Marton and Booth, 1997, p. 111), in this case, learning to program. A fundamental assumption underlying phenomenographic research is that there are a finite number of qualitatively different understandings of a particular phenomenon. The research object of phenomenography is the intention to uncover variation in the experience, or collective consciousness of aspects of the world. An established result in phenomenography is that there is a variation in ways of experiencing the conditions for learning, and that the variation in experience is critical for the outcome of learning (Marton and Booth, 1997). Its proven contribution to educational research (Bowden and Marton, 1998) makes it an ideal method for this study. In the context of teaching and learning programming, a phenomenographic approach will reveal the variation in the ways introductory programming students experience the act of learning to program.

3.2 Data gathering and preparation for analysis

Interview questions were piloted with 2 students. The questions were then revised and in-depth semi-structured interviews were conducted with another 13 participants. Table 1 summarises the participant profile of the study. One interview included two participants (participants 3 and 4). Participants were sought from the two main first year programming units at QUT. Students use Java as their programming language in these units. Students were asked to sign an informed consent agreement at the beginning of each interview. While there was no selection process designed to maximise variation, participants did represent a range of the student population. In light of low numbers of volunteers, all participants who offered to be interviewed who were either doing a first year subject, or who had recently completed a first year subject (e.g. participant 6), were included in the study. Data collection was done in two stages. The participants interviewed in the second stage (participants 10 – 13) were taking a first year unit itb410 during the summer semester and were studying under slightly different conditions than those during the year; they were in a smaller group, they had their tutorial sessions in a PC lab.

Table 1 **Participant profile**

sex		**age**				**subject**			
M	*F*	*<20*	*21<30*	*31<40*	*>40*	*ITB410**	*ITB411**	*ITB410 summer*	*Other#*
9	4	4	4	4	1	1	7	4	1

* ITB410 is first year first semester, ITB411 is first year second semester.
# a second year student, having recently completed first year, was included in the research

Prior programming experience			
nil	*Basic*	*intermediate*	*Advanced*
3	5	2	3
Type of prior experience (of 10 students)			
School	*TAFE*	*University*	*Self-taught*
4	1	2**	3
Timeframe over which experience was gained			
1-4 years ago		*10 – 20 years ago*	
6		4	

** one of which was through distance education

It is a specific design feature of the questions in phenomenographic interviews that they should (1) direct the interviewees towards the phenomenon, and (2) be broad enough to obtain meaningful responses in relation to the aim without forcing a particular structure, or way of responding upon the participant. Each interview consisted of the following 'trigger' questions concerned with the students' experiences of learning to program and the different ways they see programs and programming. Each question serves as an 'opening', from which the interviewer develops a trail of further questions in order to achieve a mutual understanding of the theme in focus.

- How do you see your current ability to program? What makes you say that? How do you decide?
- Do you enjoy programming?
- How do you go about learning to program?
- Can you describe for me how you went about learning to program when you first started? (for those who have learnt to program in the past)
- Can you write a program that works? How do you know?
- Can you write a good program? How do you know?

At the completion of each interview, the voice recording was transcribed verbatim and checked by the interviewer.

3.3 The analysis process

Data analysis aims to develop a theoretical representation of ways of experiencing the act of learning to program. The process is grounded in seeking variations in meaning which are associated with structural variation. In order to achieve this level of analysis a rigorous, iterative approach was required involving the whole research team.

Initial analysis involved the process of becoming familiar with the transcripts as a whole. Each transcript was read and re-read numerous times in an attempt to reveal broad differences in pools of meaning in the data. Chunks of text, or potentially relevant quotes were then separated from the transcript and analysed for their meaning. This served as a means of decontextualising the quotations from the individual respondent and was used to further identify common pools of meaning or categories and reveal structural differences between the categories. Each member of the research team involved in the analysis, discussed their own findings and met in order to reveal further categories, and refine the categories as had been revealed thus far.

At this point, the transcripts were once again referred to in their entirety. This process helped determine the context of some of the quotes and also revealed further relevant quotations that had not been extracted in the first round. The process also helped with the development of a narrative description of what it means to program for each category. In comparing the text of individuals, it also helped discern the variation in meaning and structure between the categories. While the description on the individual level was used to help clarify/refine the categories of description, it was not used to classify individuals as belonging to any particular category. In other words, we are not referring to individual variation in our description of the outcomes.

The analysis process was also guided by emerging understandings of how the act and outcomes of learning may be described (Marton and Booth, 1997). The particular descriptive focus inherent in phenomenography has produced two distinctive outcomes of any phenomenographic study: categories of description, and an outcome space. The primary outcomes of our investigation, therefore, are:

- Categories of description associated with the act of learning to program. These capture the critical dimensions of how students go about learning to program, and
- An outcome space which describes the relationships between the categories.

Each category of description represents one 'conception', or way of *experiencing* or *being aware of* or *constituting* the act of learning to program; where the act of learning to program is the *object* of the learning experience.

In each category referential components, that is the critical differences in meaning (the *direct object*), as well as structural components (the *indirect object*) (Marton and Tsui, in press) are highlighted. In the structural component of each category the awareness structure is usually delimited in terms of internal and external horizons. The *External Horizon* represents that which recedes to the ground, essentially the perceptual boundary associated with the participants' ways of seeing. The *Internal Horizon* represents the focus of the participants' attention, or that which is figural in awareness and simultaneously attended to.

From transcriptions of the interviews, the research team therefore sought (1) the variation in meaning associated with the act of learning to program [the referential component of the categories of description], and (2) an understanding of the awareness structures through which learners constitute the act of learning to program [the structural component of the categories].

For each category presented here, dimensions of variation, those aspects of learning to program which systematically differ in each category, are also established. The dimensions of variation associated with each different way of experiencing the act of learning to program are 1) learning activities or approaches, 2) how learning a programming language is seen, 3) key motivations in learning to program and 4) ways of seeing programs and programming.

Finally the outcome space is a diagrammatic representation of the logical relationships between conceptions, based on their structure as described in the categories of description. In other words, the outcome space captures the range of experience at the collective level (Booth, 2001a). The focus on uncovering the structural framework is fundamental to phenomenographic research, and is the factor which yields unique insights into the experiences of the act of learning to program.

4 Ways of experiencing the act of learning to program

The categories of description revealed in the data are described below. Direct quotations from participant interviews are provided to illustrate the described features of the categories. The participant number and page number of the interview are noted after each quotation. Discussion of the implications of each category for teaching and learning is included in section 6.

4.1 Category 1: 'Following'

4.1.1 Referential Aspect

Learning to program is experienced as getting through the unit. When experiencing learning to program this way, the student may struggle to keep up with the set assignments. Where marks are to be gained, the student will focus on those tasks. Time might be seen as the major factor in determining whether or not the student gets through the course. Feedback is sought from teaching staff particularly to indicate whether the student is on the right track. Students experiencing or going about learning to program this way are affected by the structure of the course and the way the material is presented. For example, if the material is presented in a way that doesn't match their expectations or perceived needs, they experience frustration.

> "..with the weekly pieces of assessment, they sort of are confirmation, you know. Keep moving along, don't drop the ball… And, I have to say in one subject I have dropped the ball, so I'm not doing well" (7:4)

> "…because it's 1%, I want the 1% so I work on it as best I can, so I want it to hand in and the guy will go 'yep that's good work' tick, 1%" (2:8)

> "I think there has to be some assurance to a person that if they follow the course, you know if they follow the tasks, that it gets somewhere…"(7:11)

> "There's no time, it's like bang, right move onto the next one. Bang you know. You've got to get through this list…umm this thing you've been given and you should have all the answers ready by now you know. But you've spent the whole week trying to pump out the assignment, how can you get the tutorial answers ready?" (1:11)

> ".. the students that are going to do well in this are the ones that don't have to work, who've got, you know, mummy and daddy have got them a laptop, they can, they can come to…you know and every spare moment you've got you can be working on your software development…" (1:10)

> "it's a pressure cooker sort of…. ummm.. you know… yeah, just the frustration I've experienced, that perhaps it could flow a lot more smoothly if the procedures were better.... Especially for the student learning. I think they're trying to reach a certain goal, with all the students being up to a certain level that's really not achievable" (1:12)

4.1.2 Structural Aspect

4.1.2.1 Description of focus (internal horizon):

When experiencing or going about learning to program this way students are simultaneously attending to tasks, feedback and the structure of the unit. The student focuses on trying to complete set tasks as a means to complete the unit. The structure of the unit plays an important role in the student's approach to learning to program.

> "…like I said, last semester, they made us hand in something each week, so we were forced to work on it instead of waiting for the tutor to explain it." (3:10-11)

The structure of the course also influences the way the students perceive their own ability in learning to program.

> "So I know I'm not on my own, and I'm getting through the assignments, and understanding the lecture notes, so I presume I'm on the right track at least." (6:8)

> "Yeah, because there's no mark-driven stuff there…. And then, when it is just purely a huge bunch of marks, just thrown at us, like [unit code] 'oh there you go there's 25% or whatever', but there's just such an overload, that you wind up… you don't know where to turn. And consequently, I don't leave it – I stress over it like there's no tomorrow, and I put plenty of time in, but I don't get anywhere. And so, with the [unit code] technique, you used to be able to… it was a manageable chunk and you could get somewhere and feel like you've achieved something, great. With [unit code] technique, I'm sure if you get to the end of it you'd have a sense of achievement, but like… sometimes you just don't get there." (2:11)

The notion of following the unit was also mentioned in the context of a long term goal for improving their employment prospects.

> "…so what I do is I have faith in the course. I just follow the course…" (7:3)

> "… and I'm familiar with the steps in the whole course. Like the subjects that lead on to others through the first and second etc. So in the end it means for me a job. Very practical in that sense. And my approach to the study, my motivation, approach is I want to get as much out of it as will get me to that goal" (7:3)

> "…I think that most of them had done an IT degree, so if they'd done a degree and passed, and done everything that I'm doing, then I must be on track." (7:9)

Dimensions of variation

Students learning to program use particular kinds of learning approaches and activities and understand the learning of the programming language in particular ways. Their motivation to learn is also driven by different forces. Students may also experience programming and programs in different ways. These factors have been identified as dimensions of variation across the different categories and are referred to below.

Learning approaches and activities.

Students seeing learning to program this way see trying to keep up with the set tasks as important. They seek feedback from the teaching staff or other elements of the teaching system (such as online marking systems e.g. FITSIS) in order to see if they 'are on the right track'.

> "…as I went through books, I didn't know where I was. I didn't have any feedback, I didn't have anyone to say where I was on track. But like with those exercises, when there was a task set, I put the task in and got back the feedback that it was correct …. they're the things that tell me well, I must be doing something, I must be getting on ok" (7:4)

Learning the programming language.

How the student understands learning the programming language is not clear in this category.

Learning motivation

The underlying motivation to learn for students experiencing or going about learning to program in this way is the desire to pass the course. The structure of the course, in particular, affects the motivation of the student to attempt set tasks.

> "…like we've been told that a lot of those questions will appear on our final exam, so I do try and learn them" (4:15)

> "[unit code] is in chunks, but we're not having to, if you like, be forced to sit down every week to get that 1%. Because if there is a % to be had, I'll certainly do whatever I need to do to get it. Whereas, if it is not there, I'm inclined to you know, I'll concentrate on other things. Wherever there is a mark, I'll go for it. And so... I guess that's it. If there's a mark there, I'll be inclined to sit down and do the work. Whereas if it's just sort of, if there's no mark associated with it, it'll get put on the backburner and I'll go and work on things where there is a mark…" (2:7)

> "The feedback like for example, like with checking the FITSIS and saying well what I put in last week is correct, I have my mark for that and following it along as a sort of, I'm doing ok, now I still have motivation to keep pushing along. If I did well last week, I'll try just as much this week etc" (7:4)

> "…because in [unit code] we had a lot of tutorial work due each week, that was a really good motivation for us to do the work and understand it before the tutorial" (3:6)

> "If there's a motivational force there, then I'll do the work. And if it's just, you know, we've covered it in the lecture, I just go 'right ok' and I sort of float along to the next one" (2:8)

Ways of seeing programming and programs

When experiencing or going about learning to program in this way, students see programming as a task to complete in order to pass the unit. There is no apparent reference to a broader context in which the act of writing programs takes place other than within the unit in order to obtain marks. Satisfaction comes from writing a program that will get the marks.

> "I certainly had a sense of satisfaction when I finished it. I had this concept – once again it was marks-driven, which works for me but I suppose not for a lot of people, but I just wanted to have the best product that I could….. that would get me the marks, and it was yeah, definitely a sense of satisfaction." (2:8)

> "But, like, last semester I probably could have written a good program …. but I don't think this semester I would write a good program" (4:10)

4.1.2.2 External horizon

The act of learning to program is seen within the bounds of the learning institution. Issues such as getting marks, time spent on completing assignments and feedback from teaching staff form the students' world in which they are learning to program.

4.2 Category 2: 'Coding'

4.2.1 Referential Aspect

Learning to program is experienced as learning to code. Students experiencing or going about learning to program this way see learning the syntax of the programming language as central to learning to program. They are driven by their belief that they need to learn the code in order to program. This may involve rote learning. As in Category 1 time is a major factor because of the amount of syntax that needs to be learned or practiced in order to get through the course. This may lead to frustration with the course. Students experiencing or going about learning to program this way may desire extra guidance from staff in terms of directing them to specific solutions and examples of code. That is, time taken to explore concepts and discover their own solutions, is considered wasted.

> "… I mean this probably relates to other questions, but, it's just the amount of time that you're spending. At the moment the way I'm learning to program is spending hours and hours and hours in front of the compiler." (1:2)

> "…well I've already spent 12 hours reading the text, why can't you just tell me exactly what page it is on, instead of making me spend another two hours trying to find an example you know?" (1:6)

> "Sometimes you just get up and try to do something else, but it's just annoys you, it really does annoy you. Just keep going until you get a result and then you're usually too tired. You go to bed and hand in an unfinished assignment. Yeah, that's the biggest thing I really want to stress, that, you can't program properly if you're given too much…. you can't learn to program if they make the goals too, the goalposts too far away, you know you really can't.. it's just … yeah" (1:12)

4.2.2 Structural Aspect

4.2.2.1 Description of focus (Internal Horizon)

Students experiencing or going about learning to program this way are simultaneously focussing on syntax and the coding task. Their focus tends also to be on the computer itself as they seek to spend as much time at the keyboard in order to practice their coding.

> "I think I have the concepts under control, but that's only 5% of it…. I think they don't understand just how hard the syntax is to grasp and….." (1:13)

> "…like I was saying before, you've got to sit there and try and get round the syntax and say you know, ok, I know what I've got to do but how am I going to write it, you know, and you're looking at … you've got your algorithm here that you know how to do it, but then you've got to spend two hours on your syntax getting that right and then looking up the syntax and seeing why doesn't it work here you know" (1:8)
>
> "…yeah like with all the tutorial classes and you know PASS classes, they're only as good as – well you're not sitting in front of a PC" (1:8)

Dimensions of variation

Learning approaches and activities

When experiencing or going about learning to program this way students might focus on searching for pieces of code to try out in their own attempts to code. They will look for examples in text books, on the internet and from other sources. Spending time at the keyboard is important for students who see learning to program in this way.

A 'trial and error' approach to inputting the code may be adopted.

> "..umm you basically, when you refer to the textbook we find ourselves looking for an example of code.…Sometimes you find it. Sometimes you don't. Sometimes you pop that example into your code and use it, and it still won't…. you know, you get frustrated" (1:3)
>
> "… it's really trial and error, it's really fumbling in the dark for a light switch you know" (1:3)
>
> "… I use the net an awful lot, trying to get examples of… sites that have plenty of source code and you know, just trying to find examples wherever you can because the book doesn't really …cover it" (1:6)
>
> "…I just look at the answers and that sort of thing, to see how they've done it and then see if I can do it myself" (3:5)
>
> "You try something that you've seen somewhere, try and fit it into something that you need to do…. If it doesn't work, you try and work out why and why not, and my understanding of the theory behind it is probably limited, but I can get it to work through trial and error, by changing variables or changing things over so that eventually you get a list of errors and you solve each error as you go, so there are no errors and that means that it pretty much works." (13:3 / 4)

At the same time students may feel that the time taken to look for code to use is a waste of time rather than a process of discovery. Input from 'expert programmers' is sought and intensive direction from teaching staff is expected.

> "The best form of help I've found is having a person beside you who knows the code". (1:4)
>
> "You can… I mean the tutors and that say you can get on the email and email but you're looking at 24 hours turn around for a response. Ummm you can come and see a duty tutor, but there only on

> at certain times and you can't fit into, you know you might have another subject on at that time, a lecture on somewhere you know, and there's only 4 windows of opportunity to see the duty tutor you know" (1:3)

> "You know they give you bits, and then you've got to work out the rest you know, and like I said before, when you're a beginner and you're already spending the amount of time a week you're spending on it, it's just frustrating. You're hitting your head against a wall and it's like, yeah ok, sure you can do it, sure you can find those bits that are missing and put them in there, but it just, you know, why not just give it to you and let you get on with subject, you know.." (1:7)

If the student doesn't receive the kind of guidance expected, they may show strong indications of dissatisfaction with the course.

> "I'm learning to program, but most of it is what I'm teaching myself, because…what the institution is giving us isn't really, doesn't help at all. Yeah, it's all really self-taught pretty much" (1:12)

> "The time thing, the time factor you could take on the chin if there was help to reduce, you know, it goes hand in hand – you would [?..only] spend as much time on it if there was more duty tutor time available, if there's people readily available to speak to ….. You know, the tutorial sessions, you've got a hundred questions after the tutor sessions and bang they're off, they don't have time, they've gone to the next tutorial you know. And I mean you're left standing there…. Either they've got to do the … make more tutorial sessions available for the students to sit down and understand what the … an experienced programmer or someone who knows the subject well, or reduce the workload you know. I feel pretty strongly about that. That's why I'm here I guess." (1:11)

Learning the programming language.

In this category students experience learning the language as learning the syntax. The understanding of learning the programming language in this way influences the learning activities and approaches used by the student. Learning the language provides a means to practice their knowledge of the syntax and vocabulary.

> "…it's like you're trying to learn Russian and you've got to know the words you know….going over the lecture notes before the lecture and that, you know, (laughs) doesn't make any difference, because you have to learn the syntax you know, and you can't learn the syntax by just reading the chapter" (1:4)

> "…learning programming is a bit like learning Chinese. You have to sit down and put in the hours and just do it. And even when it sometimes is not exciting stuff, you know sitting staring at code is like sitting staring at Chinese." (7:5)

Learning motivation

The students' approaches to learning are influenced by their perceived need to learn the syntax of the programming language in which they are working. Perseverance and a lot of hours spent in front of the computer will feature in the behaviour of a student who experiences or goes about learning to program in this way. The combined demands of assessment for their programming unit with their other subjects, may cause high levels of frustration and will possibly affect their motivation to continue with the unit.

> "…then you've got to start doing hand's on stuff to really let that soak in, and when you start doing that, when you start writing your programs and trying to compile and stuff, it really is, really it is just the time that is so frustrating – that time, when you've got other subjects to do you know." (1:2)

> " you noticed the class shrunk dramatically you know, because people just couldn't hack with it you know. I mean I'll probably get across the line, but I'm just a little bit annoyed I don't have the opportunity to perform as well as what I should and I think it's just, it all come down to the workload of the subject" (1:11)

Ways of seeing programming and programs

When experiencing or going about learning to program in this way, students see programming as the ability to write code which consists of a very specific syntax. In other words, the more code one knows, the better they will be able to program. For these students, programming is a very 'hands-on' task involving a lot of time spent in front of the computer

> "…but it's always, it will always be, in front of a PC. 99% of the time" (2:9)

> "I'd do it in front of my PC. Always. I hate using the piece of paper, because you obviously, you write something and scribble it out, you start again and scribble it out. Whereas on the PC you can pretty much type it in. It will tell you pretty quickly in java whether it is going to compile or not and if it doesn't you know you've made a mistake. If it does great, hit the run button and see what happens. That's pretty much it." (13:4)

> "…but until I get my hands on a computer and go through it, and make mistakes, and fix them up and see where I didn't type something correctly, then I don't get ahead" (7:5)

4.2.2.2 External horizon

The act of learning to program is seen within the realm of the programming language. The focus is on the syntax that makes up the language being learned. The broader context of the program itself or the programming world is apparently not part of their awareness. The programming language is seen as a means to develop one's competency with the syntax.

4.3 Category 3 'Understanding and Integrating'

4.3.1 Referential aspect

Learning to program is seen as learning to write programs through understanding and integrating the concepts (involved in programming). When experiencing or going about learning to program this way the student sees understanding as integral to learning. They tend to seek understanding of a 'bigger picture' over the small tasks they are undertaking as part of the coursework. It is not enough to type in the code and 'see if it works', but rather the student seeks to understand what they have done in order to affect the particular outcome.

Students may view learning to program as 'building on prior experience' and therefore involving a progressive sequence of concepts. They may struggle to understand one concept before moving onto the next, although sometimes they feel the need to progress through the course is more important and persevere without the sense of understanding they seek.

> "…a good working knowledge anyway, of the concepts that we'd covered, because of those individual tutes as we went along. And then it was just a matter of sitting down and integrating it together" (2:5)

> "No matter how many books you read or how good the book is, unless you can actually speak to someone and say this part of the program is doing this and it should be doing that… you really need that kind of experience of other people to help you just understand the kind of big picture" (9:4) ….

> "I try to practice what I learn sometimes. That is important, because unless… …once I learn something I must see how it, what it actually does as such or else I find it hard to understand what is actually happening" (5:2)

> "Oh ok yeah, well it's concepts. It really is. …I mean there's a whole side of things away from the keyboard. There's a whole lot of … the concepts of how …. Ok. It's for want of a better word, I think it's synergy – basically the sum of the parts [is] …. Greater than the actual parts involved. I mean,…. you've got to learn the overall concept of the programming…" (8:13)

4.3.2 Structural aspect

4.3.2.1 Description of Focus (internal horizon)

Students experiencing or going about learning to program in this way are focussing simultaneously on the task and the understanding of concepts. The focus of attention in this category is the goal of understanding the concepts involved in being able to write a program. They achieve this by gaining understanding from doing tasks – those set as part of the course of instruction, and others which the students seek out. Rather than focusing on 'fumbling in the dark' as was illustrated in Category 2, in this category the student describes their experience of learning to program in terms of gaining insight.

> "It was really a matter of tearing my hair out and spending twice as much time as I should have, learning very simple things. Until that……until the whole concept sort of came through, you know, the lightbulb sort of shone and 'ahhh is that what it meant'" (6:3)

> "…with all the visualisation tools, it was better for me to run them over and over again and watch what happened. And change some elements and see what happened then. …but to see it visually, was just so enlightening" (6:9)

> "…You really need to have a full understanding of the whole program I think… for what I needed to do there." (9:3)

Frustration may arise from feeling they are not keeping up with their understanding as the course progresses.

> "…instead of sitting there and being able to sort of almost mentally tick off 'yep that corresponds to what I think', I'll sit there and I'll just scratch my head and go 'oh what's going on' and I'm really sort of behind the 8-ball instead of on top of it in understanding it" (2:1)

> "I really sort of struggle to keep on top of it …. Ah you want a bit more sort of detail. I suppose it is essentially understanding concepts" (2:1)

> "… I don't have a good understanding or confidence – I have enough understanding that the thing works – but I don't feel confident that I understand how the parts actually work together…" (13:3)

Students experiencing or going about learning to program in this way may realise that the understanding they seek may not be immediate and requires some perseverance with tasks they don't fully understand before it makes sense. In this case they continue to follow through the course, but unlike in category 1, their focus is on the understanding that will eventually come.

> "I sort of kept going at it, but it was difficult at the time to understand how to do certain things" (4:5)

> "…. Like I've sort of tried to take that approach to it this semester, like if I don't get something, not to worry about it too much. I just sort of go, 'oh, I'll get it eventually'. That's what happened last semester. Like I didn't understand anything and I stressed out a bit and that, and like at the end I

> thought that wasn't so bad, why am I stressing out? Like there was no reason for it. So I've thought, oh I'll I give it a go. Like the approach this semester is, I'll try something and if it doesn't work, you know, oh well, it's not a big deal. It's just sort of a thing that I'm trying to see how it goes" (4:12)

> "And, so the difference is now, my… I don't waste as much time. I don't waste as much time. I think in the beginning I was like going in circles, trying to… doing things that weren't as useful. Now like I think I've a character to take each piece and try to go over it again and again until I know it really well and go to the next part. And I would use different books or different materials. In some cases I could and that's good. In some cases that wasted time and it prevented me from taking the next step which was needed to understand, to catch things that are at the back…" (7:6)

> "They were teaching a concept in 107, it didn't make sense. But half way through one of the other subjects, the same sort of thing came up again, different naming conventions, different… but you could see the underlying concepts was the same, and once you sort of saw that, it made what was said in 107 clearer. Like I didn't get it until I got that point there." (8:15)

Frustration or dissatisfaction with the structure of the course may be evident if the material is not presented in a way that matches their understanding of learning to program as learning (and integrating) a series of concepts.

> "… [unit code] has been really bad for this, is that you don't sort of get a nice building block and then lead you up to.. …so they teach you all of the components, so you know all of the components and then step you into say a big assignment which you can then go and use all of those components and understand everything. In something like [unit code], they just go at the start of the year, 'oh just go and do it' and it doesn't fit my learning style very well, because you've just got this mass of information and I don't know… I suppose this gets back to just sitting there and staring at a monitor, because you've got a mass of information you've got to try and interpret and work out just what goes where, and you don't even understand the individual components." (2:2)

> "So there's no visual kind of design approach. What we get taught is everything that exists within one of the things and that's all code, but we don't actually get taught how these methods or boxes of code, actually relates to one another." (13: 3)

> "… if you take this building and looked at every nuts and bolts and piece of furniture in the whole building, it'd be so complex. But there's none of this… you know they teach all the nuts and bolts but they don't actually show how all the parts relate. So I've written so many parts, but at the end of the assignment I can't see the forest for the trees. I know it works, but I can't understand it simply." (13: 3)

> "Now if we spent a bit of time on the overview, it'd be easier for students to know where they're going, because a lot of students don't know where they're going. They might understand a bit of syntax, but they don't understand how one part of the program will interact with another and what the parts are responsible for and how many parts you actually … or what kind of parts you need within that class and things like that. So that's my opinion, is that you learn syntax and those technical concepts, but there isn't any of this other overview how this part works with that. So more overall design kind of concept teaching would be useful." (11:6)

Dimensions of variation

Learning approaches and activities

Students experiencing or going about learning to program in this way may adopt learning activities or approaches that assist them in building up their understanding of the series of concepts they need to master in order to program. The term 'building blocks' may be used to describe their approach to gaining understanding of the concepts and in how they approach the writing of their programs.

> "…I… missed … a lecture or two, and then from missing a lecture, I didn't have much sort of to get out of the tute, and because some of the …. topics are a bit more advanced, and they all sort of build on each other …." (7:5)
>
> "…in sitting down and just doing like a small building-block piece of code, it would give you an understanding of the concepts" (2:2)
>
> "And I sort of wrote it in parts. I'd get… I think there were 4, 3 or 4 menu options, so I'd just sit down and go 'ok, I'll look at menu option 1 today' and then go off and try and write the code for that, and then wrote the code for all the options and then tried to integrate them all back into the…. So you went from the option menu, then you stepped back into the code, and I knew each of those sections worked, so, by doing it in a modular way – which is what they teach us anyway, you sort of test it as you go and then hopefully when you put it all together it will all work well" (2:5)

Variation is prominent in the learning activities and approaches associated with this way of seeing learning to program. Implementing the same thing in different ways may be used to develop understanding of concepts. The terms 'fiddling' and 'playing' may be used by the student to describe their activity in this regard. With this way of going about learning to program, experimenting is a part of learning to integrate prior experience and understanding.

> "…If I come across something that I don't understand, trying to implement it. If it doesn't work, rereading what I've been told. And fiddling around, playing. And I'm not normally afraid to try things that I'm not sure will work or not. But, if they do work, the first time, I'm often not satisfied with that because I want to know if I've really got the concept clear or not, and I'll change a few things and see if that makes a difference. And if it does, the whole concept is really a lot clearer.
>
> *Ok. So, the practicing, what does that really involve?*
>
> Well say you learn a concept, like a construct like a loop. Practicing, I mean implementing a loop several different times, several different ways, several different jobs, so that I have practice with the syntax and I have practice with using it I guess. Yeah" (6:2)
>
> "…. And because I wasn't sort of afraid to just go outside the box and try 'what if I took this part from another program altogether … and it worked, sort of thing. And I did it a lot. Like a lot of programs I've got, a lot of things I've done have been where I've tried to mix two … I suppose, ideas or two different feelers together." (8:16)
>
> "…you don't have a fear of applying something new" (8:15)

When experiencing or going about learning to program in this way, students may also seek variation in the sources of explanations of concepts in order to improve their understanding.

> "in that one contact hour of lecture there is quite a lot of information and it is hard to get through all that into your head and so…. I have to use other means of trying to understand" (5:2)

> "I use the internet quite a lot. Almost every lecture I'll take some of the keywords from what the topics were, go to the internet, search them, and in some cases I'll find examples that are better than the ones in the lecture, that are much more clear – or could be explanations that are more clear for me. Yeah…. I always look for another perspective to look at something again. That's helped me a lot." (7:7)

> "…the best thing you can do is go and find the same information elsewhere, by someone else or by some other method. Because, normally if you can see the whole thing reproduced again, but in a slightly different vein, different flow, it actually makes sense to you" (8:15)

> "…I mean it's a constant learning process, but you can't just go to one place and get the answer" (8:14)

The use of visual tools may be important in order to develop understanding of the concepts.

> "Often when I read about something and I don't know what it means, or I don't quite understand the whole thing, if I try to map out what I read, when I see it, it sort of makes a little bit more sense." (6:4)

> "So the way he arranged the lectures, when he introduced a new subject, he had a visualisation tool. And that was just brilliant for me. That was just perfect. So you could see, not physically how it works, but logically how you could view it to be working. And that was just invaluable. Without that I don't know how I would have gone." (6:4)

> "Sometimes I have to draw a map of what I think it is going to look like, well what it looks like logically I guess" (6:4)

> "…and it made it visually, very simple to understand. But to have a paragraph of text to tell me what this whole thing was all about, wasn't enough (laughs). And I really didn't know what I was going to do. I kept on going over and over and over it, not understanding it. And eventually I found somebody in the commercial world, who was actually a student here at the time, who was able to tutor me and he simply drew a picture of it and said, well, there's your elements and this is how they're numbered – the indexes, and I said 'well, is that it?!'. It was as simple as that, but I just couldn't get it from the text." (6:3)

> "I really think that a more visual approach of the concepts would really help because the nature of IT is very easy to get focused on the code" (11:6)

Students experiencing or going about learning to program in this way may also adopt a trial and error approach to learning. Unlike in category 2, however, the adoption of this approach is linked with the level of understanding they either have already, or are seeking. That is, they might see the use of a trial and error approach useful within the context of their current level of understanding:

> "Like I said before, it's trial and error, but if I didn't have the understanding, the trial and error wouldn't happen. You read the text book and get a reasonable understanding, I can't possibly say that it's a great understanding, but it's a reasonable understanding, and from there, you can play with it to get a greater understanding, and from there, trial and error to see how it fits in your code."(13:7)

Likewise, some students with a focus on understanding as a fundamental part of learning to program may feel that a focus on coding and a trial and error approach is their only means of achieving that desired understanding. Once again, unlike in category 2, the focus is not the code itself, but on coding as a means to achieving understanding.

> "My approach was to… attend the lectures, learn a few principles and then to really just hack away at the code. Although we were taught test plans and writing pseudo-code and design, I found that the nature of the problem, the programming, because it is so ethereal, it doesn't exist, I can't see it, I couldn't visualise the parts of the program. And because I couldn't visualise the parts of the program, you know, this method here and the constructors here, because there was no visual input, to visualise the parts of the program, I couldn't actually design the program before I implemented it – writing code. So I had to write code first and then hopefully after I'd written the code, maybe I'd understand the parts of what I'd written" (11:2)

> "I still don't think I totally get it, like really… maybe you don't ever, but I think more and more it's just a subtle thing that happens as you are equipped with more tools - from programming. I don't think that you can directly get that understanding – it doesn't matter how much you sit there and read that one explanation of what a class and an object or whatever is, you've just got to keep programming … Like I think really it has to be that hands on experience." (10:5)

> "Once again, that's just trial and error…. I think with more knowledge, I'd have more capacity to see, in the source code, what's not working. But at this stage, there were times when it was just random. 'Maybe if I just move it over here it will work' kind of thing." (10:3)

> "Practice is the best thing to learn, not just reading but also doing things.
> *So when you say practice?*
> Just programming. Just typing little codes even when you don't know it. Like in the beginning were typing these little programs and we had no idea what each word mean, slowly slowly, we were sort of understanding what it was all about, what it was doing.
> *Right, so you get that understanding through practicing writing code?*
> Yes. I don't think there is any other way of understanding it." (12:3)

Similarly students may be aware of a change in their own learning over time, with a shift from a focus on code to a focus on understanding as they become more proficient with the use of code.

> "At the beginning of this summer term, you know, there's obviously a very big focus on syntax and you know, leaving off a semi-colon would be disastrous ….. so now my concern is no longer the syntax, but the concepts of how things work together" (11:3)

Learning the programming language

Students experiencing or going about learning to program in this way see learning the programming language as gaining insight into the structure and logic of the language – in essence, how the language works. The language provides a means to express the concepts of programming.

> "Ummmm learning to program…..I guess it's just getting to grips with the thinking behind writing computer programs and then getting to grips with the code, the semantics of the language and … the kind of logic behind it all." (9:8)

> "…I go back to the concepts of the language. Learning the way that language kind of thinks. You know, the way they structure the programs, the way they structure information, how you know, how action x cause reaction y – that sort of thing" (8:14)

The desire to understand the structure of the language may necessitate a focus on code, however, the code is seen from a broader perspective than in category 2. For instance, being able to read and understand code facilitates and is inextricably linked with learning the language of programming.

> "…when you start out programming, it's a very new way of reading a document. Like visually it's a very different, like you know there's certain standards of alignment and to really understand right this then leads on to here which does this. It's a really new way of seeing things. And I think that that's the big thing, is the visual. To find a problem in your code, you've really got to be able to read it and see it. So I think that's the big step also. Apart from understanding what programming, code does, whatever, is learning to read code – that's a big step I think." (10:4)

Learning Motivation

Students going about learning to program in this way are motivated by their desire for insight. They are working towards a goal of understanding more than simply the task they are set as part of the unit requirements. That is, they are aware of a bigger picture, or have a greater goal than simply completing the required assignments and this affects their learning motivation.

> "One of the interesting things is of course, because QUT is not in isolation you're going to find there's an IT subject similar to [unit codes], you know it might be in Massachusetts or it might be in England or somewhere, but they might have their course notes, and sometimes even reading their stuff is actually worthwhile, because they've got a different take on the whole thing. And in that respect too, you've got to realise that the lecturer or the tutor you've got isn't the be all and end all". (8:14-15)

Ways of seeing programming and programs

Students experiencing or going about learning to program in this way see programs as consisting of syntax, code and concepts which are integrated in different ways to form the program. Being able to program means being able to apply concepts in different ways, to use what has been done in the past in new contexts and to have an understanding of broad principles involved in writing programs.

> "I mean now days of course I can actually start an application from scratch, because I've gone through the steps of actually building previous applications, so once you've got a couple under your belt, it makes sense to actually… you've got an idea how to start it and you can almost dive in and complete the task." (8:2-3)

4.3.2.2 External horizon

The act of learning to program is seen in the realm of programs and the concepts that underlie programs. The perceptual boundary is not limited to the language currently being learned, nor to a single program but to the idea of programs and programming in a broader sense. In this sense the student might refer to 'universal' concepts or principles and be aware of techniques, concepts or strategies from their past programming experience that they can bring to the present learning situation.

4.4 Category 4 'Problem solving'

4.4.1 Referential aspect

Learning to program is experienced as learning to do what it takes to solve problems. When going about learning to program this way the student has the problem as the starting point and sets out to discover the means to solve that problem. The understanding that is sought in category 3 is a fundamental component of this category. As in category 3 understanding is obtained through adopting a 'big picture' perspective, or trying to see the problem and the program as part of a broader context.

> "Whereas if you can sit down and you know the individual components but you've got this larger problem to solve, well then you can go in and do it" (2:2)

> "…the best way that I could learn that would be to, obviously look at what we've been given – go to the lecture, and then get a small problem, and sit down and just work through it and try and understand it as I went through, what each step was doing and then…. then at the end of that process, I would understand that concept" (2:7)

> "…the best way that I could learn that would be to, obviously look at what we've been given – go to the lecture, and then get a small problem, and sit down and just work through it and try and understand it as I went through, what each step was doing and then…. then at the end of that process, I would understand that concept" (2:7)

4.4.2 Structural aspect

4.4.2.1 Description of Focus (internal horizon)

Students experiencing or going about learning to program this way are focussing simultaneously on the problem to be solved and understanding concepts. The problem itself, or task to be achieved, provides the motivation to learn how to write the program and also provides the means to achieve understanding.

> "I tend to want to achieve a task in whatever field I'm currently in and I basically work out that there isn't already a resource available that does what I need so I tend to come up with a….I basically have a problem I need to solve and that gives me the …. inspiration…to try to solve it"(8:1-2)

> "Yeah, yeah, it's good to actually have a real task. Because unless……because if you're just writing programs from specs, out of books, it doesn't mean much to you, unless you've got something to actually do, then you find you lose interest. So it is good to have something to work on." (9:10)

As in category 3, prior experience is considered valuable by students who experience learning to program as learning to solve problems. The focus is not the understanding itself that is gained through building on prior knowledge, but the ability to come up with solutions to problems.

> "...you're the one inventing – if no-one's come across that problem before, you're the one inventing how to do it." (8:10)

Dimensions of Variation

Learning approaches and activities

For students who go about learning to program in this way, the problem is always seen as the starting point. Coding may be used as part of the learning process but it is within the context of solving the problem, not as the focus of what is to be learned.

> "That is just sit down and that is, for example, with those tutes in 410… sitting down and just a small… you could get your head around just a small problem and sit down and make it work. Code it out. Coding was the best way to learn, I mean it was the only way that I could learn, but having it in a manageable sort of sized piece to learn." (2: 1-2)
>
> "You've got your problem, and I'll go and sit down and usually before I start actually coding I'll sit down and I'll just have a little look, try and knock out some sort of a basic algorithm… pretty sketchy. And then it's really just a matter of textbook there (points to left), computer there and type in. You know just start and then when you compile and see how it goes and in theory hopefully it comes out the other end and works. And if it's a small enough problem, then the chances are that you'll be able to get it within a few hours, you can sit down and make a solution to it" (2:3)

When experiencing or going about learning to program in this way, there is acceptance of the importance of the planning involved prior to actually typing in code. There is a tension, however, between the desire to solve the problem in this way and wanting to 'jump in' and start to code.

> "…I mean the thing that they stress is that… just say you've got 12 months to do a program, really the first 9 months should be all preparation without actually touching the computer just about. You know like no coding. You should be solving the issues and getting the flow right, and that sort of thing. And I agree with that. It just … I guess to me it's a bit frustrating the fact that you have to wait that time before you actually get in there and play with the code." (8:4)
>
> "Well a lot of times I would pretty much have my task in mind, and I would just jump in and start writing the code. And of course as you go along you may think…. You may come across a better way to do it, or maybe you have a stroke of genius at 3 o'clock in the morning but the next day you – it works, but you can't remember what you did. That sort of thing. Or just the fact that you've got this program written and you think, now if it could only do that. But it's a major job to go back and write the whole thing because… you think it is only a little bit to change, but you have to alter the whole thing to do it…..Where if you had planned it to start with, you could have almost left openings for where the programs would go later on. Not knowing where it is going to go, but at least leave, you know, the possibility of it going there. Which is sort of what 411 is trying to do. I understand that part of it." (8:4)

Learning the programming language

Students experiencing learning to program this way may see learning the programming language as the means to solve their problems. The language itself may simply be a means to solve the problem, but is not the end in itself. It is in this way that more experienced or confident students who go about learning to program in this way do not hesitate to apply techniques they have learned in other languages to solve the problem in Java.

> "Part of the thing I think that people get wrong is that they think they're learning Java, they're learning a program. Where I think in many ways, you've got to learn the overall concept of the programming" (8:13)

Learning Motivation

Students experiencing or going about learning to program in this way are motivated by the problem they are attempting to solve.

> "I basically have a problem I need to solve and that gives me the …. inspiration…to try to solve it" (8:1-2)

They may be inspired by the problem which has been set as part of the course work, or may seek to solve a problem of their own choice.

> "…because if you're just writing programs from specs, out of books, it doesn't mean much to you, unless you've got something to actually do, then you find you lose interest" (9:10)

Ways of seeing programming and programs

Students who experience learning to program in this way focus on the utility of programs. Programming is about creating solutions to a problem or being a means to achieve some task.

> "I guess something that helps you achieve some sort of task. Something that makes life easier. Something that can be automated" (9:8)

4.4.2.2 External horizon

As in category 3, the act of learning to program is seen in the realm of programs and the concepts that underlie programs. Programs are made up of various concepts and components which are integrated in different ways to form the program, but the program is seen from the perspective of the problem which is to be solved, or the task to be achieved. Once again, the perceptual boundary is not limited to the language currently being learned, nor to a single program but to the idea of programs and programming in a broader sense. Within this category the student sees their past experience as making up the context of the learning situation and builds on their prior knowledge as a way to learn new solutions.

4.5 Category 5 ‘Participating’ or ‘Enculturation’

4.5.1 Referential aspect

Learning to program is experienced as learning what it takes to be a part of the programming community. Understanding of what it means to learn to program encompasses the way that programmers think as well as what a programmer actually does. The previous focal elements of syntax, semantics and the logic of the programs are acknowledged in this category, but the meaning of what it means to learn to program extends to the actual programming community.

> “Ummmm learning to program…..I guess it’s just getting to grips with the thinking behind writing computer programs and then getting to grips with the code, the semantics of the language and … the kind of logic behind it all. Just basically… it’s almost a different way of thinking, computer programming. You’ve got to be much more logical and you’ve got to basically write out every single step you’re going to do, whereas if you say, give someone directions to go somewhere, you miss out bits… they fill in the blanks. But with programming you’ve got to put every single exact step, and if you miss out anything, generally your program won’t function the way you expect it to….So I guess it is thinking like a programmer” (9:8)

> “…it is a real … it’s a mental leap. If you haven’t been in contact with programming before, it’s like this complete shift, like visually and also mentally, you’re kind of going …. “so what am I doing here?”
> *A shift and a leap in terms of?…*
> In terms of the way of thinking. You know, because day to day you don’t necessarily need to deal with these issues in other areas, in other industries or whatever.” (10:8)

> “I think the most important thing is just to realise that it’s a different way of thinking.” (12:2)

4.5.2 Structural aspect

4.5.2.1 Description of Focus (internal horizon)

When experiencing or going about learning to program in this way, the student focuses simultaneously on the tasks, understanding, prior experience and what they understand a programmer ‘to be’. The understanding of ‘what a programmer is’ might be linked with a student’s perceptions of how a programmer thinks, even if that student sees a difference between their own thinking and that of a programmer.

> "…I think I think more like de Bono does as opposed to the procedural way that a lot of programmers do… I think that actually helps me I must admit. That I can look at... when I'm looking at something I'm looking at the whole picture, and that's where I can see the fact if I did it a certain way in another program, in another language, it makes sense to take it across." (8:15)

Dimensions of Focus

Learning approaches and activities

When experiencing or going about learning to program in this way, students' understanding of what it means to be a programmer may affect both their approaches to learning as well as what they expect to gain from their studies.

> "…in my own little bit of research in job ads and what the market needs and the sort of programmers and what skills they need to have, there seem to be little niches of people that are familiar with certain aspects. And looking at that, it didn't… I think in the past when I had no concept at all I thought I had to know everything. Like I had to come to university and learn all the languages inside and out and know all the libraries and become some type of guru. That it was coming like this stereo… you know, stereotype of a computer guru that sits 24 hours at the computer" (7:3)

> "There's definitely a mindset. There's definitely a mindset of programming. And I think a lot of people go in with the wrong mindset about how to do it…" (8:13-14)

> "… there seem to be little niches of people that are familiar with certain aspects…. I think in the past when I had no concept at all I thought I had to know everything. Like I had to come to university and learn all the languages inside and out and know all the libraries and become some type of guru…" (7:3)

> "…now, I guess I have a different mindset for what's involved" (7:2)

> "… I think in the beginning I – just because I didn't have a clear understanding of where it was all going, like I didn't have an idea of what a programmer actually does besides just the concept of he must just sit and type out code all day …. You know, whether he had to discuss, or turn up to meetings or discuss projects on a certain scale, or… big scale work and small scale work or whatever…And, so the difference is now, my… I don't waste as much time" (7:6)

> "But you know, the thing about the [unit codes] is I guess you're learning all the aspects whereas when you get out into the real world, you won't be doing all the aspects. You won't be doing… like there'll be someone who basically their role is to write testing procedures or write procedures, on how to do something. And then your job, when you come in, is to basically follow those procedures where possible…" (8:8)

Communicating with other programmers or the programming community is a strategy adopted by the student who sees learning to program in this way. Communication might occur through email/discussion groups and by attending meetings. Students might also approach teaching staff to find out more about the direction of their studies.

> "Yeah, I just…. the goal…. like I've been to a couple of Java user group meetings. They have them in Brisbane once every month or two and I turn up there and sit in on a meeting for an hour. They have a guest speaker and they have 20 or 30 programmers in the room that have all years and years of experience working in the industry, and …. I always think to myself I wish that I could squeeze their brain a little to get an idea of I'm on the track that they were on, or that gets me to that direction. (7:9)

> "When I'm … I spend a lot of time on discussion lists, on the email and that sort of thing with other users and some of the techniques or skills that others possess are just, they blow me away" (8:1)

> "…again, speaking to I think [lecturer's name], at the start of the year, or mid year I was trying to work out where I wanted to go..." (8:17)

> "… But the main thing was people on the web. Again, more advanced users. Luckily most of them, a lot of them, worked for the actual company who wrote the software – Macromedia, so I was able to actually get… pick the brains of the people who wrote the programs, to find out how to do things." (8:11)

Learning the programming language

When experiencing or going about learning to program in this way, learning the programming language is seen as part of learning the culture of programming. It is not enough to learn the words of a language, but one needs to understand the context in which they are used. Having an understanding of the context will also facilitate the learning of the language.

> "Part of the thing I think that people get wrong is that they think they're learning Java, they're learning a program. Where I think in many ways, you've got to learn the overall concept of the programming. It's kind of like… ok, totally off this. You may be a diplomat and you have to learn French and German. But you're not learning French and German, you're learning diplomacy, you're learning how to communicate correctly…" (8:13)

> "… I mean you have to learn things like the mannerisms and all about… even some of the French history and you have to learn…." "The culture – that's what I mean. It's not literally just how to say, you know, 'here is the bagel' in French, it's the whole lot" (8:13-14)

> "Well you need to have those experiences of using the language…. Like in other languages there's certain phrases that if you directly put them into English, they make no sense at all. But if you understand the context in which you use that phrase, …. if you understand how you use that phrase, it makes all the sense in the world. I think that's the same in programming. Like you go, "ok, so that's what a while loop does", "well, why would I use it?" You need to kind of know the context I think, is the big thing, is going "ok, so that's great that it does that, but why does it do that, and where would I use it?" (10:5)

Learning Motivation

The student looks beyond the learning institution, the programming language and programs to understand what is involved with becoming a programmer. Their motivation for learning to program might relate to their desire to find employment in the programming field.

> "I could see that, for example, that there's different jobs for programmers. Like not just with languages or areas, but in the process, where they have … some programmers they write code from scratch, some that maintain. And that different levels of ability sometimes puts you in different areas. So, I guess… that gives me… I guess I take a bit of pressure off my back. I don't say to myself, well I don't have to be the best programmer in the world, because I've seen… well I know now that average programmers come out of university and go to reasonable… mediocre sort of programming jobs" (7:7)

Ways of seeing programming and programs

When experiencing learning to program in this way the student sees programming as a culture. Programs may have a variety of forms and function, however a feature specific to this category is that they are regarded as a device for communication between programmers. For instance, students who experience learning to program in this way tend to highlight the readability of their programs as an important feature of a good program. That is, they are aware that there is a community of programmers who share a language and culture.

> "The clearer the better. I mean I want to be able to read somebody else's code, so I assume that they want to be able to read mine as well." (8:9)

> "I tend to write, what I think is clear code as in most people should be able to look at it and work out what I've done…" (8:9)

4.5.2.2 External horizon

The act of learning to program is seen in terms of the world of the programmer, or the programming community. The students seeing learning to program in this way look beyond the code, the concepts, and the problems to be solved and see that they are learning about a particular world, with a particular culture. They may choose to adjust their way of thinking to join that world, or simply be aware of what to expect when they interact with that world.

5 Outcome Space

While the categories of description represent the varying ways of experiencing the act of learning to program discovered amongst the participants, the outcome space represents the relationship between those different ways of seeing. The outcome space therefore depicts the way in which the parts can be related to form a whole picture of the different ways of seeing amongst the participants interviewed (Bruce, Pham and Stoodley, 2002, p. 8). That is, it represents the phenomenon of the act of learning to program as seen by the students in this particular group. The outcome space is graphically represented in Figure 1 and the key components of this outcome space are depicted in Table 2. This depiction shows the broadening awareness brought about in the adoption of different categories.

Figure 1 **Graphical representation of outcome space highlighting the expanding external horizon of the categories**

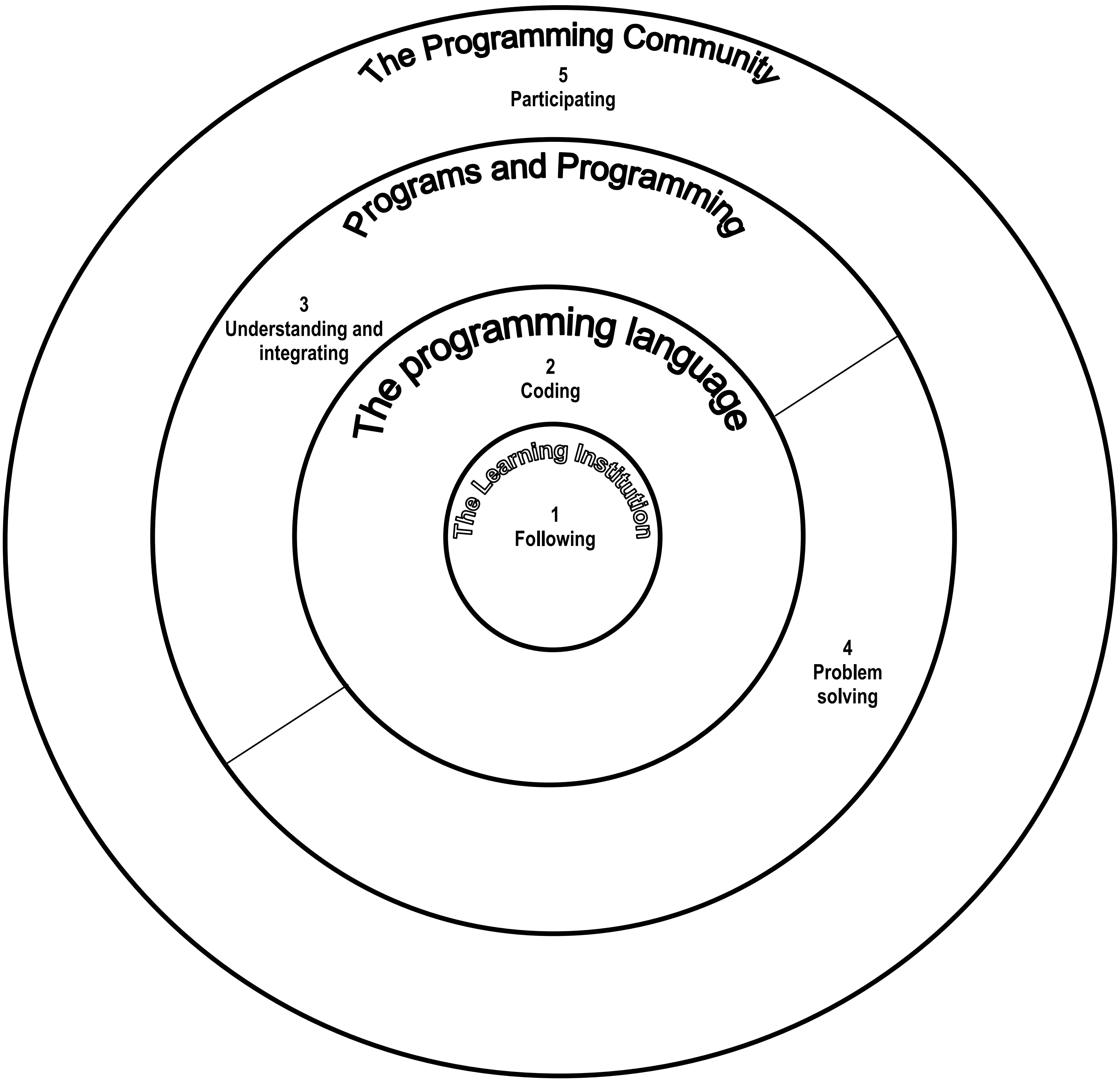

Table 2 **The outcome space of the act of learning to program**

	Label	***Referential aspect***	***Internal horizon - simultaneously attends to…***	***Dimensions of variation***				***External horizon/ (perceptual boundary)/ (Context)***
				1 Learning activities/ approaches	***2 Learning a programming language***	***3 Motivation***	***4 Ways of seeing a program/ programming***	
1	**Following**	Learning to program is getting through the unit	· Assigned task · Feedback · Structure of the unit	· Follow the course · Do set assignments, seek tutor feedback · Focus on parts rather than wholes · Surface learning orientation (e.g. rote learning)	Not evident from data	Want to pass the course	Programming is a means to complete the unit	The learning institution/context
2	**Coding**	Learning to program is learning to code	· Task · Syntax · PC	· Keyboard to practice coding · Search for code to use (examples from texts etc) · Seek guidance from staff to locate answers/examples · Focus on parts rather than wholes · Surface learning orientation (e.g. rote learning)	Means learning the syntax and vocabulary (i.e. the language is a means to practice the syntax)	Want to learn the syntax – (i.e. to be able to code)	Programming is writing code using a specific syntax	The programming language

3	**Understanding and integrating concepts**	Learning to program is learning to write a program through understanding the concepts.	· Task · Understanding (concepts& structure) · 'the big picture'□he program domaingeerl horizon □ Learning ferent wayse the other categories) many cases, in a number of different contexts and	· Integrating concepts – building on past experience · Experimentation · focus on variation – implementing same thing in different ways · seek/integrate different perspectives · Focus on 'wholes' · Deep learning orientation	Seek to understand logic/structure of the language (i.e. the language is seen as the means to express the concepts)	Seek understanding/ insight/	Programming is applying and integrating concepts	Programs/ programming
4	**Problem solving**	Learning to program is learning to solve a problem	· The task as a problem to be solved · Understanding · 'The big picture'	· Start with the problem - break it down and work it out a step at a time · See what's needed to learn to solve the problem · Focus on 'wholes' · Deep learning orientation	The language is seen as the means to solve the problem	Do what it takes to solve a problem	Programming is about coming up with a solution to a problem	Programs/ programming
5	**Participating/ enculturation**	Learning to program is learning what it means to become a programmer	· Task · Understanding · Prior experience · Understanding of what a programmer is.	· Find out what a programmer does · Try to learn how a programmer thinks · Communicate with others in the community	The language is seen as part of the culture	Wants to become part of the programming community (e.g. wants to get a job as a programmer)	Programming is a culture and programs are a means to communicate within that culture	The programming community

6 Implications for teaching and learning

The range of categories taken together suggests that some learning experiences may reinforce particular kinds of ways of going about learning to program; and therefore that teachers may wish to design learning experiences, tutorial activities or assignments that orient students towards the full range of possible ways of going about learning to program in introductory university units. Some teachers may wish to emphasise particular ways of going about learning from the set of categories described.

Individual categories raise particular implications for teaching and learning. Some possibilities are discussed below.

6.1 Category 1- 'Following'

Students who see learning to program in this way are taking a surface orientation to the subject. They are not seeking meaning in what they do but are learning answers to questions in order to pass assignments and exams. Teachers of subsequent units may, therefore, need to be aware that students coming into their units perhaps have limited understanding of the concepts they have 'learned'. For instance, students may have rote learned code to get through the assignments and subsequently forgotten it over the semester break.

The way marks are allocated throughout the unit may have an important role on student performance. Frequent and smaller assessment items may lead to more success for students who see learning to program in this way. They will tend to try to complete tasks for the marks, and will have more opportunity for the feedback so important to them in determining whether or not they are on 'the right track' in getting through the course at least.

6.2 Category 2 – 'Coding'

A surface approach to learning to program is evident in this category as in category 1. For instance, rote learning may be used in memorising code in order to complete set tasks. A focus on syntax and coding may thus also imply a less-developed understanding of basic programming concepts than would be expected by teaching staff of subsequent units.

A focus on 'parts' rather than 'wholes' is characteristic of the student who sees learning to program in the ways characterised in both categories 1 and 2. In category 1, their desire for information to be presented in manageable sized chunks, their focus on completing individual tasks as they come across them and the need for continued feedback on individual pieces of assessment, reflect this focus on learning parts with little or no attempt to place their learning in a broader context of programming. In category 2 the focus on syntax and coding, often to the detriment of understanding concepts, illustrates a fragmented approach to learning to program.

The student who sees learning to program as learning to code may feel huge pressure to learn all the code they see necessary while simultaneously trying to keep up with assignments. Although the students themselves are not seeking a 'bigger picture' (as is evident in the following categories), it may be that teaching staff need to more explicitly emphasise a broader context of programming and programming languages. In this way, students who choose to focus on code are doing so within the understanding that coding is one element of programming.

6.3 Category 3- 'Understanding and integrating'

Students seeing learning to program in this way are adopting a deep orientation to the subject. They are seeing meaning in what they do and are placing their efforts to learn to program within a 'big picture' of programming and programs. They may tend to extend themselves beyond the set assignments in an effort to further develop their understandings of concepts they are presented with. In this way, they may bring with them to their subsequent programming units a more solid grounding in programming. This contrasts with the students who see learning to program as that represented in categories 1 and 2, who may have forgotten all they rote learned in their first semester.

Teachers of those students who see learning to program in this way will most likely be expected to focus on providing context and ways to assist the student develop a sense of understanding. The use of visual tools (such as drawings, maps and animation) may be one means of enhancing this. There may be other ways which need to be explored in order to cater for those students who lack a visual approach to learning, but who still seek a deep sense of understanding. Structuring and presenting the course in a way where the student can see how each concept builds on previous ones will most likely lead to greater levels of satisfaction for students who see learning to program in this way.

6.4 Category 4 – 'Problem solving'

As in category 3, a deep orientation to learning to program is adopted by students who see learning to program as learning to solve problems. A sense of understanding is important for students in both categories. The focus on the utility of applying that understanding is the critical difference between the two categories.

For introductory students who see learning to program in this way, teachers may need to emphasise the role of subsetting problems. In other words, setting tasks which can be broken down into small problems, and facilitating the students' capacity to do this, may enhance the students' progress. Designing assignments to be done in successive stages (as modules) might be a means of addressing this.

Student choice in the nature of the problems to be undertaken as part of assessment might be particularly important for the student who sees learning to program in this way. That is, their motivation to succeed might be higher if given problems they perceive as having relevance to them.

6.5 Category 5 – 'Participating'

For students who see learning to program as learning what it means to become a programmer, it may be important that networking opportunities are built into their programming subjects. Even for beginning students, who are yet to have much past experience and understanding to build on, it would seem that contextualising the set assignments into the 'real world' of the programmer will enhance their learning experience.

6.6 The outcome space

The outcome space reveals the expanding external horizon, or perceptual boundary, of categories 1 through to 5. This is not to say that any category is better than the next in terms of potential learning outcomes, but that all categories are necessary to achieve excellent learning outcomes. That is, successful students may adopt any of the different learning approaches associated with each category at different stages of their study. It would appear that problems are likely to occur when students don't move beyond the learning experiences of categories 1 or 2. Attempts to influence the act of learning, or how students go about learning, through, for instance, changes in curriculum or the development of teaching tools such as online tutorials, need to focus on encouraging students to experience the range of different ways of learning to program.

7 Further research

The motivation for this research generally is to seek to improve the ways that students experience the learning of introductory programming. Thus, the aim of the research team is to open up the application of ongoing and future research to teaching and learning. The following points represent some of the key areas for potential future investigation which follow from this initial data collection. The main categories of future research may involve continuing work with first year students; examining changes in the act of learning over time; shifting focus onto the outcomes of learning and; broadening the teaching and learning context of the research. These areas of future research are outlined below.

1. Continue with same focus on the act of learning within first year students by:
   - Collecting more data from first year students to illuminate current findings. This may involve a revision of the interview questions
   - Conducting interviews with programming teaching staff to assess how the results of this research relate to their experiences of teaching first year students
   - Investigating critical incidents leading to change in ways of going about learning to program

- Investigating how students with different discipline bases see learning to program/programming.

Any of these approaches could be used to research the act of learning to program in any other cohort e.g. post-graduates.

2. Investigate changes in the act of learning to program over time by investigating whether or not ways of going about learning change as students encounter more programming languages by:

   - Conducting a longitudinal study either by:
     - Following the same cohort as they undertake successive units and see how they change over time in their learning approaches, or
     - Investigate how they approach learning their second, third etc languages? (e.g. C, C++, Pascal)

   - Researching second year students i.e. looking at how more experienced students go about learning programming.

3. Investigating the *outcome* of learning in introductory programming by investigating the understanding of specific concepts such as data structures or the meaning of 'object-oriented'.

4. Broadening the context of teaching and learning by developing a questionnaire to reveal correlations between:
   - Ways of experiencing learning to program
   - Ways of experiencing learning
   - Ways of seeing what constitutes academic dishonesty.

8 References

Barg, M., Fekete, A., Greening, T., Hollands, O., Kay, J., Kingston, J. H. and Crawford, K. (2000) 'Problem-based learning for Foundation Computer Science Courses'. *Computer Science Education, v.10,* no.2, pp. 109-128.

Berglund, Anders (2002) On the understanding of computer network protocols. Dept of Computer Systems, Information Technology, Uppsala University, Sweden.

Biggs, J. (1999) *Teaching for quality outcomes at university: What the student does.* Society for Research into Higher Education and Open University Press, Buckingham, UK.

Booth, S. (1992) *Learning to program: a phenomenographic perspective*, Acta Universitatis Gothoburgensis, Goteborg.

Booth, S. (1993) 'A Study of Learning to Program From an Experiential Perspective'. *Computers in Human Behavior, v.9,* pp. 185-202.

Booth, S. (1997) 'On Phenomenography, Learning and Teaching'. *Higher Education Research and Development, v.16,* no.2, pp. 135-158.

Booth, S. (2001a) 'Learning Computer Science and Engineering in Context'. *Computer Science Education, v.11,* no.3, pp. 169-188.

Booth, S. (2001b) Learning to program as entering the datalogical culture: a phenomenographic exploration. In *9th European Conference for Research on Learning and Instruction (EARLI),* Fribourg, Switzerland.

Bowden J. & Marton, F. (1998). *The University of learning: Beyond quality and competence in higher education.* London: Kogan Page.

Bruce, C. (1997) *The Seven Faces of Information Literacy,* Auslib Press, Adelaide.

Carbone, A. and Sheard, J. (2002) A Studio-based Teaching and Learning Model in IT: What do First Year Students think? *Seventh Annual Conference on Innovation and Technology in Computer Science Education*, University of Aarhus, Denmark, June 24-26, 2002.

Bruce, C. and McMahon, C. (2002) Contemporary Developments in Teaching and Learning Introductory Programming: towards a research proposal. Teaching and Learning Report 2002-2. Series Editor Peter Bancroft, Faculty of Information Technology, Queensland University of Technology, Brisbane.

Bruce, C., Pham, B. and Stoodley, I. (202) *The Collective Consciousness of Information Technology Research. Ways of Seeing Information Technology Research: Its Objects and Territories.* QUT FIT Technical Report Series. Brisbane, Queensland.

Carter, J. (2001) What they think - students' preconceptions of computing. In *International Conference on Engineering Education,* Oslo, Norway.

Carter, J. and Jenkins, T. (1999) Gender and programming: What's going on? In *Proceedings of the 1999 4th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'99*ACM SIGCSE/SIGCUE, pp. 1-4.

Cope, C (2000) Educationally critical aspects of the experience of learning about the concept of information systems. PhD Thesis, La Trobe University, Australia.

Ellis, A., Lowder, J., Robinson, J., Hagan, D., Doube, W., Tucker, S., Sheard, J. and Carbone, A. (1999) Collaborative strategy for developing shared Java teaching resources to support first year programming. In *Proceedings of the 1999 4th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'99*ACM SIGCSE/SIGCUE, Cracow, Poland.

Fincher, S. (1999) What are we doing when we teach programming? In *29th Annual Frontiers in Education Conference: 'Designing the Future of Science and Engineering Education'*IEEE Education Society; IEEE Computer Society; ASEE Educational Research and Methods Division, San Juan, Puerto Rico.

Gottfried, B. S. (1997) Teaching computer programming effectively using active learning. In *Proceedings of the 1997 ASEE Annual Conference* Milwaukee, WI, USA.

Hagan, D., Sheard, J. and Macdonald, I. (1997) Monitoring and evaluating a redesigned first year programming course. In *Proceedings of the 1997 Conference on Integrating Technology into Computer Science Education, ITiCSE* SIGCUE, Uppsala, Sweden.

McDonald. P (2002) The nature and acquisition of algorithm understanding: a phenomenographic investigation. Paper presented at Current Issues in Phenomenography: a symposium, Hosted by European Association for Research into Learning and Instruction Special Interest Group on Experience and Understanding SIG 10, CEDAM, Canberra 27-30 November.

Marton, F. and Booth, S. (1997). *Learning and Awareness*, Lawrence Erlbaum, New Jersey.

Marton, F. and Tsui, A.B.M. (in press) (Eds) *Classroom Discourse and the Space of Learning*, Lawrence Erlbaum Assoc., Mahwah.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I. and Wilusz, T. (2001) A multi-national, multi-institutional study of assessment of programming skills of first-year CS students: Report by the ITiCSE 2001 Working Group on Assessment of Programming Skills of First-year CS Students. In *ITiCSE2001*Canterbury, UK.

Ramsden, P. (2003) *Learning to Teach in Higher Education (2nd edition)*. Routledge, London.

Stein, L. A. (1999) 'Challenging the Computational Metaphor: Implications for How We Think'. *Cybernetics and Systems, v.30,* no.6, pp.1 - 35.

Trigwell, K. and Prosser, M. (1997) 'Towards an Understanding of Individual Acts of Teaching and Learning'. *Higher Education Research and Development, v.16,* no.2, pp. 241-252.

Watkins, D. and Biggs, J. (2001) *Teaching the Chinese Learner: Psychological and Pedagogical Perspectives.* Comparative Education Research Centre, University of Hong Kong, China and the Australian Council for Educational Research, Melbourne, Australia.

Williams, L. A. and Kessler, R. R. (2000) Effects of 'pair-pressure' and 'pair-learning' on software engineering education. In *The 13th Conference on Software Engineering Education and Conference (CSEE and T 2000)* IEEE Computer Society, Austin, TX, USA, pp. 59 - 65.

ACKNOWLEDGEMENTS

The authors would like to acknowledge the contributions of the programming teaching team at the Queensland University of Technology. The ongoing interest of Professor Shirley Booth, Dept of Education, Lund University, Sweden; and Anders Berglund, Department of Information Technology, Uppsala University, Sweden have also been of significant assistance in the development and completion of this project. The project team is grateful for support of Professor John Gough, Dean, Faculty of Information Technology who has encouraged us to research students' experience of learning to program, and for the financial support of the Faculty of Information Technology.

ETHICAL CLEARANCE

The University Human Research Ethics Committee deemed that this research project qualified for exemption from full ethical clearance.