Ways of Experiencing the Act of Learning to Program: A Phenomenographic Study of Introductory Programming Students at University

Christine Bruce, Lawrence Buckingham, John Hynd, Camille McMahon, Mike Roggenkamp, and Ian Stoodley
Queensland University of Technology, Brisbane, Australia

bruce@fit.qut.edu.au; l.buckingham@qut.edu.au; j.hynd@qut.edu.au; ellimac67@yahoo.com; m.roggenkamp@qut.edu.au; i.stoodley@qut.edu.au

Executive Summary

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 how the act of learning to program may be constituted amongst first year university students. Semi-structured interviews were conducted with students who had either recently completed, or were enrolled in, a university-level introductory programming subject. Analysis revealed that students might go about learning to program in any of five different ways: by (1) Following – where learning to program is experienced as 'getting through' the unit, (2) Coding – where learning to program is experienced as learning to code, (3) Understanding and integrating – where learning to program is experienced as learning to write a program through understanding and integrating concepts, (4) Problem solving – where learning to program is experienced as learning to do what it takes to solve a problem, and (5) Participating or enculturation – where learning to program is experienced as discovering what it means to become a programmer. The relationships between these different approaches to learning are represented diagrammatically. 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 are discussed. They include the following points:

1) What are the critical ways in which we want students to experience learning to program in our subjects or units or instruction, or at particular points in courses?
2) What are the implications, for students, of certain ways of experiencing the act of learning to program?
3) How can curriculum support ways of going about learning?
4) How can we help students move to more sophisticated ways of learning?
5) How can we further use the outcomes to help our students learn?

Editor: Linda Knight

Keywords: introductory programming, higher education, phenomenography, student learning, first year experience

Introduction

Although perceptive practicing teachers of programming are able to intuitively scope the perspectives of their students when it comes to learning to program, little formal empirical study has been conducted into this phenomenon. By surveying a sample of students, this study hopes to contribute to our formal understanding of the people we teach and, therefore, help influence the way we teach them. To lay a foundation from which to proceed, a review of previous related studies is given, along with an explanation of the basis of our interest in understanding students' viewpoints. Categories of students' perspectives are then described, along with these categories' relationships to each other. Some reflections on the implications of these findings are then offered.

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 discusses high failure rates as a major problem. 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. Barg et al., 2000; Carbone & Sheard, 2002; Ellis et al., 1999; Fincher, 1999; Gottfried, 1997; Hagan, Sheard, & Macdonald 1997; Stein, 1999; Williams & Kessler, 2000). Our examination of the literature on teaching and learning introductory programming (Bruce & McMahon, 2002) reveals a 'trial and error' approach to redesigning curricula largely based on the application of constructivist principles of active and collaborative learning.

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 (Bowden & Marton, 1998; Marton & Booth, 1997). 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 that bring about the kinds of learning outcomes that are desirable (Marton & Tsui, in press; Watkins & Biggs, 2001). The outcomes of this body of research, and its associated theory of teaching and learning, have had a far reaching impact on the higher education sector and development of programs for university teachers internationally (Ramsden, 2003; Trigwell & 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, 1992) education. Booth (1992) investigated variation in the outcomes of learning to program, identifying ways of experiencing or constituting programs and programming amongst advanced engineering students. Studies are also currently being undertaken investigating students' understanding of networking protocols (Berglund, 2002), and algorithmic thinking (McDonald, 2002).

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), which investigated variation in the outcomes of learning to program and in which she established students' ways of experiencing or constituting programs and programming. 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.

In conducting this research we have addressed the question: What are the different ways in which foundation year students go about learning to program?

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. Our study also opens the little explored territory of understanding learning to program from the perspective of university students undertaking introductory courses. This is an empirical study, which attempts to take the guesswork out of understanding students' views.

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. Educationally critical features of the learning community may be described in terms of complementarities in teachers' and learners' ways of seeing some aspect of the world. In the case of this investigation the pertinent aspect of the world is the act of learning to program. These complementarities (or conceptions) may be described in terms of the differing ways in which the act of learning to program is constituted, the differing ways in which teachers and learners are aware of the act of learning to program and the differing ways in which the act of learning to program appears to them. In this investigation it is the ways of constituting the act of learning to program amongst early learners that is our research object. This research object places our investigation within the terrain of phenomenographic research.

Using Phenomenography to Investigate the Act of Learning to Program

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 & Booth, 1997, p. 111). A fundamental assumption underlying phenomenographic research is that there are a finite number of qualitatively different understandings of a particular phenomenon. The research focus of phenomenography is the intention to uncover variation in the experience, or way of constituting, some aspect of the world. Sampling of participants for inclusion in a study therefore aims at capturing the breadth of variation in perspective in the population targeted. The phenomenographic approach was selected for this study on the basis of its potential to reveal variation in the ways introductory programming students experience the act of learning to program (Bowden & Marton, 1998). Thus, the current research presents a conceptual model providing insight into learners of programming, upon which decisions about teaching practice in specific contexts may be based.

In our investigation, interview questions were pilot tested and then revised before in-depth semi-structured interviews were conducted with 13 informed, consenting participants. Participants were sought from first year programming units that use Java as the programming language of instruction. The units concerned are large first year units of a Batchelor of Information Technology degree, with enrolments of somewhere between four and six hundred students each semester. The Batchelor of Information Technology has a foundational first year, after which students specialise in software engineering, data communications or information systems. First year students have not necessarily decided which area of specialisation they are going to pursue in the subsequent years of the course. Participants in our study represented a range of the student population, including some students who were studying during the summer semester where class sizes were considerably smaller. Of the 13 students nine were male, four female. Four were less than twenty years old, four were between twenty-one and thirty, four were between thirty-one and forty and one was forty. Students also varied widely in their programming experience. Three had no experience, five had basic, two had intermediate and three had advanced levels of experience. Four had studied at school, one at a Technical and Further Education college, two at university and three were self-taught. Six had gained their experience in the last four years; four had gained experience between ten and twenty years ago.

In phenomenographic research, given sampling for purposeful variation, a participant group size of between 15 and 20 is considered to be sufficiently large, without becoming unwieldy, to reveal most of the possible viewpoints and allow a defensible interpretation (Trigwell, 2000, p. 58). This sample size is not adequate, however, to justify subsequent statistical analysis. Although a larger sample size was desired for this study, the number of participants offers sufficient input to provide useful insight into the student group being surveyed. The different ways of seeing learning to program were given opportunity to reveal themselves through the breadth of representation of the student body.

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 some 'trigger' questions concerned with the students' experiences of learning to program and the different ways they see programs and programming. Each question served as an 'opening', from which the interviewer developed 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.

Data analysis aimed to develop a representation of ways of experiencing the act of learning to program, a process that was grounded in seeking variations in meaning associated with structural variation. To achieve this, an iterative approach was established involving the whole research team. The analysis process was also guided by emerging understandings of how the act and outcomes of learning may be described (Marton & Booth, 1997). Accordingly, one aspect of the experience of learning is analysed in this research, namely the 'Act' or 'How' of learning (see Figure 1).

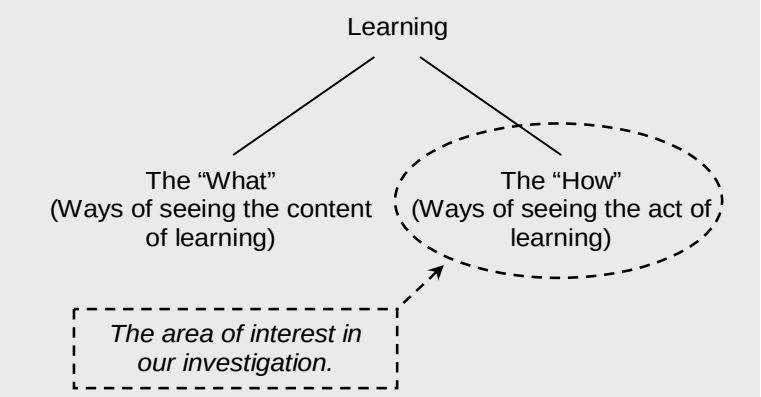


Figure 1:.Graphical representation of the domain of this research

Phenomenography develops a descriptive framework based on the two elements of meaning and structure. Meaning is represented in categories of description that regroup logically the views of the participants, simultaneously contrasting differences and clustering similarities. Structure is represented within each category and in an outcome space that indicates the relationships between the categories. The structural elements of each category and the outcome space are typically most useful for developing understanding of the phenomenon investigated.

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 that 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. In each category, referential components, that is, the critical differences in meaning, as well as structural components (Marton & 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 *Internal Horizon* represents the focus of the participants' attention, or that which is figural in awareness and simultaneously attended to. The *External Horizon* represents that which recedes to the ground, essentially the perceptual boundary associated with participants' ways of seeing.

For each category presented here systematic differences were also found in relation to students' learning activities, approaches and motivations; ways of seeing programs and programming; and ways of seeing learning a programming language. These are discussed as part of the differences in meaning associated with each category.

The logical relationships between the categories are then expressed as an outcome space. The focus on uncovering the structural framework, as revealed through the categories and the outcome space, is based on evidence from the data collected. This is fundamental to phenomenographic research and is an important factor in establishing the validity of the analysis.

In as much as it aligns with existing intuitively held beliefs, these research outcomes are also able to be validated by those who already have intrinsic, though perhaps only partial, insight into their students. The value of these outcomes lies in the provision of empirical evidence of the existence of these perceptions of learning to program and the potential provision of insight into unexpected perspectives. Furthermore, the study was conducted by a researcher who was not a teacher of programming, affording a certain distance from the field that helped avoid the projection of institutionalised views into the research results.

Ways of Experiencing the Act of Learning to Program

Our findings are presented as categories of description that represent the varying ways of experiencing the act of learning to program expressed by the participants and an outcome space that represents the relationship between those different ways of seeing. In order to provide an overview, the outcome space is described first.

Outcome Space

The outcome space depicts the way in which the individual ways of seeing learning to program may be related to form a whole picture of the different ways of seeing amongst the participants interviewed. It is an interpretation of the phenomenon, the collective experience of the act of learning to program as seen by students in this particular group. Though an interpretation, it is based firmly on the data provided through the interviews. A full research report, including illustrative quotes, is available at
http://sky.fit.qut.edu.au/~bruce/pub/papers/Prog%20final%20report%20general%20audience.pdf

The outcome space is graphically represented in Figure 2, revealing the widening awareness associated with the different categories. Also included in the outcome space are the elements in each category that are simultaneously focussed on in that category.

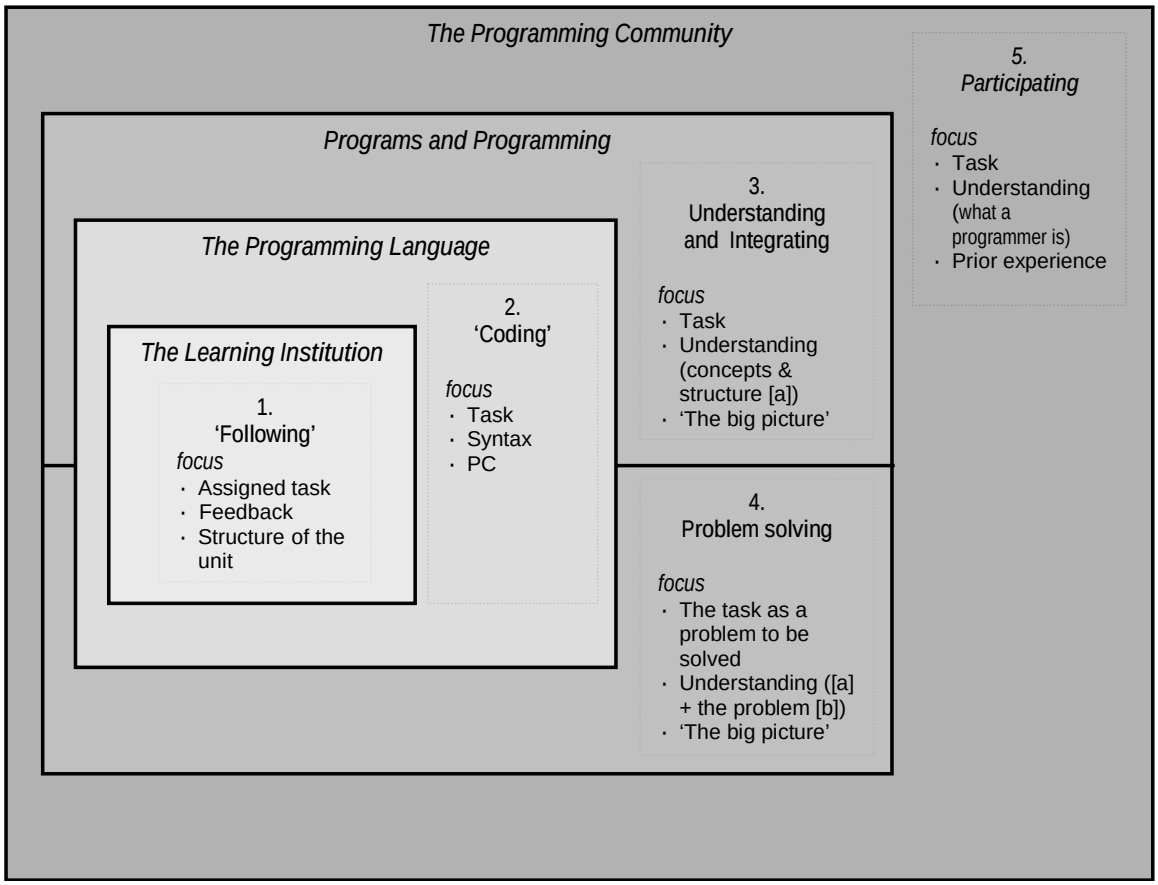


Figure 2: Graphical representation of the outcome space, highlighting the expanding external horizon of the categories

Categories of description

The categories of description revealed in the data are described below; these are illustrated with quotations from participants. The participant number and page number of the interview are noted after each quotation.

Category 1: 'Following'

In this category, the act of learning to program is experienced as following the set structure of the unit in order to 'get through'. When going about learning to program this way, a student's primary intent is to keep up with set assignments. Where marks are to be gained, students will focus on those tasks. Time might be seen as the major factor in determining whether or not the student successfully completes the unit. Students going about learning to program this way are significantly 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.

> *"…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)*

> *"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)*

Learning approaches, activities, and motivations. Students experiencing the act of 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) in order to see if they 'are on the right track'. The underlying motivation to learn is the desire to pass the unit. The structure of the unit, in particular, affects the motivation of the student to attempt set tasks.

Ways of seeing programming and programs. When 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 reference to a broader context in which the act of writing programs takes place other than within the unit to obtain marks. Satisfaction comes from writing a program that will get marks.

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

Students' focus or 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 and in the way the students perceive their own ability in 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)*

Students' perceptual boundary or 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.

Category 2: 'Coding'

In this category, the act of learning to program is experienced as learning to code. Students 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. Students going about learning to program this way may desire extra guidance towards specific solutions and examples of code. Time taken to explore concepts and discover their own solutions may be seen as wasted.

> *"…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)*

Learning approaches, activities, and motivation. When going about learning to program this way students search for pieces of code to try out, while sometimes feeling this time is wasted. They seek examples in texts, on the internet and from other sources. 'Trial and error' approaches to inputting code may be adopted. Input from 'experts' is sought and intensive direction from teachers is expected. If such guidance is not received, they may show strong indications of dissat-

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

Ways of seeing programming and programs. When 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 s/he knows, the better s/he 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.

Learning the programming language. In this category learning the language is seen as learning the syntax. Seeing learning the programming language in this way influences the learning activities and approaches used. Learning the language provides a means to practice knowledge of the syntax and vocabulary.

Students' focus or internal horizon

Students 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 time at the keyboard in order to practice 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)*

> *"…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)*

Students' perceptual boundary or external horizon

The act of learning to program is seen within the realm of the programming language. The broader context of the program itself or the programming world is apparently not part of their awareness.

Category 3: 'Understanding and Integrating'

In this category, the act of learning to program is seen as learning to write programs through understanding and integrating the concepts involved. When going about learning to program this way students see understanding as integral to learning. They seek understanding of a 'bigger picture' over the small tasks they are undertaking as part of coursework. It is not enough to type in the code and 'see if it works', rather these students seek to understand what they have done in order to affect the particular outcome.

Students may view learning to program as 'building on prior experience,' 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 so persevere without the sense of understanding they seek.

> *"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)*

Learning approaches, activities and motivations. Students going about learning to program in this way may adopt learning activities or approaches that help them understand those concepts they need to master. The term 'building blocks' may be used to describe their approach to gaining understanding of the concepts and how they approach writing their programs. Variation is also 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. Terms such as 'fiddling' and 'playing' may be used by students to describe such activity. Experimenting is therefore a part of learning to integrate prior experience and understanding. Students may also seek variation in their sources of explanations of concepts, for example using visual tools, to improve their understanding.

Some students with a focus on understanding as a fundamental part of learning to program may feel that coding or adopting a trial and error approach is their only means of achieving that desired understanding. Their focus, however, is not the code itself, but on coding as a means to achieving understanding. Students may also 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.

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. Frustration may arise from feeling they are not keeping up with their understanding as the course progresses. Frustration with the structure of the course may also appear if the material is not presented in a way that provides an overview or matches their understanding of learning to program as learning (and integrating) a series of concepts.

Ways of seeing programming and programs. Students 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 the broad principles involved in writing programs.

Learning the programming language. Students 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.

Students' focus or internal horizon

Students 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, students describe 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)*

Students' perceptual boundary or 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.

Category 4: 'Problem Solving'

In this category, 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 begins with a problem 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)*

Learning approaches, activities, and motivations. 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. 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. In this category, students are motivated by the problem they are attempting to solve. They may be inspired by a problem that has been set, or may seek to solve a problem of their own choice.

Learning the programming language. Students experiencing learning to program this way may see learning the programming language as the means to solve their problems, but not as an end in itself. 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.

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 problems or is a means to achieve some task.

Students' focus or 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)*

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)

Students' perceptual boundary or 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 that are integrated in different ways to form the program, but the program is seen from the perspective of the problem that 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.

Category 5: 'Participating' or 'Enculturation'

In this category, learning to program is experienced as learning what it takes to be a part of the programming community. Understanding 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 essence of what it means to learn to program extends to the actual programming community.

"…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?'

[Interviewer: 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)

Learning approaches, activities and motivations. 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. Communicating with other programmers or the programming community is a strategy adopted by students who see learning to program this way. Communication might occur through email/discussion groups or by attending meetings. Students may also approach teaching staff to find out more about the direction of their studies. Students look beyond the learning institution, the programming language and programs to understand what is involved in becoming a programmer. Their motivation for learning to program might relate to their desire to find employment in the programming field.

Ways of seeing programming and programs. When experiencing learning to program in this way students see 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 seeing learning to program in this way highlight the readability of their programs as an important feature of a good program. They are aware that there is a community of programmers who share a language and culture.

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.

Students' focus or internal horizon

When going about learning to program this way, students focus simultaneously on tasks, understanding, prior experience and what they understand a programmer 'to be'. The understanding of 'what a programmer is' might be linked with students' perceptions of how a programmer thinks, even if that student sees a difference between their own thinking and that of a programmer.

> *"I think the most important thing is just to realise that it's a different way of thinking." (12:2)*
>
> *"…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." (8:15)*

Students' perceptual boundary or 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.

Implications for Teaching and Learning: Some Reflections

The Outcome space and Categories of description now provide a platform for application to teaching practice. Since the specific circumstances of learning environments differ greatly, it is not possible to be prescriptive concerning the practical outworking of these research results. We can make general suggestions only. Readers will be able to use these findings as a point of comparison with and basis for application to their own situation.

It should be said that no assertion is being made that students who see learning to program in a particular way are more or less likely to achieve better results in a course of study. More research is required to establish whether indeed such links are possible. Many variables, all of which are difficult to simultaneously control, would influence this issue. Furthermore, the nature of the assessment, as much as the instruction, would have a bearing on this question.

While our categories of description are discrete, students themselves may adopt different ways of experiencing at different times during their learning experience. Essentially, their choice of focus will determine at any point of time how they go about learning to program. Biggs (1999), Ramsden (2003), and Prosser and Trigwell (1999, p. 17) have modelled possible influences on student learning outcomes including prior experiences of learning and aspects of the learning environment they currently work within. While there are potentially many factors at work influencing how students go about learning to program, as teachers of programming we are able to use the outcomes of the current investigation to ask:

1) What are the critical ways in which we want students to experience learning to program in our subjects or units or instruction, or at particular points in courses?
2) What are the implications, for students, of certain ways of experiencing the act of learning to program?
3) How can curriculum support ways of going about learning?
4) How can we help students move to more sophisticated ways of learning?
5) How can we further use the outcomes to help our students learn?

What are the critical ways in which we want students to experience learning to program? The range of categories taken together suggests that some learning experiences may reinforce particular 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. The outcome space reveals the expanding external horizon, or perceptual boundary, of Categories 1 through to 5. It would appear that learning outcomes are likely to be less successful when students don't move beyond the learning experiences of Categories 1 or 2. From the set of categories described, some teachers may wish to emphasise particular ways of going about learning at different times. For instance, a focus on learning code might be necessary early in the semester with teaching staff then facilitating the transition from a focus on code to a focus on seeking understanding and context as the semester progresses.

What are the implications of certain ways of experiencing the act of learning to program? Students who adopt a surface orientation to introductory units such as that illustrated in Categories 1 and 2 are not seeking meaning in what they do but are learning answers to questions, or strings of code, in order to complete set tasks. When entering later units, they may have limited understanding of the programming concepts they have 'learned'. In contrast, when students experience learning to program as in Categories 3 to 5, they 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 their set assignments in an effort to further develop their understandings of concepts they are presented with. In this way, they may move into subsequent learning with a more solid grounding in programming.

How can curriculum support ways of going about learning? The range of categories highlights a distinction between those students who focus on 'parts' as opposed to those who focus on 'wholes' in their learning experience. For instance, in Category 1, the desire for information to be presented in manageable amounts, the 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, also illustrates a fragmented approach to learning to program. Although these students are not themselves seeking a 'bigger picture', teachers may be able to encourage them to refocus by more explicitly emphasising the broader context of programming and programming languages, and the programming community. In this way, students who choose to focus on code are doing so within the understanding that coding is one element of programming. Such an approach may facilitate a shift, or expansion, in learning experiences across the range of categories.

Students who focus on seeking understanding in their learning experience, as in Categories 3 and 4, expect staff to provide context and ways to help them develop a sense of understanding. The use of visual tools (such as drawings, maps and animation) may be one means of enhancing this. Other ways should be explored to cater for those students who lack a visual approach to learning, but who still seek a deep sense of understanding. Likewise, students seeing learning to program as in Category 4 will appreciate material presented in a way where the application of their understanding to problem solving is highlighted. 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, by facilitating their experience of a broader range of learning experiences.

Similarly, the design of learning tasks is an important area in addressing the needs of students across the range of categories. For example, in order to facilitate learning outcomes for students who see learning as building on previous understanding, as in Category 3, 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. Where students' understanding is critically linked with application in problem solving, as in Category 4, teachers may need to emphasise the role of subsetting problems. In other words, setting tasks that can be broken down into small problems, and facilitating the students' capacity to do this, may enhance the students' progress.

Assessment tasks can also be designed to target the needs of students going about learning to program in particular ways. For instance, frequent and smaller assessment items may help students who see learning to program as getting through the course, as in Category 1. 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. Students who see learning to program as in Categories 3 and 4 may prefer assignments completed in successive stages (as modules), which build on their past efforts. For students who see learning to program as in Category 4, their choice in the nature of the problems to be undertaken as part of assessment might be particularly important. That is, their motivation to succeed might be higher if given problems they perceive as having relevance to them.

How can we help students move to more sophisticated ways of learning? We suggest that teachers of programming who want to progress their students to the more sophisticated ways of learning to program need to adapt their method of instruction in order to expose their students to those ways of seeing. This is because for learning to occur "variation must be present in the learning environment in dimensions corresponding to the aspects students have to become capable of discerning" (Bowden & Marton, 1998, p. 12). More specifically, if we want to produce Category 3 students as the outcome of our instruction, we need to incorporate Category 3 experiences in our instruction. Such experiences could be built around the several elements of the Category 3 way of seeing described here. We should not expect, for example, students who are only exposed to tasks oriented around coding specific functions to gain an understanding of overall programming structure. Nor can we expect students only focus on completing the course requirements to necessarily embrace participation in the programming community. If we want students to have a Category 5 experience, we need to design learning strategies which introduce students to this way of seeing the world.

How can we further use the outcomes to help students learn? What is it that we mean by helping students learn or 'improving learning outcomes'? It might be that increasing pass levels is not the only concern. This is exemplified in Booth's (1997) study of students' understanding of recursion, where it was noted that often students with the less developed understanding who memorized by rote did better in exams than those students with better understanding who produced seriously flawed programs. One approach to responding to the question is by endorsing the idea that learning is about coming to approach or experience phenomenon differently. In our investigation we have identified that students learning introductory programming at university go about learning to program in various ways and that each of these ways is associated with particular ways of seeing programs and programming languages. Each of these different ways is also associated with particular elements that are focal, indeed educationally critical, to that experience. While the ways of reinforcing particular approaches to learning to program are readily identifiable, seeking strategies that would bring about changes in students' focus, helping them to simultaneously discern the various elements critical to each category, presents a greater challenge. Ways of achieving such a shift need to be considered, developed and tested by teaching teams (Marton & Pang, 2003).

Conclusion

If we see learning as being about changes of experience brought about by the interaction of the complimentary viewpoints of learners and teachers, an understanding of the existing perspectives of students of programming becomes integral to our teaching practice. Such an understanding enables insight into the differences and similarities between these groups and forms a basis from which to design learning experiences which encourage change. Finally, appropriate pedagogic decision-making lies in the skilled teacher's hands. We offer the outcomes of this study as a point of departure for use with students of introductory programming.

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; Professor Ference Marton, Department of Education and Educational Research, Göteborg University; 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 the 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.

References

Barg, M., Fekete, A., Greening, T., Hollands, O., Kay, J., Kingston, J. H. & Crawford, K. (2000). Problem-based learning for foundation computer science courses. *Computer Science Education, 10* (2), 109-128.

Berglund, A. (2002). On the understanding of computer network protocols. Dept of Computer Systems, Information Technology, Uppsala University, Sweden. Retrieved October 18, 2003, from http://user.it.uu.se/~andersb/lic/

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

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

Booth, S. (1997). On phenomenography, learning and teaching. *Higher Education Research and Development, 16* (2), 135-158.

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.* Adelaide: Auslib Press.

Bruce, C. & 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. Retrieved October 18, 2003, from http://sky.fit.qut.edu.au/~bruce/pub/papers/T&L%20Report%20PB%20edited%203%20Dec.pdf

Carbone, A. & 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 Educa-*

tion, University of Aarhus, Denmark, June 24-26, 2002. Retrieved October 18, 2003, from the ACM Digital Library Web site: http://portal.acm.org

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. & Carbone, A. (1999). Collaborative strategy for developing shared Java teaching resources to support first year programming. *Proceedings of the 1999 4th Annual SIGCSE/SIGCUE Conference on Innovation and Technology in Computer Science Education, ITiCSE'99* ACM SIGCSE/SIGCUE, Cracow, Poland. Retrieved October 18, 2003, from the ACM Digital Library Web site: http://portal.acm.org

Fincher, S. (1999). What are we doing when we teach programming? *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. Retrieved October 18, 2003, from http://www.cs.kent.ac.uk/pubs/1999/917/content.PDF

Gottfried, B. S. (1997). Teaching computer programming effectively using active learning. *Proceedings of the 1997 ASEE Annual Conference* Milwaukee, WI, USA. Retrieved October 18, 2003, from www.asee.org/conferences/search/01045.pdf

Hagan, D., Sheard, J. & Macdonald, I. (1997). Monitoring and evaluating a redesigned first year programming course. *Proceedings of the 1997 Conference on Integrating Technology into Computer Science Education, ITiCSE* SIGCUE, Uppsala, Sweden. Retrieved October 18, 2003, from the ACM Digital Library Web site: http://portal.acm.org

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. & Booth, S. (1997). *Learning and awareness.* New Jersey: Lawrence Erlbaum.

Marton, F. & Pang, M. (2003). Beyond “lesson study”: Comparing two ways of facilitating the grasp of economic concepts. *Instructional Science, 31* (3), 175-194.

Marton, F. & Tsui, A.B.M. (Eds.) (in press). *Classroom discourse and the space of learning*. Mahwah: Lawrence Erlbaum Assoc.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., Laxer, C., Thomas, L., Utting, I. & 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. *ACM SIGCSE Bulletin 33* (4). Retrieved October 18, 2003, from the ACM Digital Library Web site: http://portal.acm.org

Prosser M. & Trigwell, K. (1999). *Understanding learning and teaching*. Buckingham: Society for Research in Higher Education and Open University Press.

Ramsden, P. (2003). *Learning to teach in higher education* (2nd edition). London: Routledge.

Stein, L. A. (1999). Challenging the computational metaphor: Implications for how we think. *Cybernetics and Systems, 30* (6), 1-35.

Trigwell, K. (2000). A phenomenographic interview on phenomenography. In J. Bowden & E. Walsh (Eds.), *Phenomenography.* (pp. 47-61). Melbourne: RMIT University Press.

Trigwell, K. & Prosser, M. (1997). Towards an understanding of individual acts of teaching and learning. *Higher Education Research and Development, 16* (2), 241-252.

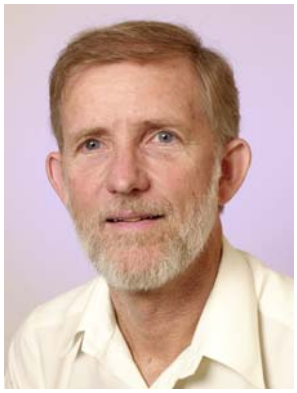
Watkins, D. & Biggs, J. (2001). *Teaching the Chinese learner: Psychological and pedagogical perspectives.* Comparative Education Research Centre, University of Hong Kong, China and the Australian Council for Educational Research, Melbourne, Australia.

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

Biographies

Christine Susan Bruce is currently Director, Teaching and Learning in the Faculty of Information Technology at Queensland University of Technology. Christine's research interests are in the area of higher education teaching and learning. She takes a phenomenographical orientation to investigating curriculum and staff development issues at both undergraduate and postgraduate level, and has for some years focused on the meaning of a lifelong learning orientation for students and teachers. She has extensive interests in information literacy and information literacy education in academic, workplace and community settings. Christine also investigates the experience of teachers, postgraduate students and researchers.

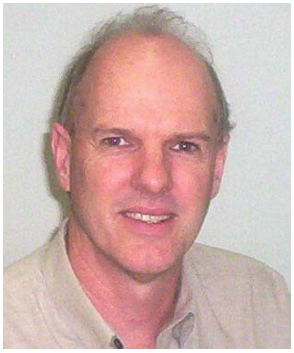
Lawrence Buckingham received his BSc(Hons) at the University of Queensland in 1982. Joining the Australian Public Service in 1984, he worked in a range of jobs including clerical, help desk and application programming positions. In 1991 he left the Public Service and worked in a PC support and application programming role in a small Queensland business. In 1997 he was awarded a Master of Information Technology degree at Queensland University of Technology. From 1998 to 2001 he lectured at QUT, focusing mainly on introductory Computer Science. Since then he has worked as an Instructional Designer in the Faculty of Information Technology, QUT.



John Hynd is a Senior lecturer in software engineering at Queensland University of Technology. He teaches language processing and compiler technology. He is a member of the team which developed Gardens Point Modula-2 and a participant in 3 QUT grants developing flexible and online teaching technology. He has consultancies with Mincom and Telstra and is Cooperative Education student supervisor and Software Engineering major coordinator.

Camille McMahon has contributed to a number of phenomenographic research projects in the Faculty of Information Technology of the Queensland University of Technology, Brisbane, Australia. These have included investigations of students' approaches to learning to program. Her main interest is in information literacy in cross-cultural development, which has led to a paper presented at a UNESCO Information Literacy Leadership Conference.

Michael Roggenkamp is currently the Assistant Dean - External Relations in the Faculty of Information Technology at Queensland University of Technology. He has had extensive experience teaching introductory programming subjects since 1977. Between 1993 and 2001 he was the Director of Undergraduate Studies in the Faculty of Information Technology. He is a member of the team which developed Gardens Point Modula-2 and is involved with the Environment to Learn Programming project (ELP). His research interests include language design and learning to program.



Ian Stoodley is a research assistant in the Faculty of Information Technology of the Queensland University of Technology, Brisbane, Australia. He has been involved in a number of phenomenographic research projects. These investigations have been of the views of IT students, academics and professionals on IT and IT research and of the views of students on learning to program. The outcomes of a number of these projects have been published in scholarly journals.