

Masters Students' Experiences of Learning to Program: An Empirical Model

Ian Stoodley, Ruth Christie, Christine Bruce
Queensland University of Technology

Faculty of Information Technology
Queensland University of Technology
Brisbane, Queensland
Email: i.stoodley@qut.edu.au

Abstract

The investigation reported here examined how Masters students who are new to information technology studies experience learning to program. The phenomenographic research approach adopted permitted the analysis of 1) how students go about learning to program, that is the 'Act' of learning to program, and 2) what students understand by 'programming', that is the 'Object' of learning to program. Analysis of data from twenty-three participants identified five different experiences of the Act of learning to program and five different experiences of the Object of learning to program. Together the findings comprise an empirical model of the learning to program experience amongst the participating students. We compare our findings with a previous study of Introductory students, suggest how our findings are significant for programming teachers and offer tools to explore students' views.

INTRODUCTION

Investigations into learning to program have largely focussed on teaching approaches with a view to helping improve students' success rates (e.g. Barg et al. 2000; Carbone and Sheard 2002; Fincher 1999; Gottfried 1997; Lister and Leaney 2003; Stein 1999; Williams and Kessler 2000). One branch of these investigations, studies into the experience of learning to program, have been conducted in Sweden and Australia amongst undergraduate students (Booth 1992, 2001; Bruce et al. 2004). These studies, based on a relational approach to learning, have adopted a phenomenographic approach to researching students' experiences (Marton and Booth 1997). This approach permits the analysis of variation in 1) how students go about learning to program, that is the 'Act' of learning to program, and 2) what students understand programming to be, that is the 'Object' of learning to program.

The experience of learning may be characterised, according to Marton and Booth (1997), using the schematisation in Figure 1. For simplicity, the Direct Object is referred to as the Object in this paper.

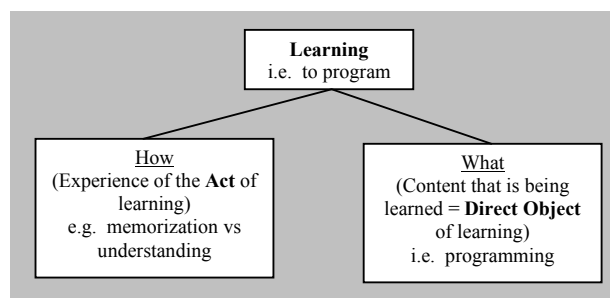


Figure 1: The experience of learning
(Adapted from Marton and Booth 1997, with permission)

Our interest in uncovering variation in the Act and Object of learning stems from the potential power of influencing student learning by designing learning experiences that draw attention to variation in the learning to program environment. If we accept, with Marton and Booth (1997) and Bowden and Marton (1998), that fundamentally learning is about changing the way in which one sees or experiences the world, then helping someone learn is about helping them make such a change. Marton and Booth (1997) argue that bringing about the appropriate change in experience or way of seeing begins with helping learners to discern that there are different ways of seeing or experiencing the phenomenon in question.

The primary contribution of this paper is that it makes explicit the character of variation in both the Act and Object of learning to program amongst a cohort of Masters students. We believe that the model we have established provides a stimulus for discussion for new as well as more experienced teachers and learners. For the educators we have worked

with, it empirically confirms insights that have come from years of teaching. The model is most likely to be easily adaptable to educators of similar cohorts, but may also be relevant to educators of undergraduate students.

RESEARCH APPROACH AND METHOD

The purpose of our investigation was to explore the different ways in which Masters students who are new to information technology (IT) studies experience learning to program. In order to be able to compare with previous studies (Booth 1992, 2001; Bruce et al. 2004) and to effectively investigate variation in the Act and Object of learning to program, a phenomenographic approach (Marton and Booth 1997) was used. The primary goal of phenomenography is to elicit variation in ways of experiencing specific phenomenon, making it the most suitable approach for our investigation.

Participants

The twenty-three students participating in this investigation were enrolled in an information technology Masters course which is intended to provide a first professional qualification in IT. After completing a core of four subjects, including introductory programming, students select studies from a range of areas such as information systems, networking and software engineering. These students are typically mature-aged, already have a degree and have had some previous formal or informal experiences of programming.

Participants were drawn from two semesters' enrolments, including different student intakes and groups who were taught by different lecturers and tutors. The broad range of participants satisfied the need for a participant profile that would allow us to elicit variation in ways of experiencing learning to program. Table 1 illustrates other characteristics of this group. (Sometimes participants selected more than one alternative or none of the alternatives offered in the profile survey.)

Gender		When first learned programming?				How first learned programming?				How skilled?		
M	F	last year	2-4 years	5-10 years	> 10 years ago	Self-taught	School	Formal course	TAFE	Begin.	Inter.	Adv.
17	6	6	5	4	8	8	6	9	1	11	7	2
74%	26%	26%	22%	17%	35%	33%	25%	38%	4%	55%	35%	10%

Table1: Participant profile

Seventeen students were interviewed and six students filled out questionnaires. Almost half the participants were international students or NESB students. Most participants had already written programs using, for example, ASP, Basic, C, C++, Fortran, Java, Lightspeed, LISP, machine code, Modula-2, Pascal and Visual Basic. A participant group size of 15 to 20, purposefully sampled for variation, is considered in phenomenographic research to be sufficiently large to reveal most of the variation possible, while keeping the quantity of data to a manageable level (Trigwell 2000).

Data gathering and analysis

A primary goal of data collection in phenomenographic research is to allow the participants to express themselves freely, within the constraints of the topic at hand. Thus, questions are designed to be open-ended to allow participants to explore the topic without being led, but at the same time directing their attention to the phenomenon of interest, in this case programming and learning to program. Our interviews were given focus, therefore, through the use of questions which directed students' attention to their experience of programming and learning to program. The questions used in this study were based on those from previous studies on learning to program (Booth 1992; Bruce et al. 2004):

1. What is "programming"?
2. Can you write a program that works? How do you know?
3. Can you write a good program? How do you know?
4. How do you see your current ability to program?
5. Can you describe the process you go through when you write a program?
6. Do you enjoy programming?
7. How have you learned to program?

Written questionnaires posed the same questions. Once transcribed, the interviews and other data were read as a whole in order to discern patterns in the viewpoints expressed. The analysis was pursued within the 'How' and 'What' parameters of Marton and Booth's schematisation of learning (see Figure 1). Variation in perspective was sought, with similar perspectives grouped together. The analysis framework established by Marton and Booth (1997) recommends a search for variation in both meaning and structure. This variation in meaning and structure were allowed to 'emerge from the data' through an iterative, exploratory process which cycled between the record of interviews, categories posited and research team members' observations. Categories of description were then crafted, to reflect the meaning

and structure sought in the analysis process. Linkages between the categories were also analysed and represented in an outcome space. Programming teaching practitioners were also asked to appraise the categories found.

Each resulting category of description depicts a single way of experiencing either the Act of learning to program or the Object of learning to program. In the categories that follow meaning is described as well as critical elements of the category's structure. Structure is depicted in terms of an internal and an external horizon - the internal horizon indicates the focus of awareness and the external horizon indicates that which lies on the periphery. For example, in category one 'Following the course' is the internal horizon and the 'Formal Learning Environment' forms the external horizon. Systematic variation between categories was also identified. For example, in the Act of learning to program across three dimensions – teaching practices, learning strategies, and approaches to learning.

In the following section the relationships between the categories are represented first, in the form of an outcome space. This aligns the outcomes with Marton and Booth's model of learning and serves as an overview of the categories discovered. The outcome space is a logical interpretation of the phenomenon, based on the data collected. Such phenomenographic analysis is explained in detail by Marton and Booth (1997).

MODELLING EXPERIENCES OF LEARNING TO PROGRAM

The outcome space depicted in Figure 2 forms a model of Masters students' experiences of learning to program. Grouped according to their external horizons, these are considered to represent a progressively expanding universe of awareness.

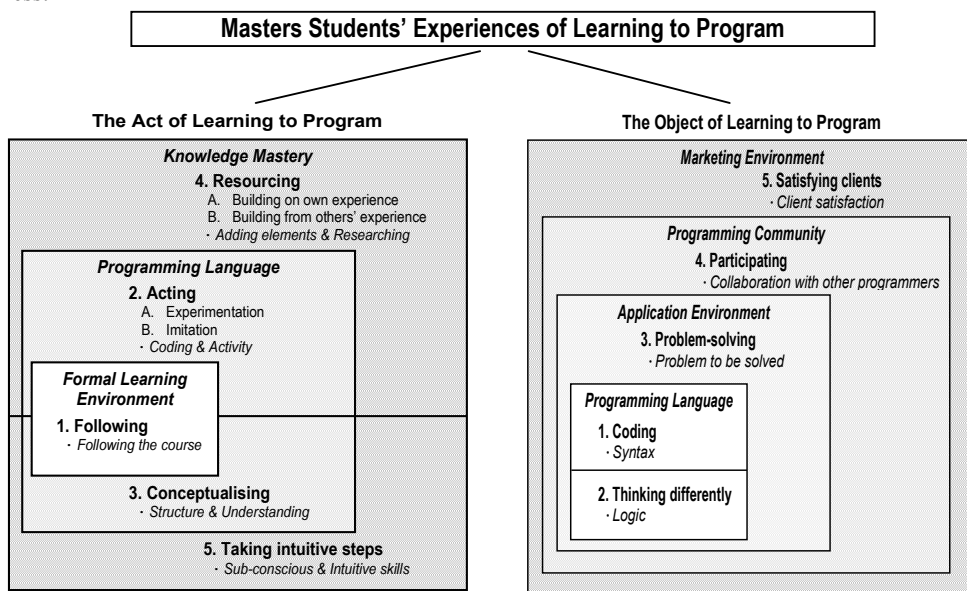


Figure 2: The Outcome Space for Masters Students' Experiences of Learning to Program

The left-hand side of the outcome space portrays the internal relationship between the five categories used to describe the Act of learning to program. The name, focus and external horizon for each category are represented in the diagram. For example, for category two the name of the category is 'Acting'; the focus is simultaneously 'Coding' and 'Activity'; and the external horizon is the 'Programming Language'. When subcategories have been identified they are listed under the category name and indicated with capital letters (A, B). Some categories share the same external horizon.

The right-hand side of the outcome space portrays in a similar manner the relationships between the categories used to describe the Object of learning to program.

The categories associated with students' experiences of the Act of learning to program may be interrelated according to their increasing sophistication as approaches to learning and their expanding external horizons. Their focus thus changes from being solely on following the course (Category 1), to simultaneous focus on coding and activity (Category 2), to simultaneous focus on structure and understanding (Category 3), to simultaneous focus on adding elements and researching (Category 4), to simultaneous focus on sub-conscious and intuitive skills (Category 5). Similarly, students' perceptual boundaries expand from the formal learning environment (Category 1), to the programming language (Categories 2 and 3), to knowledge mastery (Categories 4 and 5).

The categories associated with students' experiences of the Object of learning to program may be interrelated according to their increasing connectedness with the wider world. Thus, their focus changes from syntax (Category 1), through logic (Category 2), the problem to be solved (Category 3), collaboration with other programmers (Category 4), to client satisfaction (Category 5). Similarly, students' perceptual boundaries broaden from the programming language

(Categories 1 and 2), to the application environment (Category 3), to the programming community (Category 4), to the marketing environment (Category 5).

Each of the categories depicted in the outcome space are described further below.

MASTERS STUDENTS AND THE ACT OF LEARNING TO PROGRAM

Our analysis has identified five different experiences of the Act of learning to program: Following, Acting, Conceptualising, Resourcing and Taking intuitive steps.

Each of these categories is described in relation to their meaning, subcategories (if appropriate) and dimensions of variation. Details pertaining to their structure, as described above, are not repeated.

(Quotation references refer to the interview or questionnaire number, page number and quadrant on the page. For example, (I2.1b) refers to Interview 2, page 1, section b. Redundant insertions and pauses in the quotations are indicated by "...")

Learning to program is experienced as following

Focus: Following the course. External horizon: The formal learning environment.

In this category learners experience learning to program as following a formal course. They see responsibility for learning as resting principally on the shoulders of the instructors.

... you get frustrated if you go along to a lecture, and the lecturer just reads the PowerPoint slides to you because they're not actually teaching. They're not taking something which is difficult to understand and making it any easier to understand. (I5.5b)

When learning to program is experienced this way, learners expect that perseverance in fulfilling the course requirements will result in learning. The course requirements typically involve the completion of exercises and attendance at formal sessions (lectures and tutorials).

A *teaching practice* expected by students associated with this category is the timely provision of assignment results to learners.

Key learning strategies associated with this category are the fulfilment of course requirements and following the teacher's lead.

The *approach to learning* in this category is passive.

Learning to program is experienced as acting

Focus: Coding and activity simultaneously. External horizon: The programming language.

In this category learners experience learning to program as actively producing code. In order to learn, learners do not consider it necessary to understand what they are doing; if they are actively engaged in writing code then they believe that understanding will follow.

... understanding often follows execution, after doing something. So you have to be able to do something, not knowing what you're doing and then you'll get your true understanding after you've done it, not before you've done it. (I5:13b)

The subcategories of 'experimentation' and 'imitation' further illuminate this category.

A. Experimentation

Learners often employ trial and error, involving a cycle of coding revisions based on the results obtained. This is common in early attempts to learn to program, though it is not absent from later learning. Learners run their programming attempts with an expectation of error messages which they then use to refine the code.

I didn't understand the logic of it ... I knew that the string compare would return one or minus one or zero, but I didn't know ... what that meant or what order it would be. But instead of trying to work it out I just ran the program and I'd have a fifty percent chance of it being the right order and fifty percent wrong ... you test it and the computer gives you feedback ... and [you] learn from the test and trial and error and correct ... and ... understanding of the logic didn't happen until after I'd done something. (I5:9d)

Experimentation concentrates on the learners' own code and their own attempts at finding solutions to their programming problems.

B. Imitation

In this subcategory learners concentrate on how others' code can be copied and adapted to meet their own needs. They compare others' code with its outcomes to gain an understanding of how the program is working.

... the word imitation comes to mind. Like when I first started, I was copying programs out of a book. I was rote copying, so that's just imitating ... just typing it in, but learning about the structures ... when I came to work here, I did quite a bit of developing with other programmers and sometimes involved modifying their code and ... I'd see structures that they were using ... the concept of self-documenting code ... long variable names and ways of approaching problems and things like that, so I sort of tried to imitate ... (I4.5c)

A *teaching practice* expected by students associated with this category is the practical application of new concepts in coding exercises.

Key learning strategies associated with experiencing learning to program as acting are the copying and adaptation of examples of programming, successive drafting, response to error messages, the testing of coding attempts, the use of automated coding functions (such as macros) to produce a model, and the assessment of outputs.

The *approach to learning* in this category is active.

Learning to program is experienced as conceptualising

Focus: Structure and understanding simultaneously. External horizon: The programming language.

In this category learners experience learning to program as conceptualising the logic of programming. This is primarily a thinking process; understanding precedes action. It is important to gain a mental picture of the structure of the program before code is written.

This category is considered by the learners to be abstract and high-level, whereas coding is concrete and easily found in a textbook. Learners taking this point of view appreciate the need for a wider application of programming beyond simple coding, the need to be able to adapt what is being learned according to particular programming languages and environments. Learners place a strong emphasis on developing their own logical thinking.

... in case ... we have to learn programming ... make some good diagrams ... we draw the diagrams, we write stuff on, we write 'set to' and we assign the values to the variables and then try and understand the part actually the program is trying to do. ... Writing onto the computer ... we won't be able to understand that much, unless we sit down on a table and then we ... actually write the whole code or draw the diagrams on the piece of paper. (I1.4c)

A *teaching practice* expected by students associated with this category makes use of real world examples or employs animations, to illustrate new concepts in concrete terms.

Key learning strategies associated with this category are modelling, drawing diagrams, reflecting, writing code on paper and forming real world analogies. Learners in this category concentrate on paradigm questions, asking not only "How?" but also "Why?"

The *approach to learning* in this category is active.

Learning to program is experienced as resourcing

Focus: Adding elements and researching simultaneously. External horizon: Knowledge mastery.

In this category learners experience learning to program as gathering resources. Learners experience learning to program here as the accumulation of useful skills.

I think there's been different levels of learning to program ... You learn the basic way that a language should be laid out. You learn then at a deeper level how it accesses memory. And then you learn more abstract concepts and I guess at each stage ... it's just been building a bit on the previous thing ... (I14.5d)

Thus, learners experience learning to program as being akin to putting building blocks on top of each other, where isolated elements are added together. The goal of engaging in acts of learning is to construct a pool of resources which can be drawn on to meet a variety of programming demands. Typically, excerpts from books, lecture and tutorial notes, online help files, the Internet and people are collected into a library for future reference.

The subcategories of 'building on own experience' and 'building from others' experience' further illuminate this category.

A. Building on own experience

In *building on own experience* the student is progressing to more sophisticated concepts and methods by adding to their own experience.

Learners see themselves as taking their experience and advancing it without having to refer substantially to others' work. This may involve active coding or it may entail the re-discovery or re-organisation of past knowledge.

... where I can I do things like make little libraries or snippets of code ... once I've done something difficult once, I like to have it stored somewhere else, where I can just say, "Oh yeah, I need one of those." ... But it's important to remember where it all is if you need to pull it out again. (14.6d)

B. Building from others' experience

When *building from others' experience*, the learner is acknowledging their dependence on others. They take others' experience, conveyed in person, or embodied in books, error messages, help files or the Internet, and advance their understanding through it. Learners in this category value immediate feedback from lecturers and interaction with other learners at the very time they are writing programs.

The most valuable lessons I feel I've had have been from direct contact with people or at least close contact with people and talking about their code. (14.7a)

Some learners recognize that 'building on own experience' and 'building from others' experience' may work in partnership with each other. For example, learners call on both of these aspects of resourcing when they experiment with their lecturer's suggestions.

This category reveals a more sophisticated, independent approach to learning. The focus is on resources rather than on code and attention is placed on programming mastery rather than on finding a solution to an immediate problem.

A teaching practice expected by students associated with this category is the provision of resources (for example, libraries of source code) for learners to draw on.

Key learning strategies associated with this category are investigative research, organisation of knowledge and learning in community. These learning strategies may differ from those offered in the course.

The *approach to learning* in this category is active.

Learning to program is experienced as taking intuitive steps

Focus: Sub-conscious and intuitive skills simultaneously. External horizon: Knowledge mastery.

In this category learners experience learning to program as engaging their intuition, for example letting the sub-conscious work on the problem at hand and taking decisions based on instinct. Learners take a conscious decision to turn away from the problem, place their confidence in their own emergent mastery and provide their latent creative abilities room for expression.

... the thinking time away from the computer ... helps a lot. Because if you stay on the computer you get too close to it and you can't see the forest for the trees ... being ... confident enough ... to actually step away from the computer and allow the problem just to sit there because your subconscious will work on it and intuitively you can actually come back and get it, a different approach, see the problem from a different angle ... that helps. (15.11d)

No *teaching practice* seems to be expected by students associated with this category; the focus is off the teacher and entirely on the student.

The *key learning strategy* associated with this category is to choose to stop consciously thinking about the problem, and even leave the learning situation altogether.

The *approach to learning* in this category is active.

MASTERS STUDENTS AND THE OBJECT OF LEARNING TO PROGRAM

Our analysis has identified five different experiences of the Object of learning to program: Coding, Thinking differently, Problem-solving, Participating and Satisfying clients. These are considered to be cumulative - the latter categories embrace views held in the preceding categories.

Each of these categories is described below, giving their meaning, subcategories (if appropriate) and a dimension of variation. Details pertaining to structure are presented in the earlier description of the Outcome Space.

Programming is experienced as coding

Focus: Syntax. External horizon: The programming language.

In this category learners experience programming as sitting at the keyboard and writing code.

Programming ... is writing code ... I always think of programming ... as the actual sitting there and actually typing away, coding, etc. (17.1a)

Programming in this category relies on trial and error and successive drafting based on compiler error messages. Learners may deliberately enter code imprecisely, in order to make a start and allow an overall view of the program.

Good programming in this category is producing efficient code in a short period of time, without help from any other source.

Programming is experienced as thinking differently

Focus: Logic. External horizon: The programming language.

In this category learners experience programming as a way of thinking. This way of thinking is typically quite different from the way humans naturally think. The focus is on computer logic and being able to structure programs according to that logic. Learners who see programming from this point of view are aware of the need for systematic design.

... my algorithms tend to be very linear, one step at a time. ... it wouldn't occur to me naturally ... to look for solutions where I could use recursion or something which might be a terrifically efficient way to program something and solve a particular problem. But I don't feel as if I've ... made that change in mindset from looking at things in purely day to day linear terms - do job A then do job B - to looking at things in perhaps more... apparently complex ways ... but nevertheless very elegant and efficient. (19.2c)

Good programming in this category can be identified by the logic used in writing a program, resulting in a robust product.

Programming is experienced as problem solving

Focus: Problem to be solved. External horizon: The application environment.

In this category learners experience programming as a means to finding the solution to a problem in the wider world. This problem needs to be analysed and understood before programming can begin.

[What is programming?] Analysing a real-world problem or situation and then finding a way that computers can solve that problem. And then implementing that solution. (114.1a)

According to this way of experiencing programming, a test for a program's usefulness is whether it produces expected results.

Good programming in this category is producing a program that yields expected results efficiently.

Programming is experienced as participating

Focus: Collaboration with other programmers. External horizon: The programming community.

In this category learners experience programming as participating in the programming community. Programs need to make sense to another programmer, be easy for another programmer to read and be able to be maintained or adapted by another programmer.

... you've got to make sure that particular program is readable by any other person in the future, so that if ... modifications need to be done, I might get hit by a bus tomorrow, someone might have to fix it up for me. ... it has to be readable ... constructed well and laid out well so that people can manipulate it if it's required. (16.2a)

There is a recognition that there are standards to comply with, including ensuring you are producing a quality product that is socially responsible.

Good programming in this category is producing programs that are easy to maintain.

Programming is experienced as satisfying clients

Focus: Client satisfaction. External horizon: The marketing environment.

In this category learners experience programming as satisfying the requirements of clients. This includes meeting user needs, producing a program that is easy to use and marketing the software.

[Can you write a program that works?] It meets a business requirement and they're using it and obviously it's got output going out there and they're saying, "It works - we've processed two hundred documents a week for a year and the stats come out of the end of every week, and it monitors what's been done." ... At the end of the day, the users are out there using it and it's working ... you've got the thing

in and it's up and running. (110.1d)

In this context, programmers may successfully satisfy the budget demands of clients or produce the outcomes desired by the clients, even though the code produced is not efficient or elegant.

Good programming in this category is the creation of a product that is responsive to clients' needs. Client feedback is core to determining if a program is a good one.

DISCUSSION

Comparison with the undergraduate experience

A previous study of Introductory programming students (Bruce et al. 2004) which focussed only on the Act of learning to program found five categories, represented in the following Outcome Space (Figure 3). A comparison of these results with the Masters study reveals some similarities.

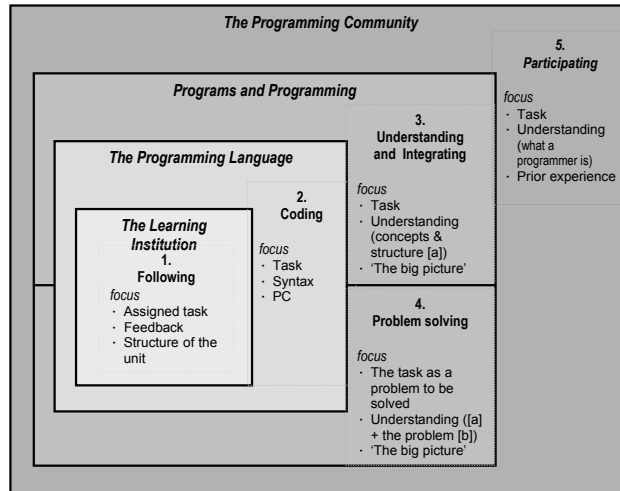


Figure 3. The Outcome Space for Introductory Students' Experiences of the Act of Learning to Program

Although the category names used to represent the perspectives are different, students from both cohorts experienced the Act of learning to program as following a formal course, as active coding and as understanding concepts (see Table 2).

Masters Act		Introductory Act	
External horizon	Internal horizon	External horizon	Internal horizon
Formal learning environment	Following	Learning institution	Following
Programming language	Acting	Programming language	Coding
Programming language; Knowledge mastery	Conceptualising; Resourcing	Programs and programming	Understanding and integrating

Table 2. Similarities between the Masters Students' Act and the Introductory Students' Act

There are also similarities between the Object of the Masters students and the Act of the Introductory students, in the categories of Problem solving and Participating which carry both the same name and similar meaning (see Table 3).

Masters Object		Introductory Act	
External horizon	Internal horizon	External horizon	Internal horizon
Application environment	Problem solving	Programs and programming	Problem solving
Programming community	Participating	Programming community	Participating

Table 3. Similarities between the Masters Students' Object and the Introductory Students' Act

The Masters students' experience of the Act of learning to program as 'Taking intuitive steps' does not seem to be represented at all in the Introductory students' views.

The similarities found between the Introductory and the Masters students accord with the phenomenographic assertion that there are a finite number of ways of seeing any one phenomenon. However, the divergence of perspective between the two groups also indicates that particular groups of students will have unique shared experiences. Continued research into the field can be expected to result in an increasingly complete view of how students of different cohorts experience programming and learning to program.

Implications for teaching practice

If we wish students to go about learning to program in some or all of the ways described in this paper, we must make students aware of the different options available to them and thus help them to expand their horizons. Similarly, if we wish them to appreciate that there are different ways of experiencing programming, we need to help them to discern the variation in experience that is possible, so they may develop a full appreciation of programming.

A foundational element in attempting to introduce students to new experiences of learning to program and programming, as indicated by Marton and Booth (1997), is an understanding of the way students have already experienced these phenomena. We offer below simple tools (Figures 4 & 5) which may be used to identify students' different experiences of learning to program and programming. Such tools may assist in helping the students discern variation, broadening their awareness of the nature of learning to program and programming.

<p style="text-align: center;">HOW DO YOU SEE LEARNING TO PROGRAM?</p> <p style="text-align: center;"><i>What statements do you agree with? (Please tick the ones that best describe how you see things.)</i></p> <p>“Learning to program is ...</p> <p>A. following a formal course.” <i>I learn to program by going to lectures and doing the required exercises.</i></p> <p>B. actively producing programs.” <i>I learn to program by writing code, for example by copying and adapting others' programs, and I hope I will understand how they work later.</i></p> <p>C. understanding the logic of programming.” <i>I learn to program by making sure I understand how a program works, for example by drawing diagrams.</i></p> <p>D. gathering resources in order to build experience.” <i>I learn to program by gathering lots of programming examples and putting them in a library for future reference.</i></p> <p>E. engaging intuitive abilities.” <i>I learn to program by stepping away from the problem and letting my intuition and subconscious work.</i></p> <p>Other: Learning to program is _____.</p>

Figure 4. Learning to program survey tool

Figure

<p style="text-align: center;">HOW DO YOU SEE PROGRAMMING?</p> <p style="text-align: center;"><i>What statements do you agree with? (Please tick the ones that best describe how you see things.)</i></p> <p>“Programming is ...</p> <p>A. sitting at a computer entering code.” <i>I program by sitting at a keyboard and typing in code.</i></p> <p>B. thinking in computer logic.” <i>I program by designing something that a computer can understand best.</i></p> <p>C. a way of finding the solution to a problem in the wider world.” <i>I program by using a computer to find the solution to a problem in the real world.</i></p> <p>D. participating in the programming community.” <i>I program by producing something that can be easily understood and maintained by other programmers.</i></p> <p>E. satisfying the requirements of clients.” <i>I program by meeting the needs of the person who asked for it.</i></p> <p>Other: Programming is _____.</p>

5.

Programming survey tool

Further, teachers of programming may be able to use the outcomes of the investigation to reflect on questions such as:

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 our courses?
2. What are the implications, for students, of certain ways of experiencing the act of learning to program?
3. How can our 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?

Ideas for utilising these questions in practice are explored in greater detail in the Introductory student study (Bruce et al. 2004).

CONCLUSION

The investigation reported here has extended our understanding of how students learn to program from the relational perspective. Previous studies have focussed on undergraduate education – the model we have developed provides a picture of the Masters students' perspectives in the Act and Object of learning to program. From the research outcomes, we have produced a comparison with undergraduate students' perspectives and have developed tools which may be used in the classroom, or other forums, to enhance learning.

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.
- Booth, S. (1992) Learning to Program: a Phenomenographic Perspective, *Goteborg Studies in Educational Sciences 89*, Acta Universitatis Gothoburgensis.
- Booth, S. (2001) 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*, Kogan Page, London.
- Bruce, C., Buckingham, L., Hynd, J., McMahon, C., Roggenkmap, M., & Stoodley, I. (2004) Ways of Experiencing the Act of Learning to Program: a Phenomenographic Study of Introductory Programming Students at University, *Journal of Information Technology Education*, 3, 143-160.
- 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 Education*, University of Aarhus, Denmark, June 24-26, 2002. URL <http://portal.acm.org/> Accessed 28 June, 2004.
- Fincher, S. (1999) What are We Doing When We Teach Programming? In *Proceedings of 29th Annual Frontiers in Education Conference: 'Designing the Future of Science and Engineering Education'*, IEEE Press.
- Gottfried, B. S. (1997) "Teaching Computer Programming Effectively Using Active Learning" in *Proceedings of the 1997 ASEE Annual Conference*, Milwaukee, WI, USA.
- Lister, R., & Leaney, J. (2003) First Year Programming: Let All the Flowers Bloom, *Fifth Australasian Computing Education Conference*, Adelaide, Australia.
- Marton, F., & Booth, S. (1997) *Learning and Awareness*, Lawrence Erlbaum, Mahwah, NJ.
- 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*, RMIT University Press, Melbourne.
- 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*, IEEE Computer Society, Austin, TX, 59 - 65.

ACKNOWLEDGEMENTS AND ETHICAL CLEARANCE

The Faculty of Information Technology, Queensland University of Technology, under the direction of the Dean, John Gough, for financially supporting the project. The research project has also been supported by programming lecturers Marlon Dumas and Yue Xu, who provided access to students and commented on the research outcomes. University ethical clearance number: QUT Project # 2941H.

COPYRIGHT

Christine Bruce, Ruth Christie, Ian Stoodley © 2004. The authors assign Griffith University a non-exclusive license to use this document for personal use provided that the article is used in full and this copyright statement is reproduced. The authors also grant a non-exclusive license to Griffith University to publish this document in full in the Conference Proceedings. Such documents may be published on the World Wide Web, CD-ROM, in printed form, and on mirror sites on the World Wide Web. Any other usage is prohibited without the express permission of the authors.

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 phenomenographic 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.

Ruth Christie is a lecturer with the School of Software Engineering and Data Communications within the Faculty of Information Technology at Queensland University of Technology. Ruth has been teaching in the area of introductory programming at the undergraduate level since the mid 1980's and at the Masters level since 1998. Ruth's major area of research and second area of teaching is the data structures and algorithms associated with computer graphics and real time animation.

Ian Stoodley is a research assistant in the Faculty of Information Technology of the Queensland University of Technology. 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.