Education and Information Technologies 4:2 (1999): 167–179

Beginners and programming: insights from second language learning and teaching

LYNNE P. BALDWIN and ROBERT D. MACREDIE
Department of Information Systems and Computing, Brunel University, Uxbridge, Middlesex UB8 3PH, UK
E-Mail: lynne.baldwin@brunel.ac.uk robert.macredie@brunel.ac.uk

This paper will consider issues that are important in the teaching and learning of programming to students in their first year of an undergraduate course in a computer science discipline. We will suggest that the current educational climate offers the opportunity to move the focus onto the learner and their experience, and that second language learning and teaching in the field of English as a Second, or Foreign, Language may be a fruitful area on which to draw. We will review a particular aspect of second language pedagogy—learner strategies—and discuss their applicability to students who are starting to learn how to program. We will consider ways in which these strategies might be useful to support learning programming at this level.

Key words: computer science education, learner centred learning, programming, higher education, English as a second or foreign language

Introduction

Teaching programming to students in the first year of their undergraduate degree course is an area of computer science education that generates a great deal of debate. Literature in this area points to the fact that those learning the complex task of programming face many problems which, as those involved in the teaching of programming, we need to help our learners overcome. Programming is a core skill and given its importance in the curriculum, as well as subsequently in the workplace, there is a perennial need for us to look for ways to help our students to learn and practise this skill more effectively. Although there has always been interest in this area, there is now a more pressing need for those involved in university-level institutions to focus their attention more closely—demands from governments are forcing such institutions to demonstrate that 'quality' teaching and learning opportunities exist. Pressure from government, together with economic developments in society, have prompted rapid and far-reaching changes in all areas of university-level education. These factors, together with the shift in mainstream education generally towards a more learner-centred approach (and thus a less teacher-centred one) means that it is time to consider how we might vary the approaches we use in the basic teaching of programming.

This paper draws on parallels between the learning and teaching of a 'natural' language to non-native speakers—that is, English as a Second, or Foreign, Language (ESL or EFL)—and considers the second language pedagogy that underpins these areas. It describes research carried out in the area of learner strategies in second language pedagogy which, we argue, may provide useful insights into how we might aid our learners in the task of learning to program.

Computer science education: areas of concern

Computer science is a relatively new discipline, and computer science education newer still. A glance through a popular publication in the field, the SIGCSE Bulletin of the American ACM (Association for Computing Machinery), will show that there are two areas of particular interest and concern to those involved in the teaching of computer science: programming generally; and CS1 students in particular. CS1 is a term used in the literature in the United States and refers to students in their first year of an undergraduate degree course in the area of computer science. As this is a usefully brief term, CS1 will be used in the rest of this paper to refer to students in their first year of an undergraduate degree course in the discipline of computer science. These two areas, programming and CS1 students, are inextricably linked; programming is taught, almost without exception, in the first and subsequent semesters of the first year of an undergraduate degree course at university level. As such teaching programming to CS1 students will form the focus of this paper.

There are many general issues that concern programming and CS1 students. It is, for example, argued that the focus on programming at such a critical time, and for so long, has unfortunate consequences [10, 11, 34]. Another concern is that CS1 students mistakenly believe that computer science *is* programming with a few ‘extras’, and that programming is somehow fundamentally uninspiring [11, 21, 33]. A far reaching problem that is often aired is that students are lacking in the necessary ability to apply computing in the real world of work [6, 10, 34]. There is also discussion as to whether CS1 students should learn, for example, a general-purpose programming language such as C++, together with a supporting professional programming environment, and work with a set of problems from the area of number and symbol processing or whether to use a mini-language [1]. Given the diversity of these concerns and the competing debate that surrounds them, it is hardly surprising that there is little agreement as to which approach, or approaches, we might adopt in order to assist our learners in developing programming skills. This is despite the content of the academic curriculum in computer science being dynamic [3, 7] and open to suggested improvements.

Although few would disagree that programming is a complex and difficult cognitive task [9, 14–17, 25], literature on both the learning, and teaching, of programming employs dramatic language—references to learning to program as a ‘daunting challenge” are common. Further examples from the literature suggest that those new to programming can face “appalling difficulties [trying to wrest] a solution out of their venturing” and can “feel unsure about what they are doing [and] harbor fears” [21]. Working with a computer can, it seems, also become “a threat to self-esteem and one’s standing with peers and teachers”. This will be familiar to many teachers of programming, who find their students reluctant to persevere with programming.

Programming is a core skill and one which our students should enjoy, yet learning to program seems fraught with difficulties. This may not be helped by the relative infancy of programming language pedagogy, but there have been other factors which may have limited developments in teaching programming to CS1 students.

The quality of Computer Science education at university level

One topical factor is the provision of consistently high-quality teaching of programming at university level. This has to be seen against the general education climate today, where there has been a major shift in focus from teacher-centred, teacher-led activity, following criticism that teachers have fostered what is termed ‘learned helplessness’. Today, teachers are being asked to look for ways of empowering students, and also to look for ways of helping students to empower each other as well. Learners are now viewed as active participants in the learning process, constructors of meaning not merely receivers of information. There is thus a growing demand for the activities in the classroom to become more learner-centred and for students to take more responsibility for their own learning. There has been a move away from describing teachers as ‘teachers’; it is becoming more common to see them referred to in the literature as facilitators, *assisting* learners in the learning process.

These changes have also found their way into Higher Education. There are pressures on all those involved in teaching to look more closely at the teaching, and learning, taking place in their institutions. These changes in education in general, and Higher Education in particular, mean that it is now a good time to reassess the current approaches that we adopt in order to assist our learners in the learning of programming. A recent report [26] published in the UK confirms what is doubtless similar elsewhere, namely that the percentage of excellent providers (11%) is the lowest of the eight subject areas so far assessed. Assessors were critical, it seems, of not only limited resources, but also “pedestrian teaching, reduced opportunities for independent learning and limited staff development”, despite well qualified and enthusiastic academic and support staff. The Report offered no explanations for the poor results other than the ‘reasons for this are complex’, but noted that computer science and related subjects are relatively new; a comment echoed in research carried out in the US [27]. Prospective students are not, however, deterred; the number of applications has been on the increase for several years. In the recruitment period for the intake of students for the academic year 1997/98, for example, applications were up 20% on the previous year and this is set to happen again in 1998/99.

There are also changes taking place in the wider context as well. With a large amount of this money being spent on state-funded education, those involved in Higher Education are now being asked to better account for how that money is spent and to demonstrate the ‘quality’ of the service provided to students. In response to this, universities are now looking at the quality of the teaching, and learning, within their institutions. Another recent change is the number of students deciding to continue their education beyond the normal school-leaving age. Until a few years ago, only around 8% or so of school leavers in the UK went on to study at university level. Today, that figure is nearer 30%. Another change can be seen by looking at the learners themselves. Around 50% of those learners coming into the two undergraduate degree courses (BSc in Computer Science and BSc in Information Systems) in our own institution, a ‘traditional’ university, do not have the ‘traditional’ qualifications that students used to have.

The above suggests a changing educational climate focusing on quality and a changing learner-base. If we are to improve the teaching and learning of programming to CS1 students, we might like to consider who those learners actually are.

CS1 learners

To shift our attention away from teaching and towards learning, it is necessary to look more closely at our learners. The literature in the SIGCSE Bulletins seems to divide CS1 students into two distinct groups: majors and non-majors. The former are students who are studying for an undergraduate degree and whose main subject is in the area of computer science. The latter are students who are studying for an undergraduate degree but whose main subject is something other than computer science but which includes some element of computer science. These classifications reflect the structure and terminology used in the US and are referred to differently in, for example, the UK. However, regardless of what terminology is used to describe such students, there is an implicit belief or understanding that there is some kind of difference between the two groups. The literature appears to suggest that these different groups of students have different levels of motivation. The groups may also reflect how far they are going to go with programming. If these *are* different kinds of students, then how, exactly, are they different? Or indeed, similar? If we are expecting different levels of knowledge and competence in programming from students studying for different degrees in the wide field of computer science, what different levels of 'expertise' do we expect them to have at the end of their studies and how can we define, or measure, this?

For CS1 students a broad expectation is that, at least from an educational perspective, the learners are 'novice' programmers. This seems to be the meaning applied in SIGCSE Bulletins, where learners in a CS1 level class, who are, by implication, new or relatively new to programming tend to be termed 'novices'. There is, however, some variation in other literature as to what a 'novice' is. A dictionary definition of 'novice' is 'a person who is new to or inexperienced in a certain task, situation etc.'. Mayer [17] defines them as "users who have had little or no previous experience with computers, who do not intend to become professional programmers, and who thus lack specific knowledge of computer programming" (p. 131). Definitions such as Mayer's raise questions which should be considered in framing our approach to teaching programming. These include whether CS1 students generally (or individually) intend to become 'professional programmers'; whether prior knowledge is a factor to be taken into account when teaching programming to CS1 students; what kind of prior knowledge is relevant and how; and whether age is an important factor [21].

These questions bring the focus firmly onto the characteristics of the learner, a trend which we have already noted as have increasing importance attached to it. It is the need to focus on the *learner*, and the *learning*, rather than the *teacher* and the *teaching*, that we need to look at much more closely if we are to understand more about—and effectively support and enable—their learning of programming.

Beginners and the learning experience

Our view of how we should approach CS1 programming mirrors those of EFL practitioners: "learners are entitled to an educational experience that is motivating, responsive to their needs, and effective in providing them with the requisite skills for life and jobs" [4]. Indeed, in common with Robertson and Lee [23], we see EFL as a field which may offer insights into how we might improve the learning experience for CS1 student learning programming.

A glance through the literature on the teaching and learning of English as a Foreign, or Second, Language (EFL or ESL) to adult students whose first language is not English and who do not live in an English-speaking environment, reveals that the appalling difficulties, threats or fears that lie in wait for novice CS1 programmers do not lie in wait for the 'novice English speaker'. New *teachers* might feel afraid of going into their first few classes, where students are often paying a sizeable proportion of their hard-earned income (50% is not unusual) on English classes and the weight of responsibility can seem daunting, but in our long experience of teaching EFL (one of us has taught EFL extensively), *students* invariably report that they have learnt more than they ever thought possible and that they have found the experience of learning very enjoyable, despite having to come along to lessons in the evening, and at the end of a long day of work.

As mentioned above, programming is a complex and difficult task; but arguably no more difficult than that faced by many EFL students. Such students may have only a basic level of school education and may be unable to write in their own, very different, script (Arabic, for example), yet they have to learn to speak, read and write in English, supported by an EFL teacher who does not speak the language of the learners. These achievements suggest that the second language pedagogy used in EFL may be an important field to look at more closely.

Insights from second language pedagogy

This suggestion is not new; it has previously been argued that the insights provided in the subject area of second language pedagogy could be of use to those involved in the area of 'second language' pedagogy in computer science [23]. Much of the literature in computer science is, at first glance, immediately familiar to the applied linguist—syntax, semantics, translators—and a glance at the job advertisements often mirrors that of ability in a natural language: for example, "Ideally you'll be fluent in HTML, Perl and Java" [8].

Research in the field of second language learning since the early 1970s has moved away from looking at methods of teaching to looking at individual learners and how learner characteristics affect how they learn a second language. With the current emphasis on the need to adopt more learner-centred approaches in the teaching of computer science, the work done in this area could provide us with some useful insights.

Also from the early 1970s, the work of cognitive scientists has given us greater insights into the nature of the workings of the mind. If we, as educators, or facilitators, are to develop a greater understanding of learning and teaching, then trying to understand *how* people learn programming or any other subject or skill is vital. Cognitive scientists have started to look closely at not only formal logical reasoning and problem solving, but at everything that goes on in the mind between input and output. However, like computer science, this field is relatively new. It is also problematic in that it is particularly difficult to observe, measure or test what is going on in the 'black box' of the human brain. Cognitive science looks at, among others, perception, memory, learning, inference and concept formation.

One view of human cognition views people as processors of information, with our sense perceptors processing incoming data in some way. As this data comes in, we attend to and process it in some way, identify it and move it into the short-term or working memory; thus transforming 'input' into 'intake'. We then carry out a number of mental operations and the resulting changed or modified 'product' is somehow stored in long-term memory for later use. The mental operations that we carry out to make sense of and store incoming information are referred to as processes, and the changes we make to these are referred to as organisations of knowledge, or knowledge structures. How we manipulate the incoming information, and later retrieve what has been stored, are referred to as cognitive strategies.

Cognitive strategies and learner strategies

Learners also employ cognitive strategies to support their learning. As such, research on learner strategies in second language learning can be viewed then as part of the general area of research on mental processes and structures that make up the field of cognitive science. However, the term 'learner strategies' is, according to Wenden [31], ill-defined and literature on the subject refers to them interchangeably as 'techniques', 'tactics', 'potentially conscious plans', 'consciously employed operations', 'learning skills', 'basic skills', 'functional skills', 'cognitive abilities', 'language processing strategies', 'problem-solving procedures', indicating that there is "little consensus in the literature concerning either the definition or the identification of language learning strategies" (p. 7).

Researchers have come to recognise two major kinds of learning strategies: metacognitive and cognitive strategies. Although it is difficult to separate the two, the view of O'Malley *et al.* [19] is often quoted. They say that metacognitive strategies refer to knowledge about cognitive processes, and also the regulation of cognition or executive control or self-management through such processes as planning, monitoring and evaluating. Metacognitive strategies are those that learners use to oversee, regulate or self-direct language learning; noted by Wenden [28–30] in her research into how second language learners regulate their learning by planning, monitoring and evaluating. In contrast, cognitive strategies refer to the steps or operations used in learning or problem-solving that require direct analysis, transformation, or synthesis of learning materials. Cognition consists of those strategies through which an individual obtains knowledge or conceptual understanding.

It may therefore be useful to use the term 'learner strategies' in the way that Wenden suggests in one of the most well-known books in second language learning and teaching, [31; pp. 6–7] namely that learner strategies are:

- **language learning behaviours that learners actually engage in to learn and regulate their learning of a language**. These language learning behaviours have been termed strategies and in order to find out what they are, it is necessary to observe and record what learners do, or say they are doing, while working on a task.
- **what learners know about the strategies they use, that is, their strategic knowledge**. By asking the learners to report on specific or general aspects of their language learning, learners' reports may reveal which strategy, or strategies, they have actually used in a particular type of task. (This, though, is no guarantee that they actually *did* use them; they may have assumed that this is what they did, or thought they should have done).
- **what learners know about aspects of their language learning other than the strategies they use**. This includes personal factors that help language learning, or general principles that students follow in order to learn a language successfully, what is easy or difficult about learning a specific language, and how well or poorly they can use the language. What learners know about these aspects is assumed to influence their choice of strategy.

What learner strategies exclude in the definition above are variables which may provide a background to learning success, which Rubin [24; p. 19] clarify as:

- psychological characteristics (risk-taking, field dependence and empathy are examples)
- affective variables (liking or disliking the teacher is an example)
- social style (outgoingness or timidity for example)

Learner strategies in Computer Science education

Research in the field of second language learning has been concerned with trying to answer the following questions; which, by changing the word 'second language' to 'programming', mirror the ones we need to know to find out how students learn to program [31; p. 6]:

- what do learners *do* to learn programming?
- how do they *manage* or *self-direct* these efforts?
- what do they *know* about *which* aspects of their learning process?
- how can their learning skills be refined and developed?

Despite what has already been uncovered in the field of second language learning, there are still many unanswered questions about these strategies. What is not yet clear, according to Wenden [31] is whether or not these strategies are:

- general characteristics or specific techniques or actions
- conscious, unconscious or sometimes one or the other
- learned behaviours or part of our genetic mental makeup
- if they are learned, under conscious control at all times or able to become automatic

- tied to specific learning content and tasks or more general in their use

There is also the question as to what might trigger learners to use these strategies [31; p. 7]. Wenden lists six criteria that, generally, appear to characterise the language learning behaviours that have been referred to as strategies [31; pp. 7–8]:

- Strategies refer to specific actions or techniques (such as decomposition). They do not describe a learner's general approach (for example whether learners are risk-takers or not).
- Some of these actions will be observable (for example, asking questions) and others will not be observable (for example when students make a mental comparison).
- Strategies are problem-oriented. Students need to learn, and, in the language of cognitive psychology, to facilitate the acquisition, storage, retrieval or use of information.
- There are language learning behaviours that contribute directly to learning (for example practice strategies), there are also language learning behaviours that contribute indirectly.
- Strategies may sometimes be consciously deployed, or can become automatic, thus remain below consciousness or potentially conscious.
- Strategies are behaviours that are able to be changed, so students can modify or reject 'old' ones, and unfamiliar ones can be learned.

Research by Naiman *et al.* [18] focused on personality traits, cognitive styles and strategies that were critical to successful language learning. According to the list of five general strategies and related techniques, good language learners:

- actively involve themselves in the language learning process by identifying and seeking preferred learning environments and exploring them
- develop an awareness of language as a system
- develop an awareness of language as a means of communication, an interaction
- accept and cope with the affective demands of the second language
- extend and revise the second language system by inferencing and monitoring

Programming competence or proficiency ranges from zero to expert, and is different for different learners at different stages of learning. The stages between these competences give insights into how different learners tackle the learning of programming. There have been few attempts made to identify the characteristics, strategies and techniques of successful programmers through actual observation or interview. We are currently devising and running a study that will provide necessary information about successful and less successful learning of programming. We expect that the interviews that comprise part of the study will "confirm, modify or disprove existing hypotheses and hunches" [18; p. 9] about learning programming.

The first part of the study will involve data collection via a questionnaire. As we know very little about the nature of the preconceived ideas that learners have about programming, which beliefs that specific types of students hold or how these beliefs affect their learning of programming, this questionnaire will seek to report on the views of the learners and those who teach, on programming. This study replicates the work done by Horwitz [12] in foreign or second language learning. In her 'BALLI (Beliefs About Language Learning Inventory), she sought to assess student opinions on a variety of issues and controversies related to language learning. We have taken this Inventory, and have made only minor modifications in the text to reflect that it is programming, rather than natural language learning that is the focus of our preliminary research agenda. This instrument will be used, as it was in the original study by Horwitz, to assess the beliefs of students and teachers about learning language, or in our case, learning programming. Responses to this will assist us in coming to understand three areas. First, an understanding the nature of student beliefs and the impact of those beliefs on strategies used in learning programming. Second, assist us in understanding why teachers choose particular teaching practices. Finally, it will help us determine where the beliefs of those teaching programming, and our students, might not be in agreement.

The second part of the study will comprise an interview questionnaire consisting of a directed and a semi-directed part. This part of the study will involve asking the interviewees, who have been rated as highly proficient, to describe their learning of programming experiences. An analysis of the results will identify the strategies, and describe the overall approach to the learning of programming which appear to be essential to success. It will also enable us to gather information about other factors which influence success in learning programming. Our hypothesis is that an analysis of the five major strategies identified by the good learners in the Naiman *et al.* [18] study above, will reinforce the value of aspects of second language learning pedagogy to the study of those learning computer programming languages.

As a way into thinking about these strategies, the remainder of this section provides introductory discussion around the contextualisation of Naiman et al's strategies within the learning of computer programming. We will present each of the original strategies before discussing a possible reinterpretation of them with respect to programming.

Strategy 1: actively involve themselves in the language learning process by identifying and seeking preferred learning environments and exploring them

Clearly the opportunities normally associated with learning to program will be different from those associated with learning a natural second language. However, the emphasis this strategy places on active choices on the part of the learner around the appropriateness of particular learning environments would, we contend, still stand. The value a learner may find within formal teaching sessions, laboratory sessions with expert supervision, informal practical classes with peer support and interaction, and through learning with non-practical application of programming concepts, will vary. It is important that learners are encouraged to make informed and balanced choices about the benefits of combinations of learning environments. These will clearly be linked to the current requirements of the needs of the learner in relation to their current programming expertise. Reflection on the value of these different learning environments, in isolation and in combination, is likely to form a key part of the learner's development both with respect to programming and to their wider engagement with their academic programme of study.

Strategy 2: develop an awareness of language as a system

The parsimoniousness of this strategy belies its extreme complexity yet centrality to our consideration of learning to program. It is vital that we engender an understanding of programming not as a collection of language statements but rather as an holistic view of problem representation. We have to encourage learners to see the programming language as a systemic vehicle through which they can appropriately codify complex problems. We believe that too much emphasis is placed in teaching programming on language constructs in isolation. Whilst this might support the immediate learning of syntactic structures, it tends to distract learners from developing a view of programs as encapsulated problem representations. This may well be because syntactic knowledge is simple, and easy to correct. High level issues in programming which predominantly revolve around levels of semantic expression represent the extreme challenges for learners and teachers alike, and it is here, we would argue, that efforts should be targeted.

Strategy 3: develop an awareness of language as a means of communication, an interaction

This strategy is interrelated with strategy 2 since learners should be concerned with communicating their abstract notions of a problem with which they are concerned through a programming language. The outcome of this is the execution of the program on a computer platform. Such 'running' of a program (and the associated output) is ultimately another level of problem communication. Within teaching programming languages, the strategy suggests that it is important for us as teachers to situate programming within the context of computer operation. It is only through understanding the possibilities and limitations of computers that learners will develop a coherent and broad appreciation of what can be achieved through computer programs. In our experience, this can be sadly lacking in those in the early stages of learning how to program, and can lead to extreme frustration when writing programs. Learners often find it difficult to develop strong and appropriate understandings of what they can achieve through programming because they have not developed an appropriate model of the relationship between programming language, program and computer architecture.

Strategy 4: accept and cope with the affective demands of the second language

Anyone who has been involved in the teaching of programming will simply be able to explain and recount the frustrations and disappointments exhibited by those learning programming. In one sense then, the teacher has a role as expectation manager. Learners must be encouraged to see programming as a challenging and rewarding journey, and to accept that it will at times be difficult and problematic. The ups and downs of learning to program are natural and learners must be sensitive to this, and learn to deal with this as part of the normal process of learning to program. Part of this rollercoaster journey is developing an awareness of themselves as active and engaged learners grappling with complex material. We would argue that it is only through developing personally constructed understandings of issues in programming that learners will formulate deep and meaningful understandings of the discipline.

Strategy 5: extend and revise the second language system by inferencing and monitoring

The development of learners to competent programmers and beyond depends greatly on the ability of learners to infer meaning in new instances of programs from their existing knowledge base. Not only are new examples of programs rife in teaching (for example, one only has to note their prevalence in programming textbooks) but being able to understand existing code is a key task in maintaining computer programs (a 'profession' with which many programmers become familiar when they leave programming as a purely academic discipline). Monitoring can be seen as closing the inference loop. Learners have to re-examine their knowledge base in the light of new programming example and adjust and develop as appropriate. It may be, for example, that more elegant ways of approaching a particular recurrent task are encountered, and learners have develop a sophisticated awareness of the value in reconstructing their knowledge base to incorporate these new approaches.

The teaching of strategies to CS1 learners

As our previous contextualisation and discussion suggests, there is much work to be done in moving from our unsubstantiated opinion to a solid and repeatable research base. The definition and completion of such a research programme is ongoing. However, our work will not end there. If, after this extensive research, we come to learn more about the strategies that beginners employ when learning to program, there is then the question as to whether it is possible to train students in the use of these strategies and how this can be done. Cognitive learning theorists have found mounting evidence to suggest that "an ideal training package would consist of both practice in the use of task-appropriate strategies, instruction concerning the significance of those activities, and instruction concerning the monitoring and control of strategy use" [2]. Cognitive demands, however, vary according to the level of the students and different instructional practices are needed: introductory level students benefit from direct instruction, whereas more advanced students perform better with less direct guidance and more opportunities for autonomy [13]. Wesche [32] found that her students used many of the strategies mentioned above and added that those who improved most rapidly were students who exhibited a greater variety and quantity of learning behaviours. She also observed that many of the learning behaviours occurred together, concluding that it may be complexes of behaviours rather than specific ones which characterise different kinds of learners. What is also not yet clear is what makes a particular student choose, whether consciously or unconsciously, a particular strategy at a particular time for a particular activity, although Oxford [20] summarised the following possible variables on learner strategy selection: motivation; gender; cultural background; type of task; learning style; age and stage of learning. Information about these, she added, may, or may not, be useful in predicting strategy use.

Wenden [31], however, cautions that those who encourage the teaching of strategies to students are often over-concerned with techniques. She stresses that such self-instructional strategies have to be accompanied by a change in the way they are viewed by the learner. If this is not the case, attempts to train learners in strategies may well meet resistance, leading to their ultimate failure. Wenden succinctly sets out an overview of what we too must consider in adopting 'learner strategies' in teaching programming to CS1 students:

> "together with the training in the use of strategies, the fostering of learner autonomy will require that learners become critically reflective of the conceptual context of their learning. They must be led to clarify, refine and expand their views [...] they should also understand the purpose for which they need to learn [...] learners will also need to learn to believe in their potential to learn and to manage their learning and to be willing to assume a more responsible role in the process" [31; p. 12].

Discussion

A key theme from our discussions concerns how our beginner, CS1 students learn programming. As facilitators in the learning process, we need to consider how we can help to prepare our students for the learning experience. The challenge for us is to find out how, and how efficiently, learners learn programming. We feel that this can be supported by a systematic examination of learners' perceptions of their learning. We need to define the strategies which good, and poor, beginner programmers use and place them within a typology. We feel that discovering which strategies work best for which particular kinds of learners, and the subsequent drawing up of an approach for teaching such strategy use will help improve the learning experience and will move research in the area forward.

Fanselow [5] sums up by saying that the exploration of learning strategies and descriptions of the results of these explorations will not tell us *the* way to go. He says that exploration of this 'new territory' should be seen as the classic tales of journeys, namely that "the journey itself is more important than the destination". This is surely true of our students' 'journey' through university and by looking at what our students have to say about their learning, we may ourselves learn: to take what they say seriously.

References

Brusilovsky P., Calabrese E., Hvorecky Y., Kouchnirenko A. and Miller P. (1997) Mini-languages: a way to learn programming principles. *Education and Information Technologies* **2**(1) 65–83.

Brown A. L. and Palinscar A. S. (1982) Inducing strategic learning from texts by means of informed self-control training. *Topics in Learning and Learning Disabilities* **2**, 1–17. Special issue on metacognition and learning disabilities.

Computing curricula 1991. (1991) *Communications of the ACM* **34**(6) 68–84.

Duff T. (1997) Comment: raise the standard! *ELT Journal* **51**(3) 269.

Fanselow, J. F. (1987) Forward. In Wenden A and Rubin J. (eds), *Learner Strategies in Language Learning*. Prentice-Hall International (UK) Ltd, Hemel Hempstead, UK, pp. x.

Gersting J. L. and Young F. H. (1997) Content+experiences=curriculum. *SIGCSE Bulletin* **29**(1) 325–329.

Gibbs N. E. and Tucker A. B. (1986) Model curriculum for a liberal arts degree in computer science. *Communications of the ACM* **29**(3) 202–210.

Advertisement from the Guardian newspaper. 16 August 1997. http://www.venus.co.uk.

Guindon R. (1990) Designing the design process: exploiting opportunistic thoughts. *Human Computer Interaction* **5**, 305–344.

Haddad H., Tesser H.. and Wartik S. (1997) Megaprogramming education. *SIGCSE Bulletin* **29**(1) 282–286.

Holmes G. and Smith T. C. (1997) Adding some spice to CS1 curricula. *SIGCSE Bulletin* **29**(1) 204–208.

Horwitz E. K. (1987) Student beliefs about language learning. In Wenden A. and Rubin J. (eds), *Learner Strategies in Language Learning*, Prentice-Hall International (UK) Ltd, Hemel Hempstead, UK, pp. 119–129.

Husic F. T., Linn M. C., and Sloane K. D. (1989) Adapting instruction to the cognitive demands of learning to program. *Journal of Educational Psychology* **81**(4), 570–583.

Jeffries R. A., Turner P., Polson G. and Atwood M.E. (1981) The processes involved in designing software. In Anderson J. R. (ed), *Cognitive Skills and their Acquisition*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 255–283.

Kim J. and Lerch F. J. (1997) Why is programming (sometimes) so difficult? Programming as scientific discovery in multiple problem spaces. *Information Systems Research* **8**(1) 25–50.

Letovsky S. (1986) Cognitive processes in program comprehension. In Soloway E. and Iyengar S. (eds), *Empirical Studies of Programmers*, Ablex Publishing, Norwood NJ, pp. 5–79.

Mayer R. E. (1989) The psychology of how novices learn computer programming. In Soloway E. and Spohrer J. C. (eds), *Studying the Novice Programmer*, Lawrence Erlbaum Associates, Hillsdale NJ, pp. 129–159.

Naiman N., Frohlich M., Stern H. H. and Todesco A. (1978) *The Good Language Learner*, Ontario Institute for Studies in Education, Toronto, Ontario, Canada.

O'Malley J. M., Russo R. P., Chamot A. U., Stewner-Manzanares G. and Kupper G. (1983) *A Study of Learning Strategies for Acquiring Skills in Speaking and Understanding English as a Second Language: Uses of Learning Strategies for Different Language Activities by Students at Different Language Proficiency Levels*, InterAmerica Research Associates, Rosslyn, Va.

Oxford R. (1993) Research on second language learning strategies. *Annual Review of Applied Linguistics* **13**, 175–187.

Perkins D. N., Hancock C., Hobbs R., Martin F. and Simmons R. (1989) Conditions of learning in novice programmers. In Soloway E. and Spohrer J. C. (eds), *Studying the Novice Programmer*, Lawrence Erlbaum Associates, Hillsdale NJ.

Roberge J. and Carlson C. R. (1997) Broadening the computer curriculum. *SIGCSE Bulletin* **29**(1) 320–324.

Robertson S. A. and Lee M. P. (1995) The application of second natural language acquisition pedagogy to the teaching of programming languages—a research agenda. *SIGCSE Bulletin* **27**(4) 9–12.

Rubin, J. (1987) Learner strategies: theoretical assumptions, research history and typology. In Wenden A. and Rubin J. (eds), *Learner Strategies in Language Learning*, Prentice-Hall International (UK) Ltd, Hemel Hempstead, UK, pp. 15–30.

Simon H. A. (1973) The structure of ill-structured problems. *Artificial Intelligence* **4**, 181–201.

Subject Overview Report QO 8/95: Quality Assessment of Computer Science 1994. (1994) Higher Education Funding Council for England.

Taylor H. G. (1997) The evolution of standards for accrediting computer science teacher preparation programs. *SIGCSE Bulletin* **29**(1) 67–71.

Wenden A. (1982) The Processes of Self-Directed Learning: A Study of Adult Language Learners. Unpublished doctoral dissertation, Teachers College, Columbia University.

Wenden A. (1986a) What do language learners know about their language learning? A second look at retrospective accounts. *Applied Linguistics* **7**, 186–201.

Wenden A. (1986b) Helping L2 learners think about learning. *English Language Teaching Journal* **40**, 3–12.

Wenden A. (1987) Conceptual background and utility. In Wenden A. and Rubin J. (eds), *Learner Strategies in Language Learning*, Prentice-Hall International (UK) Ltd, Hemel Hempstead, UK, pp. 3–13.

Wesche M. B. (1975) The Good Adult Language Learner: A Study of Learning Strategies and Personality Factors in an Intensive Course. Unpublished doctoral dissertation, University of Ontario, Canada.

Woodman M., Law A., Holland S. and Griffiths R. (1997) The object shop—using CD-ROM multimedia to introduce object concepts. *SIGCSE Bulletin* **29**(1) 345–349.

Zachary J. L. (1997) The Gestalt of scientific programming: problem, model, method, implementation, assessment. *SIGCSE Bulletin* **29**(1) 238–242.