Lessons Learnt from a Decade of Structured Support for Novice Programmers

[1]Meriel Huggard, [2]Ciarán Mc Goldrick
Trinity College Dublin, Dublin, Ireland, Meriel.Huggard@tcd.ie [1];
Trinity College Dublin, Dublin, Ireland, Ciaran.McGoldrick@tcd.ie [2]

Abstract

While some Engineering students find learning how to program rewarding, others struggle to develop any appreciation of, or proficiency in, fundamental programming constructs. Indeed, students typically view programming modules as necessary pre-requisites for progression beyond the freshman years, rather than as a means to help them develop their critical thinking and problem solving skills.

Over the past decade the Programming Support Centre in Trinity College has sought to provide a positive, supportive atmosphere where students can voluntarily seek one-to-one or small group assistance with challenges they face while learning to program. During that time the Centre's service offering has evolved to meet changes in students' needs and to appraise faculty members of novice programmers perceptions of the challenges they face when attempting a programming assignment.

We assess the Centre's impact on student learning and detail how the natural tensions that exist between students, the Centre staff and faculty members have been mitigated. We discuss the qualitative and quantitative metrics used to assess the performance of the Centre over the past decade and suggest ways in which it may evolve in the course of the next ten years.

1. Introduction

Prior to beginning their undergraduate studies many students are already negatively disposed towards learning to program. This is true, not only of those studying courses with a significant technology component (such as Engineering), but also of those who intend to major in Computer Science [1]. It has been suggested that those who find programming difficult simply have "no aptitude" for learning to program [2], and a wide variety of tests that seek to quantify a student's aptitude for programming have been developed over the past three decades [3,4,5].

It is more than fifteen years since Winslow [6] argued that the literature had not informed either curriculum design or the creation of textbooks. His views were echoed in a more recent survey of the literature on teaching introductory programming [7] which concluded that active research in the area "had limited effect on classroom practice". Winslow's conclusion that, in order to create competent computer programmers, the key is "practice, practice, practice – starting with simple facts and problems and working up to more and more complicated facts, strategies and problems" [7] was echoed in Stamouli's 2009 Ph.D. thesis [8] ("Learning Object-Oriented Programming from the Students' Perspective") where she concluded that students with programming experience perceived programming constructs in a "more complete and advanced way". Thus, in order for students to become competent programmers, they need to regularly engage with practice problems. However when faced with a programming problem many students don't know where to begin their solution and consequently become demotivated.

Those teaching novice programmers often have to make the difficult choice of which language will provide the best instructional metaphor for the student cohort. This choice has grown more difficult as the number of candidate programming languages has grown rapidly over the past two decades. The factors influencing this fraught decision include the motivation and skillset of the faculty member delivering the module, the technical nature of the language, its perceived commercial relevance and its pedagogical underpinnings [7]. The most popular programming languages being taught today are C, C++ and Java [9], however there is much debate on their appropriateness for those learning to program for the first time [10,11]. An inherent tension arises between industry/faculty requirements and sound pedagogical principles in this choice of first programming language. As a result there are many students learning programming for the first time through languages that they, their instructors, the academic community and, indeed, industry feel are unsuited for the purpose.

Another factor which impacts on student learning is the very large classes that can arise when a subject is a compulsory part of the curriculum. For example, in Trinity College Dublin there are three large student cohorts studying programming: two groupings of typically 200 students and another of approximately 130 students. These groups are learning either Java or C++. Such large groups make it very difficult for the instructor to accurately gauge each student's progress or to target additional support to students who are struggling to keep up with the course content.

Novice programmers' experience of their learning environment highlights the importance of practical tutorial and laboratory classes [8]. Faculty members must be well supported by any teaching assistants or laboratory tutors assigned to their courses. These assistants are often drawn from a pool of local postgraduate students and might not have the programming skills necessary to allow them help novice programmers in a meaningful way. It can be very difficult for the faculty member to train tutorial assistants in the skills required while simultaneously managing very large class groupings. Moreover the motivation of the assistants themselves may not be aligned with that of the instructor (or the students) in that they may have undertaken the role for little reason other than the salary involved

In summary, many students find themselves in large classes taking programming courses they feel are largely irrelevant to their future careers [13]. This situation is exacerbated when they learn programming languages that may be inappropriate for novice programmers and when they receive less than wholly effective support in their tutorial and laboratory classes. In this paper we explore one targeted initiative that provides support to such students in their efforts to learn how to program.

2. Supporting Novice Programmers

About fifteen years ago many U.K. universities became very concerned with the decline in their students familiarity with, and mastery of, basic mathematical skills - particularly on courses with a strong mathematical component such as Mathematics, Computing and Engineering [14]. One way in which this concern was addressed was though the provision of Mathematics Support Centres. These Centres provide additional resources and tuition to students experiencing difficulties with mathematics. Many Universities use diagnostic tests to help identify freshman students who are likely to encounter difficulties with mathematics and these students, in particular, are encouraged to make use of the support services available to them. While all students taking mathematics courses in an institution can avail of the services available at the support centre, attendance is not mandatory. The Mathematics Support Centres proved very successful [14, 15] and the number of centres grew rapidly to the point where almost all Universities in the UK and Ireland offer some form of structured support to students undertaking mathematics modules.

The Trinity College Programming Support Centre (PSC) was established a decade ago and was conceived as an evolution of the mathematics support paradigm for addressing the needs of novice programmers. Its objective is to "motivate students to improve their programming skills and enhance their personal academic success" [16]. The Centre was not intended as a

replacement for traditional lectures, laboratory classes and tutorials; rather it aims to complement these structures by helping students develop into more competent, independent, self-assured programmers. Students attending the Centre are encouraged to bring their problems and solution attempts with them and they will not receive support and advice unless they have made demonstrable efforts to get to grips with the material themselves.

The Centre's aims [16] are to:

- Ease the transition of students into programming courses
- Motivate students to adopt a positive attitude towards programming and their studies in general
- Develop the students' programming and critical thinking skills in a supportive and practical fashion.

2.1 Infrastructure and Resources

The Programming Support Centre is a dedicated facility (see Figure 1 below) that is structured in a less formal layout than that of a laboratory or a lecture theatre. It is equipped with a number of workstations, screens, a printer and a scanner. There is also a small library, seating, a dedicated study area and a whiteboard. However the key resource the Centre provides is its staff. Each staff member works with students on a one-to-one basis, or in small groups of two or three. The staff member schedule is listed on the Centre's webpage so students who wish to work with the same tutor on a return visit to the Centre can see when they are available. The staff members are experienced programmers who are encouraged to keep up-to-date with the latest pedagogical developments in the teaching of programming. They are also responsible for collating data on the Centre's performance and are encouraged to suggest ways in which it should evolve to meet the changing needs of students.

Figure 1: TCD Programming Support Centre

In its first few years of operation the Centre was open for approximately sixteen hours each week (during term time) and operated as a drop-in service only. The opening time slots were chosen based on identified gaps in the students' timetables (often around lunch time and after scheduled classes in the afternoon). However it was found that the students often only made use of the centre when they had programming assignment deadlines - so that in one week a

given timeslot could see two or three student visits to the Centre while at the same time the following week upwards of sixty students could be found waiting for the Centre to open.

Initially intensive efforts were made to publicise the Centre and its services to students. These included an introduction to the Centre during freshman orientation days, posters in communal student areas, clear signposting to the Centre, a dedicated web site detailing the Centre's location and opening hours and the services provided, and regular emails to class mailing lists encouraging students to make use of the "free" service offered to them. Faculty members were also asked to regularly remind students that the Centre was available to them if they were finding programming challenging.

Despite these efforts initial surveys at the end of the first year of operation showed that student awareness of the Centre was disappointingly low (44% of students said they were unaware of the Centre's existence). However it was found that the simple process of asking the students to fill out a voluntary five minute questionnaire on the Centre helped to raise awareness significantly as there was a notable increase in footfall in the weeks following the survey. This tactic was adopted in subsequent years and students were surveyed during the fourth week of term to help increase awareness. It was also found that publicity needed to extend to all years of the undergraduate degree programs, rather than to first year students only.

2.2 Staff Recruitment and Development

Staff are drawn from the postgraduate community in the University. They are ideally recruited during their first year of study, so that they remain on the staff for three or four years. Staff recruitment takes place on an annual basis to ensure continuity from one year to the next. There are typically ten tutors associated with the centre, as well as a Centre manager who is in charge of the day-to-day running of the centre. Initially staff training was provided by the manager of a Mathematics Support Centre in a neighbouring University. However in subsequent years training was carried out by the Centre manager and by academics associated with the Centre. This training addresses issues such as what constitutes support, how students should be encouraged to become more independent learners and how to avoid doing assignment problems for students.

To avoid a conflict of interest Centre staff are generally not drawn from the pool of teaching assistants and laboratory tutors of the programming modules undertaken by students. However this can lead to a tension between the Centre and faculty members as the Centre is often viewed as poaching the "best" tutors from the programming modules. There can also be an artificial tension between the module tutors and the Centre staff, with both groups complaining that the other is "doing the work" for the students.

The Centre's staff are the key to its success. While they need to be skilled programmers they also need to be capable of dealing with the problems of novice programmers in a sympathetic and supportive way. They also need to be skilled in deflecting student queries on how to complete a given assignment into engagement on the aspects of programming that are causing the student difficulties. One technique that proves successful is to ask a student to point out the section in their notes or textbook which is proving challenging. If the student says that they find it all confusing then the tutor offers to start on page one. This usually leads the student to move them towards the area in the notes that are causing them most concern. Tutors also divert students into the discussion of practice problems from textbooks or course notes that help them understand the concepts needed to complete their assignments. Much of this work is done without the use of a computer, although sometimes students seek help in setting up the necessary software on their personal computers to enable them to carry out assignments at home.

While the majority of the Centre's queries come from students learning Java, C and C++, these are not the only programming languages being taught within the University. Hence the Centre

needs to provide support for students encountering difficulties with languages as diverse as Processing, Eiffel, Perl, Lua and ARM. Thus the Centre staff needs to be drawn from those with a wide range of programming experience. For languages that are less widely used support may only be provided on certain days aligned to the availability of staff that are familiar with the language in question.

3. Lessons Learnt from the First Decade

The first lesson learnt was that you can never have too much publicity – it took quite a few years for the student awareness of the Centre to rise from an initially low value of 44% to close to 87%. Visits from Centre staff to individual class groupings seem to be most effective, followed closely by student surveys where they are asked to answer questions about the Centres services (this serves a dual purpose as it allows us to gather feedback on the centres performance). The webpage needs to be updated regularly, otherwise students question the validity of the information provided.

The Centre needs to work very closely with Faculty members in relation to tutorial assignment schedules and the number of assignments given. Getting students to engage in a timely fashion with course materials is best achieved by encouraging academic staff to set regular assignments or homework. Students are prone to work in a just-in-time fashion and are inclined to seek help very shortly before an assignment is due. This runs contrary to the Centres mission to help students develop their programming skills. The Centre is not a service for the completion of assignments.

Academic staff needed to be reminded that the Centre is a supplement to existing laboratory and tutorial classes and is not intended to allow them to focus on teaching the "top-half" of the class with the view that the Centre will help the remaining students who find programming more challenging. The Centre has also proved useful in helping to make faculty members more aware of the substantial body of literature available on the teaching and pedagogy of programming (See, for example, [7] for an excellent overview of this field).

It was found that few assignments are set during the first weeks of any term, so the Centre does not open until week three of the academic teaching year. The opening hours must be very carefully chosen as it was found that if students were free before or after their programming laboratory class then they would often only attend the centre at these times. On one memorable day almost 80 second year Engineering students arrived at the Centre directly after their programming class. While the Centre developed strategies for coping with very large influxes of students (having additional staff members on call during peak hours and dividing the students into separate groups depending on their needs), it was decided to avoid opening in these timeslots. It was also found that attendance at the Centre on Fridays was very low and so the Centre no longer opens on this day of the week.

The Centre encounters students whose programming difficulties are rooted in the transition to third level study or in other aspects of University life. Clearly such issues are outside both the remit and skillset of the Centre's staff. These students are directed to the University's Tutorial Service which provides a confidential support service to all undergraduate students. The Centre has also had to deal with students who treat the facility as a friendly place to go between lectures and has put strategies in place to encourage these students to become more engaged with the College community (e.g. by suggesting they join some of the student societies which have common room areas where students can congregate between lectures).

One successful innovation was the introduction of small group appointments for students who have already attended the centre – such appointments are seen as a way of encouraging students to continuously engage with programming challenges outside of coursework deadlines. Groups requesting an appointment are put in contact with an available tutor and they then arrange a mutually agreeable time to meet up. Such appointments are limited to at most one hour and sometimes may be much shorter. Appointments are deliberately scheduled with a

variable lead time of at least one day so that students cannot reliably align an appointment with a deliverable deadline. As a result of the introduction of the appointment system the centre has reduced the number of drop in service hours from sixteen to ten.

3.1 Evaluation

The TCD Programming Support Centre was funded under an initiative to improve the retention and completion rates on courses with a significant I.T. component. The original funding application argued that offering a service that aims to encourage students to become more independent, self-directed learners should hopefully lead students to be more successful in overcoming difficulties they encounter during their studies. However there are no well-established metrics for measuring student retention and so it is difficult to measure the Centre's direct impact on retention [5]. Rather annual evaluation of the Centre's performance is carried out with a view to assessing its effectiveness and its impact on student learning.

The Centre complies with the University's ethical policies on data collection and retention. Module lecturers are not informed which students from their class have made use of the Centre's services. Students are told what data is collected and how it is used and stored. Any data collected is anonymised. The data recorded includes the number of individual student visits per week (and whether these were to the drop-in or to appointment sessions), what courses the student queries were drawn from, what their programing problem was (e.g. with linked lists, with setting class paths). The number of repeat visits by individual students is also recorded as the Centre aims to help students become more independent learners. As mention previously student surveys are also conducted regularly and students who visit the centre are encouraged to fill in comment cards about the service. All of this data is reviewed annually and used to assemble a report on the Centre's performance for the funding body. This reporting process also provides an opportunity for the Centre staff to comment on its operation and to reflect on how its services can be improved.

4. Conclusion

The Trinity College Programming Support Centre has grown steadily over the past decade and continues to strive to engage with students who are finding programming particularly challenging. The Centre's service set has evolved, not only to meet changes in students' needs, but also to engage more fully with Faculty to make them aware of their students perceptions of the challenges they face when undertaking a programming assignment. The Centre staff have been active in considering the problems experienced by novice programmers (see for example [1, 5, 8, 13, 16, 17]) and will remain engaged with the Engineering and Computer Science Education research communities in order to continue to improve the quality of the service offered. The Centre is also seeking to broaden its remit by incorporating support in areas shown to influence student success - for example those identified using the concept of Computer Experience [13]. The Centre must constantly seek ways of engaging with students. Social media, synchronous online support and a greater emphasis on autonomous E-instructional resources all feature in the Centre's strategy document for the next decade.

Acknowledgements

The TCD Programming Support Centre is funded from the Information Technology Investment Fund administered by the Higher Education Authority, Ireland.

References

1. Meriel Huggard, "Programming Trauma: Can it be avoided?", British Computer Society Grand Challenges in Computing: Education, Newcastle, England, 2004, pp. 50-51.

2. T. Jenkins, “On the Difficulty of Learning to Program”, Proc. 3rd LTSN for Information and Computer Science Conference, Loughborough, UK, 2002, pp.65-71.
3. L.J. Mazlack, “Identifying potential to acquire programming skill”, Comm. ACM, Vol 23, 1980, pp14-17.
4. P. Byrne and G. Lyons, “The effect of students attributes on success in programming”, Proc. ACM ITiCSE, 1999, pp. 49-52.
5. Eileen Doyle, Ioanna Stamouli and Meriel Huggard, “Computer anxiety, Self-efficacy, Computer Experience: An Investigation Throughout a Computer Science Degree”, Proc. IEEE Frontiers in Education, 2005, pp. S2H-3-S2H-7.
6. L.E. Winslow, “Programming Pedagogy – A Psychological Overview”, SIGCSE Bulletin, Vol. 28, No. 23, 1996, pp. 17-22.
7. Arnold Pears, Stephen Seidman, Lauri Malmi, Linda Mannila, Elizabeth Adams, Jens Bennedsen, Marie Devlin, and James Paterson, “A survey of literature on the teaching of introductory programming”. SIGCSE Bull. Vol. 39, No. 4, 2007, pp. 204-223.
8. Ioanna Stamoui, “Learning Object-Oriented Programming from the Students’ Perspective”, Ph.D. Thesis, University of Dublin, Trinity College, 2009.
9. Tiobe Programming Community Index, http://www.tiobe.com, accessed 14 May 2011.
10. S. Hadjerrouit, “Java as first programming language: a critical evaluation”, ACM SIGCSE Bulletin, Vol. 30, No. 2, 1998, pp. 43-47.
11. R.P.Mody, “C in education and software engineering”, ACM SIGCSE Bulleting, Vol. 23, No. 3, 1991, pp. 45-46.
12. M. Savage and T. Hawkes, *Measuring the Mathematics Problem*, The Engineering Council, London, U.K., 2000.
13. Meriel Huggard and Ciaran Mc Goldrick, Computer Experience - Enhancing Engineering education, International Conference on Engineering Education, Puerto Rico, 2006, pp. T4C-21-T4C-25.
14. D. Lawson, A. C. Croft and M. Halpin, “After the diagnostic test – what next? Evaluating and enhancing the effectiveness of mathematics support centres – part 1”, MSOF Connections, Vol. 1, No. 3, 2001, pp. 19-23.
15. D. Lawson, A. C. Croft and M. Halpin, “After the diagnostic test – what next? Evaluating and enhancing the effectiveness of mathematics support centres – part 2”, MSOF Connections, Vol. 2, No. 1, 2002, pp. 23-26.
16. Ioanna Stamouli, Eileen Doyle and Meriel Huggard, “Establishing Structured Support for Programming Students”, Proc. IEEE Frontiers in Education, 2004, pp. F2G-5 - F2G-9.
17. Ioanna Stamouli and Meriel Huggard, “Object Oriented Programming and Program Correctness: The Students’ Perspective”, Proc. ACM ICER, Kent, U.K., 2006, pp.109-118.