Taxing Our Best Students

Janet Carter (`J.E.Carter@kent.ac.uk`)[1]
Nick Efford (`nde@comp.leeds.ac.uk`)[2]
Stephan Jamieson (`stephan.jamieson@durham.ac.uk`)[3]
Tony Jenkins (`tony@comp.leeds.ac.uk`)[2]
Su White (`saw@cs.soton.ac.uk`)[4]

Abstract: A significant challenge that faces any teacher of introductory programming is the diversity of the class. At one extreme there will be students who have never programmed before, while at the other there will be students who have many years experience of programming.

Handling this diversity is difficult. The temptation for the instructor is often to focus on the novice group and to assume that the others will get by with minimal supervision. This is understandable, but it can be risky. There is a very real risk that the neglected group of experienced programmers become bored and disengage from the course. At the worst, they can lose motivation and fail or drop out altogether.

This paper describes and presents the outcomes of a project aimed at challenging the more experienced programmers in four introductory programming classes at four different UK institutions. The project took the form of a competition in which students were asked to devise and solve a series of programming challenges.

Keywords: programming, competition, diversity, retention, motivation

1   Introduction

The diversity apparent in any introductory programming class presents a challenge to the instructor. In a typical class there will be many students who have never programmed before and who have little or no idea of what the activity involves. The trials and tribulations of this group have been well documented. At the other end of the spectrum there will be students who have been programming for many years; some of these will have been formally taught and others will be self-taught. There may even be students who have worked for many years as professional programmers. The question facing the instructor is how to present a course that will engage (and be of benefit to) all these students.

This diversity does not mean that some of the students cannot benefit from a programming course. Rather, it is simply that their needs are different. A novice needs to understand the mysteries of loops, conditional statements and all the usual programming minutiae. But the experienced programmer can still learn; there is a chance for consolidation, to pick up a new language, or to explore more advanced topics. The challenge for the instructor is how to provide content suitable for both these groups and, of course, all those who fall between.

Many institutions are becoming increasingly concerned about attrition rates in computing courses. In the UK, computing courses routinely have among the highest drop-out rates of any subject, something that is obviously of great concern and a definite "hot topic" in the discipline. Any initiative aimed at addressing this problem can only be encouraged. When applied at the level of a single course, it is clearly vital that instructors enthuse and motivate their students [2]. This is especially true of programming [9].

---

[1]Computing Laboratory, University of Kent, Canterbury CT2 7NF, UK
[2]School of Computing, University of Leeds, Leeds LS2 9JT, UK
[3]Department of Computer Science, University of Durham, Durham DH1 3LE, UK
[4]School of Electronics and Computer Science, University of Southampton

A glance through the proceedings of any conference addressing computing education will reveal many papers discussing the problems of teaching programming to novices. Any number of tools and techniques have been described. Most have their own band of enthusiastic supporters and some claim significant success. Most have their skeptics and detractors. But it seems that at the moment the emphasis in innovation in teaching programming is firmly on the problems faced by the novice.

The existing literature on the motivation of students also appears to take as its focus those students who are in some sense reluctant to learn [8]. Although this is questioned [7] the premise appears to be that students are not motivated and that something needs to be done about it. The students who are motivated appear to be neglected, at least until they have lost their motivation!

Discussion of the problems and motivations of novice programmers is not to be discouraged, of course. It is, however, important to remember that this work does not address the needs of a significant part of the cohort. It is not difficult to appreciate that students who have worked as professional programmers are unlikely to warm to being asked to develop trivial (to them) programs using tools such as BlueJ [3] or Alice [1]. Their needs are different, and if their needs are not addressed they may well lose motivation and interest. And if they lose motivation and interest they will most probably leave.

The project described in this paper – TOPS ("Teaching Over-Performing Students") – was conceived as an attempt to address the needs (both educational and motivational) of these students. Within the framework of a reasonably light-hearted competition, the intention was to stretch and develop their existing programming talents. These students already had some level of mastery of the subject material, and so the challenge was to ensure that they achieved high levels of engagement [2] and thus extended their knowledge. We also hoped to achieve the recognized benefits of facilitating reflection [4] by encouraging our (potentially) high-achieving students.

The remainder of this paper reports on the organization and outcomes of this competition.

2   Context

The four institutions involved in this project were the Universities of Durham, Kent, Leeds and Southampton. All of these have computing departments of some repute, and all recruit from the same, higher, end of the pool of students. All four currently use Java as their main introductory programing language.

Each institution also has existing arrangements in place to support programming students with different pre-existing skills. For example, at Leeds [6] a "Fast Track" is provided for the experienced students, covering advanced aspects of Java such as GUI programming and threading, and Southampton offer much the same [5]. At Kent an informal group called "Cool Stuff in Computer Science" holds evening meetings in which students engage in collaborative projects, usually involving robots, and invite academics to give talks about topics they would like to know more about. Both Leeds and Durham allow students to specify a programming project that forms part of their final assessment, in the hope that the more advanced students will choose more demanding and challenging projects.

Each institution also has, of course, its own ways of supporting those students who find programming especially challenging, but a discussion of these is outside the scope of this paper.

3   Existing Competitions

There are, of course, a number of programming competitions aimed at computing students, and the question must arise of why the students could not simply have entered one of these. Why do we need another programming competition?

Some of the existing competition are sponsored by large corporations such as IBM and Microsoft and there are also those promoted by the professional bodies (notably, in the UK, the British Computer Society). These can indeed be used to motivate students (and all the participating institutions actively encourage their students to take part), but the practicalities can often outweigh the advantages. Some use development tools that the students are not learning, and some follow timetables that make it impossible for students to attend as required. Many seem biased towards the US curriculum.

4   Planning

The aims of the project can be summarized:

- To build a community of practice among those working to address the needs of the most able students.

- To produce a proof-of-concept activity to show how it is possible to motivate and challenge the most able students.

- To identify future work that could support others seeking to develop the most able students.

The plan was to design a competitive activity within the context of an existing curriculum. While the four participating institutions did not follow the same curriculum, there was enough commonality in both aim and practice to make the design of a competition reasonably straightforward.

Java clearly provided the common platform for programming, and also provided a useful avenue for seeking sponsorship from Sun Microsystems. Apart from this, there was sufficient difference between the programming environments used to mean that any competition could not rely on a particular development platform or indeed operating system.

Previous experience in similar projects had shown that a "sense of place" can be invaluable when working with colleagues from other institutions. The main competition was therefore augmented by a series of visits (Kent to Southampton, Leeds to Durham and vice versa) whereby those managing the activity could gain an insight into how colleagues at other institutions went about teaching programming.

5   The Competition

The competition was intended to be serious, but at the same time still fun. It was effectively part of the students' program of study, so it was important that it addressed the various learning objectives set by the four participants.

5.1   Format

The competition that was devised involved students working in pairs to tackle various Java programming challenges. Collaborative programming is an increasingly popular method of learning (and programming) [10] and so it seemed only sensible to adopt it.

After some initial discussions, it was decided to allow the students to set the challenges themselves, so that each team would set a challenge for each of the other three. If the event was to be completed in one day, the time available for the tackling of the challenges would be relatively short, so it was decided that each should be designed so as to be doable by a pair of programmers in one hour.

5.2   Choosing the Participants

The existing teaching at each institution had already identified those students who might be seen as potential participants. The available funding would allow about six to attend the competition itself, so some sort of selection process was required at each venue. The local details of this process were left to local staff to decide.

Durham held a party to choose their team, Kent students held a selection meeting, with the team eventually being chosen on the flip of a coin. Leeds and Southampton both found that the teams almost selected themselves, with the number of interested students being the number required. At each institution the selection was public in that all students in the year knew about the competition and were encouraged to support "their team".

5.3   Venue

Ease of transportation meant that London was the obvious place to hold the event. Sun Microsystems offered to host the event at their "Tech Day" at Central Hall in Westminster on 14th March 2007. This was especially opportune as this was a Java-based event, and the students would also have the chance to attend a keynote address by no less a Java figure than James Gosling.

Sun included the competition as a "proper" part of the day, including signs and so on. All the participants registered for the Tech Day, and there was plenty of time for the students to wander around the rest of the show.

5.4   Setting the Challenges

The setting of the challenges was not restricted to those students who would attend the competition day itself. Around eight students were involved at each of the institutions, a useful way to involve students who were unable for whatever reason to travel.

The students were given a week to create an outline of their challenge. These were due well before the event itself so that the challenges could be validated by staff from the other institutions. Once approved, the students had a further two weeks to create a detailed specification, including any required files and a marking scheme.

The challenges that were created were all very different. Southampton devised a "tamagotchi" program, relating to some aspects of their programming course. The Durham team provided a "Six Degrees of Separation" program, allowing the user to work out who is linked to who and in what way; this, they reasoned, would be useful at the Tech Day if they wanted an introduction to James Gosling. Kent also gave a challenge relating to the Tech Day itself – processing GPS data to navigate around the event. Finally, Leeds took a slightly different view of the problem, and created a debugging challenge.

5.5   Logistics

With the challenges ready, and even with a date set, transporting all the participants to the venue was not trivial. The UK public transport system is complex, and aspects of it are often arcane and baffling. A good deal of money was saved, for example, by having the Leeds team travel half way to London, leave their London-bound train, and await a slightly later London train from an adjacent platform to complete the journey.

The students from Durham had a greater distance to travel than the other three teams, so they took the train to London the day before the competition and stayed overnight at a local youth hostel. The Leeds, Kent and Southampton teams travelled to London on the day, all setting off at

Figure 1: The Competition Room

approximately 7am. The Leeds team managed to arrive just before the actual competition began but all the others arrived in time for James Gosling's opening keynote.

After the keynote the students followed the signs to the room the competition was held in; the room was set up ready for the students as they arrived. Each group took command of a different corner of the room. From this point, each pair of students was now competing against every other pair in the room, even those from their own institution; the collaboration had ended with the production of the challenges.

5.6   Attempting the Challenges

Six students, working as three pairs, had attended the competition from each institution. Each pair had one hour to attempt each of the challenges set by the other three institutions. The specification required that each challenge should be solvable using just a laptop and basic tools, but many students took extra equipment with them, just in case it was allowed or needed.

At the end of each challenge, markers collected the responses and removed them for marking. The final marking session gave the students the chance to wander around the rest of the event. Two images from the competition are included as figures 1 and 2 to give a sense of the event.

5.7   Marking and Prizes

The marking schemes provided by the students were adequate, and were used to rank each pair's attempt at each challenge. This ranking was then used to determine the overall winner.

A prize was provided by Sun for the winning pair. In addition, there was a prize for the best challenge. The prize for setting the best challenge was awarded to each participant from the winning institution, including those who did not attend the Tech Day.

Figure 2: Challenged Students

6 Reflections

The project can be seen as a success but, as with any activity of this kind, there were unexpected issues and things that could have been done better.

6.1 The Challenges

Setting programming tasks to be completed by students from different institutions is not straight-forward. The competition took place in March, which was before the end of the term at all four institutions. Questions therefore arose of what had been covered in each course. It would have been disastrous, for example, if one team had set a challenge that required an aspect of Java that the other teams had not learned.

This issue was addressed by staff comparing notes about syllabus and curriculum well before the competition started. It was complex but not insurmountable.

This is also where the "sense of place" achieved by visits and peer observations became invalu-able. A visit to another institution, and a chance to meet there with staff and students, gave participating staff an insight into how the subject was taught at one of the other institutions. It is something that everyone who teaches programming should do when they get the chance.

With hindsight, though, some of the challenges may have been a little too complicated. It is not possible to write a great deal of Java in just one hour, something that could perhaps have been better appreciated. Some members of the Leeds team were discovered still working on the Southampton challenge several weeks after the competition!

6.2 Student Feedback

A key aim of the project was to enthuse and motivate the students. They certainly appeared to enjoy the day. The atmosphere was rather tense as they attempted the first challenge, but once the students settled down to working on tasks that were intended to stretch their abilities, things became rather calmer.

The students were asked to give their feedback on the day:
"Working together was great. Everyone worked amazingly well in teams".
"I liked that we were supposed to work at our natural pace and that we had to think."
"It was really intense, but great fun."

7 Future Work

We believe that this project has served as an effective pilot for an activity that should be extended.

7.1 Expansion

The only significant limit on the number of teams taking part was the budget available for travel and prizes. We will seek to extend the competition in the future to involve other institutions. This will have two benefits: it will obviously give more students the chance to compete, but it will also allow more programming teachers to observe and learn from the practice of their colleagues.

7.2 Timing

Holding the competition during the academic year was not a major problem but, especially if more institutions were to become involved, it would be easier to hold it at the end of the session. At this point travel difficulties would be less and those setting the challenges could be more confident in the topics studied by the others.

It has also been suggested that the competition might be extended to students at the end of their second year. While including students at this stage of their careers was never an aim of this project, we will consider how others could be involved.

8 Conclusions

We believe that stretching our best students is as important as supporting the weaker students. In the context of introductory programming this means that providing motivating challenges for the students who arrive as programmers is as important as providing scaffolding to support those who arrive with no such experience.

We also believe that the competition described in this paper has been an effective method for both stretching and supporting these students. They have had the opportunity to tackle quite complex Java programming challenges, and they have had a chance to attend a high-profile industry event. Aside from this, they have also gained an insight into the difficulties of setting academic assignments!

We will seek to run the competition again in the next academic session, hopefully with more students from a more diverse range of institutions.

Finally, an intangible benefit of this project is that the most able programming students from four institutions have had the chance to meet. We recognize the benefits of their instructors meeting and sharing knowledge and experience, and it is entirely possible that this is also true of our students.

9 Acknowledgments

References

[1] Alice. Alice – Learn to Program Interactive 3D Graphics. On line at `http://www.alice.org/`.

[2] J. Biggs. *Teaching for Quality Learning at University*. Society for Research into Higher Education, 1999.

[3] BlueJ. BlueJ – the interactive Java environment. On line at `http://www.bluej.org/`.

[4] J. Cowan. *On Becoming an Innovative University Teacher*. Society for Research into Higher Education, 1998.

[5] H. C. Davis, L. Carr, E. Cooke, and S. White. Managing diversity: Experiences teaching programming principles. In *Proceedings of 2nd Annual LTSN for Information and Computer Science Conference, London, UK*, pages 53–59. Learning and Teaching Support Network for Information and Computer Science, 2001.

[6] J. R. Davy and T. Jenkins. Research-led innovation in teaching and learning programming. In *Proceedings of ITiCSE '99, Krakow, Poland*, pages 5–8. ACM, 1999.

[7] N. Entwistle. Motivation and approaches to learning: Motivating and conceptions of teaching. In S. Brown, S. Armstrong, and G. Thompson, editors, *Motivating Students*, pages 15–23. Kogan Page, 1998.

[8] S. Fallows and K. Ahmet. *Inspiring Students: Case Studies in Motivating the Learner*. Kogan Page, 1999.

[9] T. Jenkins. The motivation of students of programming. Master's thesis, University of Kent at Canterbury, 2001.

[10] J. T. Nosek. The case for collaborative programming. *Communications of the ACM*, 41:105–108, 1998.