Variety is the Spice of Life: Creating and Retaining Enthusiasm in Our Students

Dimitri Panayi **Xiaosong Li** **Alison Young**

School of Computing and Information Technology
Unitec New Zealand,
Auckland, NZ
dpanayi@unitec.ac.nz

This paper suggests that we as educators of programming principles need to make a fundamental mindset shift in the way we construct our teaching material, by thinking 'outside the box', and using innovative teaching strategies. It hopes to ignite further thought and allow self-analysis of our current educational practice of teaching programming.
The teaching of programming is globally recognized as difficult. Creating and retaining enthusiasm within our students is a challenge with high drop-out rates being common. Programming has a long history of pre-conceived and standard ways of delivering key programming constructs and ideas, instilled from a generation of educators that were taught this way themselves. More recently, educators of programming have been exploring ways of involving students and allowing 'active' participation to occur.
Programming is also commonly thought of as being too theoretical and pattern-oriented; hence the tendency to think it lends itself well to a pure, traditional lecture-based environment. This paper shows that this is not the case, in other fields of study 'active' learning and other experiential approaches have proven to be very successful.
We have also reviewed innovative teaching strategies and reflected on them within our own teaching practices. In particular, making use of students as 'active' participants using experiential, problem-based learning, and using portfolio assessment, are explored within the context of teaching programming. In each of the following five sections, we discuss an example teaching strategy, and finally we summarize the paper.

Keywords

Teaching strategies, experiential learning, Active learning, Participative learning, Portfolio assessment, Programming courses.

1. INTRODUCTION

The teaching of programming is globally recognized as difficult in creating and retaining enthusiasm within our students, with high drop-out and low passing rates. Programming has a long history of pre-conceived and standard ways of delivering key programming constructs and ideas, instilled from a generation of educators that were taught this way themselves. Programming is also commonly thought of as being too theoretical and pattern-oriented; hence the tendency to think it lends itself well to a pure, traditional lecture-based environment, where the lecturer is the repository of all knowledge and has the duty of imparting that knowledge to the students, "empty vessels" into whom knowledge is funneled, usually through a series of lectures to which the students listen, and from which they hopefully 'learn'. It is reported by many programming lecturers that this traditional approach doesn't help to address the existing common issues in programming courses.

More recently, educators of programming have been exploring ways of involving students and allowing 'active' participation to occur. A number of reports on these innovative teaching practices have been studied. From these reports and our own teaching practice we realized that we as educators of programming principles need to make a fundamental mindset shift in the way we construct our teaching material, by thinking 'outside the box', and using innovative teaching strategies. We hope to ignite further thought and allow self-analysis of our current educational practice of teaching programming.

The central principle involved in the innovative teaching strategies that we studied is active learning, where innovative teaching strategies are used to allow learners to be actively engaged in programming teaching environment. In other fields of study active learning and other experiential approaches have proven to be very successful. It is argued that traditional lectures promote a surface approach to learning which might be effective in some disciplines or subjects but which is inadequate in such a practical subject as programming (Jenkins, 1998). Without active learning, it has been proved that students don't

gain deep understanding about some programming concepts such as OO (Object Oriented) (Biddle, 2000).

This paper studies some of the innovative teaching strategies and reflects on them with our own teaching practices. In particular, making use of students as 'active' participants using experiential, problem-based learning, and using portfolio assessments, are explored within the context of teaching programming. In the following sections, we discuss five examples of teaching strategies to facilitate active learning.

2. USING GRAPHICS TO MOTIVATE THE STUDENTS

From the research we have studied it appears tha it is a common practice that lecturers demonstrate programming concepts graphically. From students' feedback, we understand that this approach is helpful, but not sufficient to address those issues indicated in the first section. A possible reason for this could be that not all graph type explanation is easily understood by all students without good logical training.

Gearailt (2002) has reported a different approach of using graphics to teach Java programming concepts to get the students interested enough to put in the effort needed to understand stored programs, variables, loops, selection, functions, testing, incremental development, arrays and persistence. Gearailt further argued that it is possible to improve the degree of active learning taking place by introducing colour and other graphical displays at an early stage. In this study they allowed the students to program on a sequence of rapidly-changing images which looks as if something is actually moving. It was then reported that the students absorbed almost without effort many of the programming concepts such as loops and arrays.

Reflecting on this approach with our own teaching, we agree that graphics does promote active learning. Our past experience of requiring the students show a photo for each item in their assignment application confirmed this. However, in some situations, programming and demonstrating an animation picture requires even more complicated programming techniques than loops and arrays. How do you get the students gain these techniques in the first place?

3. A PHYSICAL DEMONSTRATION APPROACH

Jenkins (1998) reported an innovative approach in teaching introductory programming. The key idea of this approach is to map the students onto the program components or computer roles, and let the students simulate computer activities.

For example, in case of teaching arrays, they use a group of the students to perform the functions of an array with each student taking the part of a single element. Once the array has been set up the instructor can demonstrate some common applications and operations using arrays. The instructor then explains that each member of the array can remember only one value, but that the array as a whole can remember many values. The first step with the array set up is for the instructor to "initialise" the array by telling each student in turn to record some initial value. The other students will observe the necessary "loop" procedure. The process can then be written down in pseudo-code or Pascal as required, and is then repeated with more direct reference to the code itself.

It is reported that this approach is particularly good to present dynamic processes and is suitable for the students with a knowledge gap. A number of our staff have successfully used this approach and it makes a lot of sense to us.

As programming has a lot to do with low level computer operations, and these operations are sometimes underlying the program, they are therefore not obvious to the students. This is why it is hard for a student to understand a piece of programming code when such a low level operation is involved. Letting students physically simulate computer activities brings these operations to the surface. We believe that this approach is helpful in teaching remote computing. For example, letting one student play the server and several students playing clients, then messages can be passed among the students.

The issues we can see with this approach is that it does graphically demonstrate the operations to the students and by actually acting out the part they gain

a better understanding of the functions they are required to code. These "ice breaking" activities are an excellent way to demonstrate programming principles but it is crucial for the lecturers to carefully transfer the students activities back to the programming concepts they are trying to impart.

4. USING A TECHNOLOGY-ENHANCED LEARNING ENVIRONMENT

Clancy *et al*. (2003) created a flexible new technology platform that would enable students and instructors to utilize computer technology effectively in all stages of an undergraduate course. Instead of listening to lectures, students would be participating in computer-based activities (e.g., programming exercises) and instructors would be freed up to interact more closely with students as they worked in pairs or small groups. It consists of three tools: (1) a course builder, which enables instructors to develop online activities for students, (2) a web-based learning environment, which delivers all student activities and collects all student work in a relational database, and (3) a course portal, which serves as the learning management system for the course.

This tool is very similar to our web based learning environment, Blackboard. However, we didn't use our Blackboard system to such an extent. This approach suggests to us that Blackboard can motivate active learning and we have a big potential to use our online web based system more efficiently. It also suggests that we could develop more advanced teaching tools. An animation tool that simulates the students' activities described in section 3 might achieve the same results and therefore save teaching resources. However, developing a software tool usually has a resource or staffing cost.

5. LETTING THE STUDENTS TEACH THEMSELVES

Jerram (2002) has reported a very exciting approach that lets the students teach themselves. In this approach, they have replaced all the lectures with seminars in which student participation and involvement is the primary focus. The purpose of this approach is to promote critical self-reflection; self-directed learning; development of ability to transfer skills, personal responsibility for domain-specific knowledge as well as application of critical thinking skills. It is reported that the results are successful. The students' teaching sessions unfailingly demonstrated comprehensive competence in their own subject and excellent communication skills in helping their 'students' understand and acquire the skills and software packages taught.

We were greatly encouraged by this report ans as a result we tried one such teaching session in one of our third year undergraduate courses. The students were mature enough to conduct such a teaching session. All the students knew C++ or Delphi before they came to this course. We picked a VB.NET syntax session since the students already had similar knowledge in C++ or Delphi. The students were all asked to run the session voluntarily and all immediately agreed to do this. They were all then notified and explained the purpose of this approach. Finally only one student volunteered to run the first session. The other students did participate more freely than a usual lecture session. However, the students were asked if they want another such a session in a survey, among 15 responses, only one said "yes", nine said "no" and four said, "don't care".

The issues we can see for this approach are it needs to be well prepared; it assumes that all members of the class are knowledgeable and have something to contribute; the students might think that they are not paying their fees to teach themselves, however students are reluctant to participate in anything that isn't assessed and will count towards their final grade no matter how much they would learn from the activity.

6. PORTFOLIO ASSESSMENT

Portfolio assessment provides for collaborative reflection, including ways for students to reflect about their own thinking processes and metacognitive introspection as they monitor their own comprehension, reflect upon their approaches to problem-solving and decision-making, and observe their emerging understanding of subjects and skills (Pikulski and Cooper, 2004).

In portfolio assessment, it is especially important for teachers and students to work together to prioritize those criteria that will be used as a basis

for assessing and evaluating student progress, both formatively and summatively. Additionally, they can work collaboratively to determine grades or scores to be assigned (Pikulski and Cooper, 2004). Since students are involved in this process, portfolio assessment should facilitate active learning. Plimmer (1999) and Jerram (2002) have used portfolio assessment and reported successful results.

We have made some efforts in this direction, for example when we mark the Web application assignments, we give the students' an option to participate in the marking process. We let the students demonstrate their program to the lecturer and answer questions. We give them opportunities to demonstrate any additional features they may have added. Then we repeat our marking criteria to the students and tell the students which grade category they have achieved. We also ask the students if they have any disagreement with the grade given. We have been using this approach for five semesters, and in each of them we have above 70% of the students taking the option to participate in this grading option. A further pleasant but unexpected outcome is that we rarely get students disputing their grade for the assessment.

7. SUMMARY AND FUTURE WORK

We have discussed five examples of teaching strategies to facilitate active learning in a programming course. None of them is used as a sole teaching strategy. We believe we need a balanced approach for our teaching, applying different teaching strategies at the different stages of teaching and for different aspects of content.

We have analysed existing innovative strategies and discussed possible strategies that might improve our students' learning in the future. One common feature of these teaching strategies is they require innovation and inspiration by the lecturer. However in this study neither the students' learning styles nor any extra resource requirements have been taken into consideration. In the future we plan to trace the current status of the examples and apply suitable examples to our own teaching practice, and then evaluate and publish the results.

REFERENCES

Biddle, R. (2000). Active Learning For Object-Oriented Design. Proceedings of OOPSLA 2000 Companion Minneapolis, Minnesota, p137-138

Clancy, M., *et al* . (2003). New Roles for Students, Instructors, and Computers in a Lab-based Introductory Programming Course. Proceedings of *SIGCSE'03,* February 19-23, 2003, Reno, Nevada, USA. p132-136

Gearailt, A. (2002). Using Java to increase Active Learning in Programming Courses. Proceedings of Principles and Practice of Programming in Java 2002, p107-112

Jenkins, T. (1998). A Participative Approach to Teaching Programming. Proceedings of ITiCSE '98 Dublin, Ireland, p125 – 129.

Jerram, C. (2002). Applying adult education principles to university teaching. Proceedings of HERDSA (pp. 369-375), Perth, Australia, 7 - 10 July.

Lister, R., Learney, J. (2003) . Introductory Programming Criterion-Referencing, and Bloom. Proceedings of *sigcse '03*, February 19-23, 2003, Reno, Nevada, USA. p143-147

Pikulski, John and Cooper, D. (2004) Portfolio Assessment. Accessed May 27, 2004. http://www.eduplace.com/rdg/res/literacy/assess6.html

Plimmer, B., (2000). A Case Study of Portfolio Assessment in a Computer Programming Course. Proceedings of the 13th Annual Conference of the National Advisory Committee on Computing Qualifications, July 2000,Wellington, New Zealand. p 279-284

Prince George's County Public Schools. (2004) Portfolio Assessment. Accessed May 27, 2004. http://www.pgcps.pg.k12.md.us/~elc/portfolio.html