Experiences of student teacher in a Scottish high school – a case study.

Adam Kurkiewicz
School of Computer Science
University of Glasgow

March 5, 2016

Abstract

From October 2015 until March 2016 I have participated in teaching Computer Science in a Scottish high school through a partnership of University of Glasgow, Cleveden High School and a NGO "Science Connects". What follows is a case report on my experiences of teaching, personal reflections on my teaching methods, perceived difficulties of students, challenges regarding both the process of teaching and the process of learning as well as some observations on the Scottish education system.

1 Teaching Materials & Methods

As part of my teaching I delivered 9 workshops centered around programming and algorithms, which are my areas of interest. 8 of these workshops were 'double-period' 90 minutes workshops, and were almost entirely interactive. One workshop was "single-period", 45 minute workshop. Several teaching methods were employed.

1.1 Guided Discovery

An example method was Guided Discovery, which focused on students independently, but with guidance, rediscovering a studied concept. An example of this method is a workshop I delivered on sorting algorithms. Students

would rediscover a sorting algorithm (selection sort) starting from an unformal sorting method (just sort those cards), through a more formal method (split the array of cards into two sections, left and right, left section initally empty, using a single swap make the left section one element bigger, preserving a rule that left section contains only fully sorted elements. Repaeat until finished) through developing pseudocode on paper to eventual implementation in visual basic on a computer.

1.2 Finish the Implementation

Another teaching method used was FINISH THE IMPLEMENTATION Students were presented with a partial implementation of an algorithm and were requested to write between 1-10 lines of code to finish the implementation. I originally planned students to solve around 10 such problems per 90 minute session, however in practice only about 3 prolems per student were solved with the rest of the problems assigned as homework.

1.3 Gamification

Yet another teaching method used was GAMIFICATION. Students were chosen to represent function calls of a recusrive function. An inital student had to ask another student for a value of the function with specified parameters and would later "freeze" waiting for an answer. A student asked would either give an answer straight away (recursive base case), or make another recusrive call. Upon obtaining an answer from the student asked, the frozen student would unfreeze and continue the execution of an assigned function call, which could mean performing an elementary operation, like addition, freezing on another function call or answering a frozen student who originally asked for the value of the function call.

1.4 Workshops

Let us remark that the traditional LECTURE-STYLE method was never used, and perhaps more standard today LECTURE AND ACTIVITY method was used only once out of 9 workshops. The following topics were covered in the workshops:

- Virtual Machines (LECTURE AND ACTIVITY)

- Assignment Operator (Finish the Implementation)
- Sorting Algorithms I (Guided Discovery)
- sorting algorithms II (Finish the Implementation")
- operations on arrays (Finish the Implementation)
- Recursion I (Guided Discovery)
- Recursion II (Finish the Implementation)
- Stacks and Queues(Guided Discovery)
- Program Execution (Guided Discovery)

Finally let us remark that the choice of teaching methods employed was taylored to the classrom size (4), student age (Advanced Higher, 17 years old), and the attitude of the supervising teacher (freedom to experiment, allowance to teach a lot, and expectations to teach algorithms aligned with my interest).

Most of the materials can be downloaded from github (Kurkiewicz 2016a; Kurkiewicz 2016b; Kurkiewicz 2016c).

2 Evaluation of the methods employed

Firstly, let us take note of the special situation I found myself in. With the class size of 4 and two teachers in the class at all times the teacher:student ratio was 1:2, which is a huge departure from regular 1:(20-30). I am painfully aware that such favourable ratio cannot be maintained at all classes in all schools for political and economical reasons. I have not gathered any experience at teaching regular class sizes, therefore I could not say how well the methods would scale with the number of students. With this reservation in mind I present a list of identified strengths and challenges along with supporting cases.

2.1 Instant feedback

The main reason I have decided to employ such drastically interactive teaching methods is my own expersience of studying at University of Glasgow. Two classes I took over the course of my degree had particularly rich experience, a foundation course in programming (CS1P), and introductory course to mathematical analysis (2E). Both courses had made use of "clickers", which are wireless devices used by the lecturers to gather instant feedback from individual students. The way clickers were used in both courses was through asking the students a question requiring an application of an algorithmic, mathematical or logical type of thinking, followed by a few minutes for students to think the problem through, foollowed by students answering the question individually, followed by presentation of student answers and critical discussion between the lecturer and the audience. The reason I found this method appealing was Instant Feedback both for the lecturer, regarding students understanding of the concept, and for students regarding their own level of understanding.

It's easy to understand the importance of the lecturer knowing what's in the students heads. At first it wasn't obvious to me why the student's awarness of their understanding was so important. Only after I had started delivering my workshops I notices that my students very often would build incorrect or inconsistent mental models of the concepts presented, without any awarness of the problem. One of the students thought that only 3 passes of bubble sort would sort the list. However in practice as many as n passes are required. Only after an actual attempt by the student to implement bubble sort had I and the students realised that he misunderstood the concept. He could also immediately, on the spot, fix his mental model and move on to further expand his knowledge. Instant Feedback allowed us to find the weakest spot in understanding and helped to move on without leaving any misconceptions behind.

2.2 Self-paced learning and going beyond

My class of 4 had students of varying ability. At the very first meeting my teacher remarked that he's not expecting 2 students to pass, wheras the other 2 he considered brighetr. The methods employed in particular "finish the implementation" allowed for self-paced learning. The first question was always warm-up, and with little supervision could be done by all students in

reasonable time. An example here is implementing a function swapping two elements of an array. The last question was almost certainly out of reach, if only due to time conctraint reasons, as the task was to implement a novel sorting algorithm (with appropriate guidance on possible directions). Any student could progress through the workshop in their own time, going as far as they want.

2.3 Deep understanding and forcing to think

The material has been designed in such a way as to expose students to novel, previously unseen tasks without the opportunity of direct application of memorised solutions. The material has been designed to force thinking, and trying modifications of previously seen patterns. One of the students labeled as bright found this approach somewhat shocking. Initially she demanded the answers she could memorise without a serious attempt at the solution on her own. All students admitted that the way they were thaught before involved being shown a piece of code they would have to memorise, and learn to type it into an IDE. Unfortunately the coding skills of students when I started working with them were very poor with all students unable to accomplish even very simple tasks involving a few array operations or writing a single loop without extensive suprevision. After a few sessions of intense programming the ability to solve novel problems increased and even the poorest students were able to solve some problems without much supervision.

2.4 Peer Instruction

Completely unforseen effect of the teaching methods was students starting to help each other. Especially active in this field was one of the students previously ranked as weak, which was a warning to me to not label students prematurely.

3 Challenges

Some of the methods employed had drawbacks, which were difficult to overcome. In particular the FINISH THE IMPLEMENTATION method had sometimes demotivating effect on students. Students would often procrastinate

or just freeze in front of the screen "not knowing what to do". Given a small group of students this was overcame with the help of favourable teacher:student ratio, as there were always at least two people helping the students progress if they get stuck. I also tried to balance the fun with the work by interleaving GAMIFICATION with FINISH THE IMPLEMENTATION.

Another challenge was finding the time to prepare the material. It took me about 3 hours to prepare a FINISH THE IMPLEMENTATION workshop, 1-2 hours to prepare a game and 8 hours to prepare the traditional LECTURE AND ACTIVITY class.

A possible drawback was also perceived slowness. As my teacher pointed out, it was taking a lot of time to cover topics clearing all students doubts and insisting on each and every student writing code. Arguably code memorisation could bring quicker resuls, perhaps even effective if the effort is measured by a poorly designed exam. However, I believe that developing deep understanding is worth the extra time spent on material. Unfortunately, due to the broadness fo curriculum programming and algorithms couldn't be focused on 100% of the time.

4 Conclusion

The teaching methods described have succeeded to develop deep understanding of presented concepts in students thaught. It remains to be seen whether the methods will scale to classes of bigger sizes.

References

Kurkiewicz, Adam (2016a). *Practicing Foundations*. URL: `https://github.com/picrin/practicingFoundations`.

— (2016b). *SelectionSortVB*. URL: `https://github.com/picrin/SelectionSortVB`.

— (2016c). *VMWorkshops*. URL: `https://github.com/picrin/VMWorkshops`.