Introducing Teachers to Computational Thinking Using Unplugged Storytelling

Paul Curzon
Queen Mary University of London
London, UK
p.curzon@qmul.ac.uk

Peter W. McOwan
Queen Mary University of London
London, UK
p.mcowan@qmul.ac.uk

Nicola Plant
Queen Mary University of London
London, UK
n.j.plant@qmul.ac.uk

Laura R. Meagher
Technology Development Group
Fife, Scotland, UK
Laura.Meagher@btinternet.com

ABSTRACT

Many countries are introducing new school computing syllabuses that make programming and computational thinking core components. However, many of the teachers involved have major knowledge, skill and pedagogy gaps. We have explored the effectiveness of using 'unplugged' methods (constructivist, often kinaesthetic, activities away from computers) with contextually rich storytelling to introduce teachers to these topics in a non-threatening way. We describe the approach we have used in workshops for teachers and its survey based evaluation. Teachers were highly positive that the approach was inspiring, confidence building and gave them a greater understanding of the concepts involved, as well as giving practical teaching techniques that they would use.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education—*computer science education*

Keywords

Computational thinking, K-12 education, unplugged

1. INTRODUCTION

The UK has recently overhauled the way computing subjects are taught in schools, with similar initiatives being introduced in many countries. The previous ICT curriculum, focussing on the use of current technology, has been replaced by a Computing curriculum based on the development of computer science concepts and skills. This curriculum, which starts in primary school, has computational thinking at its heart. Seymour Papert introduced the term 'computational thinking' in the context of suggesting an alternative, computationally-based mathematics education [9]. Jeanette Wing [10] then turned it into a mainstream idea in computing education: that it was the skill set that learning computer science delivers. How best to teach it, however remains an open question.

A big issue facing these initiatives is that not all the teachers involved have computing backgrounds. This means there is a knowledge and skill gap, especially with respect to subject-based pedagogy. As computational thinking is central to the syllabus, it is important that teachers both have a deep understanding of it and have ways to help students develop the skills. We describe the implementation and evaluation of an initiative to begin to address this problem.

To support teachers in understanding how computational thinking skills fit the new syllabus, the UK Computing at School working group created a framework for introducing computational thinking into classroom activities [4]. This builds on 'Computing Progression Pathways' [8]: an assessment framework which links UK computing syllabus topics to computational thinking skills. Both are based on breaking the computational thinking skills into a simplified set of core areas that can easily be integrated into a practical assessment framework for teachers to apply: algorithmic thinking, evaluation, decomposition, abstraction and generalisation. These are then broken into 'classroom techniques' that encompass the wide range of topics suggested as part of the computational thinking skill set.

We developed 5 workshops for teachers on computational thinking themes based on existing, unplugged style computing activities. Unplugged Computing [1] is the approach to teaching computing concepts using constructivist, often kinaesthetic, activities away from computers. It has successfully been used for teaching computing at all age groups from primary school to university level (see e.g., [6][5]). We used a mixture of activities created and made publicly available by our previous projects [3]. Other activities were being used as part of an introductory undergraduate programming module. The computational thinking themes were explicitly drawn out in our workshops during the activities and reinforced at the end of each. We have suggested [3] that it is important to teach computational thinking in a way that demonstrates how the whole coherently fits together, and that it is understood as a combined skill set from the start. We build on this idea here using unplugged activities embedded in rich open problem-solving stories to give contextually rich scenarios within which computational thinking is applied.

There are several potential uses of unplugged workshops: to teach computing concepts (such as binary numbers) directly to school students; to introduce the unplugged style of teaching to teachers; to demonstrate activities for teachers to use in class, and as an explicit teacher training tool to teach computing concepts to teachers. Our focus here is most closely aligned to the latter use, as part of teacher training with the others as secondary benefits.

The contributions of this paper are two fold. We give a coherent set of workshops for teaching teachers computational thinking concepts, and we provide evidence that it works to fill teachers' knowledge gaps about computational thinking, is confidence building, and is seen as a useful practical way that they can teach computing to school students.

2. THE WORKSHOPS

Our first unplugged workshop focuses on computational thinking in general, the second on algorithmic thinking, the third on unplugged programming techniques and the fourth on the human factors side of computational thinking. All lasted 90 minutes. The style of delivery is an interactive lecture, combining volunteers demonstrating activities at the front with short whole class activities. All aimed to draw out a variety of computational thinking skills/concepts and demonstrate that they are not used in isolation. We hoped both to give a deeper understanding of those concepts and give ideas for new ways to teach them. After each activity a summary was given of the computational thinking skills that had been covered. A fifth 2 hour workshop used a selection of the same activities and was given to trainee primary school teachers, rather than current practicing teachers.

The workshops were run independently and different groups of teachers attended each, though there was some overlap. Full descriptions of the workshops and activities are available via Teaching London Computing (teachinglondoncomputing.org).

2.1 Workshop 1: Computational Thinking

The first workshop consisted of 4 activities in two groups. The first two followed the 'story' given in [2]. It uses the context of helping someone with locked-in syndrome (total paralysis due to a stroke) to communicate. The first activity involves the audience pairing up and communicating by blinking. One person says the letters of the alphabet. The other blinks when they get to the letter they wish to communicate. The audience are asked to think of improvements as well as details of this algorithm that need further thought to make it work. This illustrates algorithmic thinking. The audience suggest things like predictive texting and frequency analysis as improvements, illustrating the concept of generalisation in the sense of transforming solutions between domains. The concept of abstraction is illustrated when discussing the efficiency of the algorithm used, in that the analysis is done in terms of operations rather than time. The importance of evaluation is discussed in terms of an algorithm's functionality, its performance and its usability.

The second activity involves playing a game of 20 questions to illustrate how a a divide and conquer approach leads to much faster algorithms. This illustrates the concept of decomposition in the divide and conquer solution, generalisation in taking a solution from one domain (a child's game) and applying it to this new area, and more on evaluation of performance. As a whole these two activities tell a contextually rich story that draws out how a wide range of computational thinking skills are used in an integrated way to solve problems – even when the solution does not include technology.

In the second half we do a card trick: "The Australian Magician's Dream". A volunteer cuts the pack, then after repeatedly dealing out cards and throwing away one pile each time we are left with a single card. Despite no one being able to predict in advance what it will be, the card is the same as one in a sealed envelope.

It is a self-working trick – an algorithmic trick. Revealing how it works leads to a discussion of what an algorithm is, and how just as programmers write instructions that always give a desired effect whatever happens, so do magicians. The main difference is that one set of instructions is for a computer to follow and the other for a magician. A diagrammatic proof of how the trick is done demonstrates algorithmic thinking, and also abstraction. A final activity, demonstrates how the same algorithm applied to a binary encoding on a punch card can be used to search for any card. This is another example of generalisation as the same algorithm is used in the trick and for searching for data, (though one is a sequential algorithm and the other a parallel version).

2.2 Workshop 2: Algorithmic Thinking

The second workshop covered three activities based on a game, a magic trick and a puzzle, all linked to algorithmic thinking. The first, the 'Intelligent Piece of Paper' activity, involves making a claim that a particular piece of paper is more intelligent than anyone there. After a discussion of whether paper can be intelligent and what intelligence is, it is revealed that the paper plays Noughts and Crosses and has never lost. Two volunteers play a game, one doing what the paper tells them based on the instructions written on it and the other similarly following the audience's instructions. Often the paper wins by forking its opponent. This leads to a discussion of what an algorithm is, how the paper contains an algorithm for playing perfect noughts and crosses, and how the algorithm was created. It introduces algorithmic thinking, logical thinking in coming up with the rules, abstraction in the way the rules are written, and the importance of evaluation in being sure it works.

The second activity uses a magic trick, 'Invisible Palming'. It involves invisibly moving a card from one pile to another. After doing the trick once, the whole audience are split into pairs and given packs of cards. All do the trick in parallel. Even though they do not know how they do it, the trick works! It is a self working trick - an algorithm. It emphasises the idea that algorithmic solutions allow people or computers to achieve things without needing any understanding as long as they follow the algorithm - they could do the trick with no idea of how it worked, just as computers follow algorithms with no idea of what they are doing. This also illustrates decomposition as well as abstraction when looking at the separate parts of the algorithm. It also involves evaluation with respect to functionality and in the sense that it is about solutions working for people. An algorithmic trick will only work if it has a good presentation, just as programs need good interfaces and interaction design.

The final activity illustrates the idea that there can be more than one algorithmic solution to a problem and that different algorithms can be more or less efficient. It uses solitaire style puzzles, moving pieces on a linear board. The audience are asked to solve the puzzle not just in the sense of swapping the pieces' positions, which can be done easily by trial and error, but in the sense of having written an algorithm. They must be able to do the puzzle again by following their instructions without thinking. They are encouraged to look for the fastest possible solution. This demonstrates how evaluation of algorithmic solutions concerns not just functional correctness but also performance.

2.3 Workshop 3: Unplugged Programming

The first activity of workshop 3 involves programming a robot face that makes different expressions depending on sounds the audience make. The unplugged twist is that it is made of audience members holding eyebrows, eyes and a mouth made of card and tubes. They follow instructions for their feature of the face. This demonstrates algorithmic thinking in programming the face, and object-based decomposition and abstraction in the high level instructions used.

This is followed by activities on understanding variables and sequences of assignment statements: this is a critical barrier where novices get stuck. We trace (i.e., follow step-by-step) a swap program swapping colours between variables. The activity involves students holding boxes with name labels to act as variables and coloured paper for the values. It is used to visualise misconceptions. A key point is explaining (and acting out) that the boxes are ones with a shredder and copier included - to deal with copying values and overwriting old values with new ones. The program is acted out step-by-step, moving values about, explaining what is happening and why, as well as encouraging questions about each step. In the process the teachers in the workshop ask lots of questions about their own misconceptions and potential misconceptions of the students, which can be dealt with by acting out a corresponding scenario. This leads on to a pencil and paper dry run test to highlight similar misconceptions that still persist. It draws on the ideas of Dehnadi and Bornat of a way to judge programming potential [7]. We use it instead in a diagnostic way to highlight mental model misconceptions and fix them. This is both a powerful way to identify misconceptions of the teachers and a way to teach them the power of pencil and paper tracing to teach programming.

Finally we act out control structures by compiling a simple if-based program onto audience members who each represent a command or expression and are wired together with rope to represent control flow. An additional person acts as the screen. A tube is used as the program counter to keep track of who should take their action at any point. When the tube returns to the lecturer the program has executed and something appeared on the board (the screen) based on input from the audience (the keyboard).

2.4 Workshop 4: The Human Side

The fourth workshop emphasises the need to understand people in computational thinking. In a card trick, 'The 4 Aces', the audience are told they must keep track of the aces as hands are dealt. A volunteer, who everyone thought had four aces, turns out to have none: the magician has them instead. It shows that algorithmic thinking (in magic and computing) has to take human limitations in to account, in this case our limited focus of attention. A magician designs a system so everyone makes a mistake at the same time, software engineers must use the same 'tricks' to ensure no one makes mistakes. This is reinforced using change blindness images - where something is changed in an image as it flashes, and even though the change is large the audience struggle to see a difference.

This is then put in the context of medical device design and a nurse overlooking a mistake in entering a drug dose: here the device design could have drawn their attention to it or prevented the mistake. These ideas are reinforced by a second trick that involves a magical jigsaw where a robot appears and disappears. This emphasises how a system can be designed to be so complex the human brain cannot take in important information. These activities emphasise that the design and evaluation aspects of computational thinking have to take a human-computer interaction perspective.

Finally we show a video 'Microwave racing' – essentially user testing turned into a game. It illustrates how different interface designs of microwaves make the supposedly simple task of microwaving frozen peas easy or hard. It reinforces how the lessons covered apply to the design of a wide range of computer-based systems.

3. EVALUATION

To evaluate the usefulness of our approach to support teachers we conducted a paper-based post-event survey. 126 people filled out forms (58% female, 42% male) 40 attended workshop 1, 22 workshop 2, 21 workshop 3, 24 workshop 4 and 19 attended the primary computing workshop. A 5-point Likert scale was used ranging from 'strongly disagree' through 'disagree', 'neutral' and 'agree' to 'strongly agree'. The responses were extremely positive throughout.

The first 5 questions were common to all workshops. They asked whether the workshop was: Useful, Interesting, Inspiring, Confidence Building and Fun. For all questions the average rating was well over 4 on the 5 point Likert scale. 94% were positive (agreeing or strongly agreeing) about it being useful, 96% that it was interesting, 92% that it was inspiring, 89% that it was confidence building, and 94% that it was fun. 98% (n=124) were positive that they would recommend the workshop to others. Only 1 person was negative about any of these questions. Similarly 95% (n=122) agreed or strongly agreed that they would use ideas from the workshop in their teaching (though which activities varied).

For workshops 1, 2 and 5 we asked whether as a result of the workshop they now had a better understanding of computational thinking (in general). 89% (n=81) answered positively with an average rating of 4.35. Only one person disagreed. We similarly asked if as a result of the workshop they had new ideas about how to teach computational thinking. The results were similar. 90% (n=77) answered positively with an average rating of 4.31. Only two people disagreed. For workshop 4 on the human side of computational thinking we asked the same two questions, except asking them with respect to "computational thinking: about people". These results were even more positive. 96% (n=24) answered positively (average rating 4.63). No one disagreed. All (n=24) were positive that the workshop had given them new ideas about how to teach the topic (average rating 4.75).

After the second workshop on algorithms, we asked about both improved understanding and whether the workshop gave good ways to teach: what an algorithm is, what a program is and testing/efficiency of algorithms. All but 2 respondents were positive. None were negative.

The third workshop addressed the early stages of novices learning programming, in an unplugged way. The questions were therefore more focussed on programming topics. We asked whether, as a result of the workshop, the teachers better understood: programming, variables and assignment, if statements and flow of control, and compile versus run time. Variables / assignments and compile time versus run time are specific topics we have found novices can struggle to understand. The results were again highly positive. The question about better understanding programming was also asked of those on the primary computing workshop which included the face activity and box variables. Virtually all respondents were positive about each aspect.

We also asked whether the teachers felt they did now have better ways to teach these topics. There were only small differences with the answers overwhelmingly positive.

The first and fourth workshops, though focussing on computational thinking more generally, also covered some specific computing topics around search algorithms, HCI and data structures (so linked to the data representation aspect of computational thinking). We therefore also specifically asked about these. Again the answers were very positive, though two respondents disagreed that their understanding of topics around search algorithms and divide and conquer was improved. The fourth workshop with respect to human-computer interaction was again overwhelmingly positive.

In answer to a question about the best thing about the session many activities are named. Magic tricks (done in all but one session) are named most often. The 'Box Variable' activity explaining variables and assignment and how to overcome misconceptions was particularly popular. Several commented on the unplugged approach that it made things "accessible" or "simple to understand" and that the best thing was "Being able to visually see the task being completed". Several suggested that it had given them practical ideas or emphasised the practical nature of the approach or the way it put computing into a context.

4. CONCLUSIONS

Unplugged activities are not only a powerful way to introduce children to computing concepts, this work shows that they are also a powerful way to introduce these concepts to adult teachers. In particular computational thinking ideas can be successfully introduced in this way.

Teachers attending the workshops in informal conversation afterwards with the lecturer and other members of staff helping with organisation were highly positive. This anecdotal feedback was backed up by the formal survey where the answers were overwhelmingly positive across all workshops. Only a handful of respondents were negative about any aspect. We have thus provided evidence that this approach of using unplugged activities embedded in contextually rich stories is an effective way to explicitly introduce computational thinking ideas as well as more traditional computing topics to teachers themselves, not just to children. However, this was based on their immediate perceptions. Ideally a follow-up study is needed to find how they fared when actually using the activities.

The evaluation suggests that unplugged activities make for an inspiring and fun session for teachers that they also find useful, interesting and confidence building. Confidence building is particularly important, given the challenge facing teachers adapting to a new curriculum that requires them to learn new skills and body of knowledge. As one participant commented on the form about the best thing about the workshop: "realising the approachableness of computer science. It is now less daunting to teach."

5. ACKNOWLEDGMENTS

Teaching London Computing is funded by the Greater London Assembly (LSEFR1094). cs4fn was funded by the EPSRC (EP/F032641/1) with support from Google's CS4HS programme. This paper was written with support from CHI+MED (EPSRC EP/G059063/1). The robot face activity was developed with Quintin Cutts, Steve Measure and Steve Brindley (University of Glasgow). Microwave Racing was created by Dom Furniss (UCL) as part of CHI+MED. Jo Brodie and Chrystie Myketiak also supported this work.

6. REFERENCES

[1] T. Bell. A low-cost high-impact computer science show for family audiences. In *Proceedings of the Australasian Computer Science Conference (ACSC 2000)*, pages 10–16, 2000.

[2] P. Curzon. *Computational Thinking: Searching to Speak.* Queen Mary, University of London. Available from www.teachinglondoncomputing.org, 2013.

[3] P. Curzon. cs4fn and computational thinking unplugged. In M. E. Caspersen, M. Knobelsdorf, and R. Romeike, editors, *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, pages 47–50, November 2013.

[4] P. Curzon, M. Dorling, T. Ng, C. Selby, and J. Woollard. *Developing computational thinking in the classroom: a framework.* Computing at School, Available from community.computingatschool.org.uk/resources/2324, June 2014.

[5] P. Curzon and P. W. McOwan. Engaging with computer science through magic shows. In *Proceedings of ITiCSE 2008, The 13th Annual Conference on Innovation and Technology in Computer Science Education ACM SIGCSE*, pages 179–183. ACM, June 2008. Madrid, Spain.

[6] Q. I. Cutts, M. I. Brown, L. Kemp, and C. Matheson. Enthusing and informing potential computer science students and their teachers. *ACM SIGCSE Bulletin*, 39(3), 2007.

[7] S. Dehnadi and R. Bornat. The camel has two humps, February 2006.

[8] M. Dorling and M. Walker. *Computing Progression Pathways V2.0.* Computing at School, Available from community.computingatschool.org.uk/resources/2324, June 2014.

[9] S. Papert. An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, 1(1):95–123, 1996.

[10] J. M. Wing. Computational thinking. *Communications of the ACM*, 49:33–35, 2006.