Teaching Computer Science to 5-7 year-olds: An initial study with Scratch, Cubelets and unplugged computing

Benjamin Wohl
Highwire DCT
Lancaster University, Lancaster, UK
b.wohl@lancaster.ac.uk

Barry Porter Sarah Clinch
School of Computing & Communications
Lancaster University, Lancaster, UK
b.f.porter | s.clinch@lancaster.ac.uk

ABSTRACT

Changes to school curriculums increasingly require the introduction of computer science concepts to younger children. This practical report compares three existing tools for teaching computer science concepts: unplugged computing, tangible computing and MIT's Scratch. We specifically focus on the use of these tools for school pupils aged 5-7. We describe a comparative study with 28 pupils from three rural UK primary schools that explores engagement with, and effectiveness of, each tool. As far as we are aware this is the first such comparative study of its kind. We demonstrate that the studied tools can be used to successfully introduce core computer science concepts to pupils as young as 5 years of age, that the methods used by teachers to deliver computing curriculums may greatly impact the learning outcomes, and that particular care needs to be taken to ensure that pupils focus on learning concepts rather than learning tools.

Categories and Subject Descriptors

[Social and professional topics] Computing education – *Computer science education*

Keywords

Scratch; unplugged, Cubelets; computing curriculum; primary education; early years; tangible computing

1. INTRODUCTION

In recent years organizations such as Computing at School (CAS) have been advocating for more computer science to be taught in UK schools [3]. In September 2014, revisions to the English National Curriculum introduced computing as a statutory requirement for children at all stages of schooling. Although a positive development, many primary schools are faced with the new challenge of delivering computer science education to young pupils. The new UK Computing Curriculum begins with pupils aged 5–7. However, to date, little research has explored how to teach the complex concepts of computing to children of this age.

WiPSCE '15, November 09 - 11, 2015, London, United Kingdom

ISBN 978-1-4503-3753-3/15/11...$15.00

DOI: http://dx.doi.org/10.1145/2818314.2818340

In this paper, we explore techniques for teaching computing in school to some of the youngest pupils, with particular focus on delivery in small rural primary schools. We compare three different methods for teaching basic concepts of computer science to 5–7 year olds. Our research question is: *"What is the effectiveness of various methods of delivering the UK computing curriculum to key stage one pupils in small rural primary schools?"*

Motivated by the introduction of computing to the UK National Curriculum, the study was conducted in June 2014 (prior to implementation of the new curriculum in September 2014). We targeted three small schools in the North West of England. Within the study, pupils and schools were introduced to three different techniques for learning about computer science: unplugged computing, tangible computing, and Scratch programming. All three techniques have been used in prior work in the classroom or with children, with a range of age groups, but rarely has the focus been on assessing the effectiveness of delivering specific computer science concepts to very young children. As well as a lack of research in the area of teaching computing to this age group, there is particular need to compare various methods of classroom delivery of computational concepts, in order to support the future teaching of these topics [7].

2. RELATED WORK

In recent years a range of new resources have been developed to engage young people in computing. These include tools such as unplugged computing, tangible computing and screen-based environments such as Scratch. However, there is limited study of the effectiveness of these tools with younger pupils, and we are not aware of any research that performs comparisons across all three methods.

2.1 Computer Science Unplugged

Computer Science Unplugged (CS Unplugged) is an approach to teaching computer science where the pupils themselves enact the algorithms. The approach was initially developed at the University of Canterbury in New Zealand, with detailed descriptions of activities that use computational logic in an unplugged environment [4]. These activities range from creating algorithms to counting with binary. Use of CS Unplugged has mainly been studied with older children (high school and middle school) with a focus on pupil interest in the subject of computer science as separate from programming or ICT. However, the impact of this to date has mixed reports in the literature [1, 6].

2.2 Tangible computing and Cubelets

Tangible interfaces have been shown to be effective for teaching complex computation concepts, simulating simple computation processes at a high levl of abstraction [18].

A number of tangible computation devices have been developed to be useable with little or no instruction, often using a system that resembles Lego or stacking blocks that allows the units to interact in a complex manner [10]. Tangible interfaces have been shown to be particularly beneficial for 5–6 years olds engaged in whole-class activities and discussion, although they seemed to be less effective when used in small-group activities [8].

For this study we used a system called Cubelets, which connect using magnets and transmit data and power between the blocks [16]. Cubelets have been used to some extent in the classroom and in settings like science museums; however, there has been limited study of the effectiveness of the Cubelets for teaching computer science [13].

2.3 Scratch

MIT's Scratch programming language launched in 2007. Initially developed for after-school clubs, Scratch allows users to create programs within a semi-object-oriented environment. Scratch aims to be "more tinkerable, more meaningful and more social" than any other programming environment for young people [14]. Scratch has fostered an online community where users can save and share their work online, and where remixing each others' programs is encouraged [11].

Scratch has been shown to be an effective way to introduce programming and teach computer science concepts with pupils aged 13-14 (with a few exceptions) [12].

2.4 Comparative Studies

There have only been a small number of studies which compare methods of delivering computer science concepts to young people.

One of the most relevant studies took place in 2005 and worked with 4-6 year olds using a tangible ubiquitous computer system versus a desktop based game to teach children about various common hazards. This study found that tangible interfaces were significantly more effective at delivering the content [5]. This study also highlighted the difficulty of studying and comparing different environments. Another study in a school context, involving age groups 5–6, 7–8, and 11–12, examined the use of a tangible computing interface and an isomorphic screen-based interface for programming a robot. The study found that the tangible interface was easier to use and more enjoyable for the two youngest age groups, while for the older age group the tangible interface was considered to be more enjoyable but more difficult to use than the screen-based alternative [15].

Finally, outside of an education setting, the study reported in [9] demonstrates that tangible interfaces were also found to be more attractive than screen-based environments for families and young children to program a robot in an informal museum environment.

All of these papers demonstrate the challenges of comparing various delivery methods as it can be difficult to reproduce all aspects needed in one environment versus the other. For example, in all of these examples efforts were made so that the desktop and physical interfaces mirrored each other at least in appearance if not functionality. As far as we are aware ours is the first study to compare techniques for delivering specific computational concepts, rather than using computing interfaces to deliver other teaching outcomes.

3. METHODOLOGY AND APPROACH

This project focused on the three main concepts from the UK computing curriculum: 'algorithms', 'logical prediction' and 'debugging' [17]. Our aim was not to evaluate the value of teaching these concepts to this age group, but rather to study the effectiveness of three different methods of delivery.

As previous work has highlighted that teacher understanding and engagement can be limited [2], in this study we purposefully involved the classroom teachers in all sessions.

3.1 Schools and Participants

We conducted our study in three UK primary schools. The schools were all located in rural Cumbria and each served approximately 70 pupils (min 68, max 74). For the purposes of this study we focused on pupils in "Year 1" (aged 5–6) and in "Year 2" (aged 6–7). Across the three schools, we involved a total of 28 students aged 5–7 years (15 boys, 13 girls). Scratch and Cubelets sessions were conducted in a normal classroom environment and unplugged sessions were conducted in a school hall or outside.

School A. Our sessions at School A were conducted with a group of ten students: five from Year 1 and five from Year 2 (3 boys and 2 girls from each age group). Sessions took place in the morning running from 09:30–11:45, with a half hour break beginning at 10:30, resulting in approximately 1 hour 45 minutes of class time.

School B. At School B the sessions were conducted with the entire cohort of Year 1 pupils, 11 in total (4 boys and 7 girls); no Year 2 pupils participated in the sessions at this school. Sessions took place in the morning running from approximately 09:30–12:00 with a break at 10:30 (i.e. around 2 hours of class time). Pupils in School B referred to extensively using computers to conduct research and produce Publisher and PowerPoint documents.

School C. In School C the sessions were conducted with the entire cohort of Year 2 pupils, 7 in total (5 boys and 2 girls); no Year 1 pupils participated in the sessions at this school. Sessions took place in the afternoon running from approximately 13:15–15:30, with a 15 minute break beginning at 14:30, resulting in around 2 hours of class time.

3.2 Session Execution

All three sessions followed a similar structure, starting with an introduction and reflection either on the pupils' experience with technology (in the first session) or what they remembered from the previous week (in subsequent sessions). There was then a review of the main concepts, followed by an introduction to the tool with a number of short activities and a 'paper model' worksheet, followed by time to complete a challenge with the tools.

For all three activities the challenge involved creating a system which involved two inputs (for example movement and light) and two outputs (for example sound and light). All of the sessions were planned to take between 1 hour 45 minutes and 2 hours, with a break towards the middle of the session. Typically the first hour was facilitator-led learning, with the second half of the session (after the break) allowing the pupils to explore the tool or technique with minimal direction (see link at end of the paper for detailed lesson plans of the sessions). Creating a paper model at the

halfway point provided an opportunity for the pupils to both demonstrate their understanding of the technique and also make prediction about how to solve the challenges.

The sessions were conducted the following order:

School A: Cubelets/Unplugged/Scratch;

School B: Scratch/Cubelets/Unplugged; and

School C: Unplugged/Cubelets/Scratch.

The first session in each school started with a 15 minute introduction to the terms algorithm, logical prediction and debugging. These terms were explained through the pupils helping the facilitator to write an 'algorithm' for brushing their teeth, they then were asked how they would know if they had successfully brushed your teeth (logical prediction). Finally they were asked to close their eyes and go through the algorithm step by step, 'debugging' the algorithm.

Although this entire process was not repeated at the beginning of following sessions the pupils were given a chance to 'refresh' what these concepts meant in subsequent session. They were also given a chance to recall what they had done or learned from the previous session.

3.2.1 Unplugged

In these sessions the pupils were introduced to the concept of 'inputs' and 'outputs' through the game of 'Simon Says'. The pupils were then split into pairs with one person as an output (wearing a blindfold) and the other an input (told to respond to a hand signal from the facilitator). The pupils then sent a signal by holding hands. The pupils were then introduced to the boolean concepts of 'or', 'and', 'xor' and 'not' and with the facilitator created more complex systems which incorporated these concepts. The pupils worked in groups of 4–5 (2 inputs, 1 or 2 logic gates, and one output). To simulate a range of inputs, the facilitator used a range of verbal and visual cues to signal to the 'input' children when to run across to the logic gate. Pupils who were 'outputs' were given a small torch that they turned on or off depending on the signal they received from the logic gate.

3.2.2 Cubelets

In these sessions, after an introduction to the Cubelets system the class was split into two groups. Each group had a similar set of four 'think' Cubelets, 2 'sense' Cubelets, three 'act' Cubelets and one battery. The groups were initially introduced to all of the output (act) and input (sense) Cubelets and allowed to explore different combinations of them. They were then introduced to the 'think' Cubelets and were able to explore how these cubes changed the behaviour of their creations. As the groups had 3 different inputs they could explore what happened when more than one input was paired with a single output. The pupils were then given a short amount of time to explore different combinations in their small groups.

3.2.3 Scratch

In these sessions we used the online version of the Scratch software and the pupils worked in pairs. Before the session the facilitator created a number of generic Scratch accounts opened to a prepared project[1] at the beginning of the session. A select number of Scratch commands were already available in 'workspace' for the pupils to work from. During the Scratch session the pupils were first introduced to the different commands available in Scratch. They were then shown a number of specific commands to move the Scratch cat and allow the sprites to interact.

3.3 Paper Models

Paper models were initially used as a way of assessing the pupils' ability to make logical predictions about algorithms (see Fig. 1). At the halfway point of the session the pupils were asked to use these models to create a representation of what they had already done and also a representation of an algorithm that they had not done. After noting the children's response to the models the research team felt that they could also be used as a proxy for understanding. Details of all paper models are available online for reference[2].

3.4 Data Collection

Data collection from the pupils was in two forms: paper models and interviews. Paper models were used to assess understanding of the tool, both whether the pupils were able to create a model that would work (based on their knowledge) or reproduce one of the examples they had already created. In each session the pupils had approximately 15 minutes to complete the paper model. The models were marked as either Y (for showed understanding), N (did not show understanding) or M (unclear if there was understanding or not). Maybes included where the model was correct but the pupil couldn't explain it or vice-versa, or where it was difficult to determine if understanding had been gained.

During the final half hour of the session the pupils were interviewed in pairs or as a group of three. They were asked to explain the three concepts and also to rate the session for fun on a scale of 1-10. They were then asked 5 qualitative questions: 1.) What do you think you learned from the activity? 2.) Would you like to do this activity again? 3.) How does this compare to the previous weeks activity/ previous experience with technology? 4.) Do you have any questions about this activity or computer programming in general? 5.) What do you want do next on your project? In transcription the answers to questions 1) and 4) were coded as either pertaining to the concept or the tool of each session. Questions 2) and 3) were used as quantitative measures and the response to question 5) was coded as being either relevant to the session or not.

4. FINDINGS AND DATA

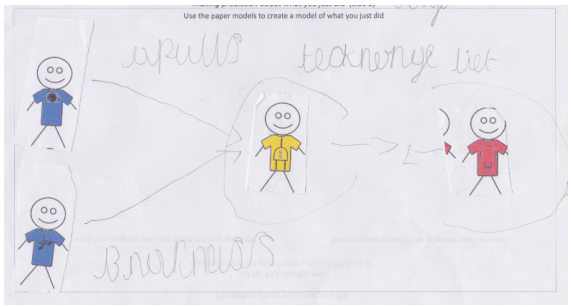
Due to the nature of the study, focusing on small rural primary schools, the sample size was relatively small (28 pupils over 9 sessions). Although qualitative and quantitative data was recorded, a greater weight has been put on the qualitative results in combination with the quantitative findings. The main quantitative measures examined were the understanding based on the paper models and the 'fun' scores given by the pupils for the sessions. The data from the interviews was coded based on the pupils' responses as: answered without prompting (3), answered with some prompting (2) or did not provide adequate answer (1).

4.1 Findings from Interviews (qualitative)

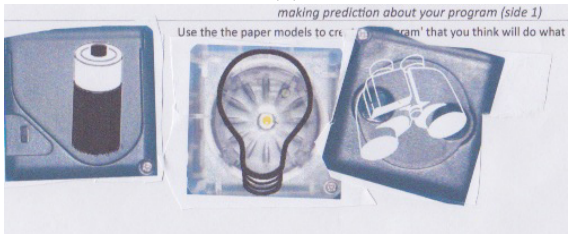
The results of comprehension scores based on coded interviews are shown in Figure 2.

[1]http://scratch.mit.edu/projects/21482378/

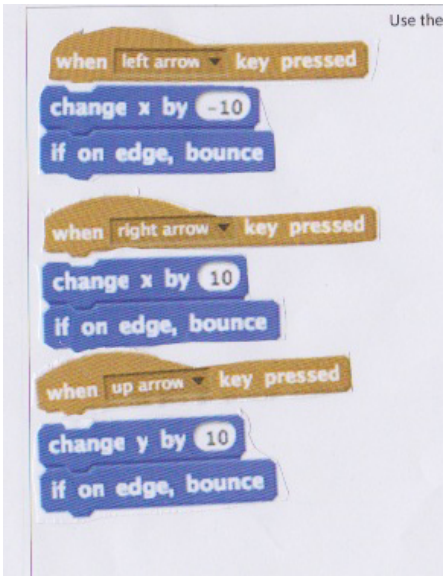
[2]http://dx.doi.org/10.6084/m9.figshare.1381820



(a)



(b)



(c)

Figure 1: Sample paper models for (a) an unplugged session; (b) a Cubelets session; and (c) a Scratch session. In (a), light input and a distance input go to an 'and' gate which will turn on a light. The model in (b) represents a battery cube, light output and a distance sensor (a functioning circuit), whilst that in (c) represents the commands needed to make the 'Scratch Cat' move left, right and up.

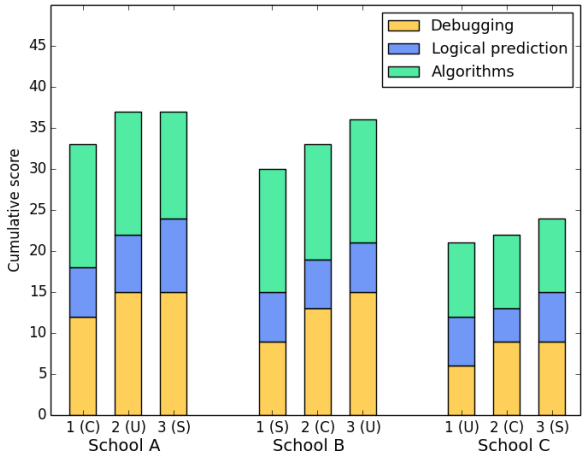


Figure 2: Graph of pupils' understanding, based on coded interviews. The bars are arranged in the order in which the sessions ran in each school. 'C', 'S' and 'U' refer to Cubelets, Scratch and unplugged respectively. The size of a bar segment is calculated by summing the values of the coded interview responses from each student, where 3 means full understanding of that concept and 1 means no understanding. For School A and B, the maximum possible value for each bar segment is 15, while for School C this is 9 (with fewer participants).

In School A, all interviews showed an understanding of the concept of an algorithm. Understanding of logical predictions increases slightly over time, though by the final session it remains low compared to the other two concepts. Understanding of debugging also increases slightly and finishes in line with the understanding of algorithms.

With School B, all pupils could define an algorithm after the first session, though a single outlier seems to have lost this comprehension after session two. School B showed no increase in understanding of the concept of logical prediction, but does show a consistent increase in understanding of debugging. This was, however, at a slower rate than the other schools such that it was not until the final session that the whole group had a full understanding of debugging.

In School C, all interviews showed an understanding of algorithms from the first session and throughout the subsequent sessions. In terms of the logical prediction concept, one interview demonstrates a loss of understanding in the second session (Cubelets) before rising again in the final session. Similarly to the other schools, understanding of debugging increases steadily over time, matching that of algorithms by the end.

The qualitative questions also revealed that, depending on the session, the pupils were more inclined to focus on either the concept or the tool, and that some sessions elicited more relevant questions and next steps than others. When asked "what do you think you learned", out of all 38 interviews pupils primarily answered with 'tool'-based answers after Scratch and with 'concept'-based answers after unplugged; Cubelets sessions meanwhile yielded an even mixture of concept- and tool-based answers.

Finally, we note that the order in which the sessions were carried out does not appear to have affected overall concept understanding as measured by interviews.

4.2 Findings from paper models and fun scores (quantitative)

The results for pupil comprehension based on their completion of paper models are summarised in Figure 3. Although the understanding of concepts here for the most part shows a progression from session to session, there is no observable relationship in understanding based on different session orders. School B seemed to grasp the paper models exercise more readily than the other schools; although this may relate to the session order it is possible that it involves

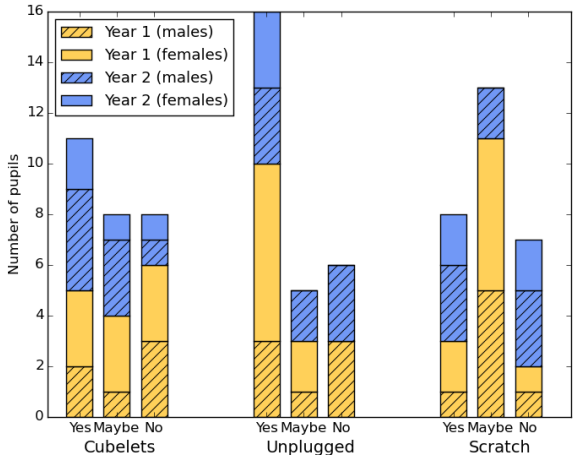


Figure 3: Graph of pupils' concept understanding, based on paper models.

other factors such as using computers more in their regular lessons. Across all schools the paper models revealed the highest level of understanding after the unplugged session.

Breaking down this same data by year group there are some interesting trends that emerge. Among year one pupils (aged 5 and 6) there is a high number of 'maybes' during the Scratch session. This may be due to the greater challenge experienced by the younger students with cutting and pasting the more detailed Scratch elements. Among the year one pupils there is also a strong pattern of greater understanding in the unplugged session. Among year two pupils (aged 6-7) in Scratch the pupils separated evenly into either understanding or not understanding. Half of these older pupils showed understanding with Cubelets and unplugged with a higher number of maybes for Cubelets. In general the year two pupils showed more understanding through the paper models than the year ones; this may indicate that this is a better method of assessing understanding with the older pupils. Although there is not much that can be gained by looking closely at the breakdown of the data by gender, the very strong increase of understanding based on the unplugged paper models among girls is of note.

Our main observation from the 'fun scores' is that they generally *decreased* over time, presumably due to a decrease in novelty factor of the sessions. Within this general trend it is then interesting to observe that when starting with 'Scratch', the following session (Cubelets) was ranked as more fun. There is also a notable difference for unplugged when it was used first (9.7 highest score recorded) to when it was the last session (6.36 lowest score recorded). Here, it seems when unplugged is put in context of the more 'technological' sessions it is viewed as being less exciting. It can be hypothesised from this that working with the technology carries its own excitement and incentive, so for the children moving from technology to 'unplugged' (effectiveness of delivering concepts aside) feels like a step backwards.

5. DISCUSSION AND ANALYSIS

5.1 Delivery of Key Concepts

Algorithm: the pupils understood this concept easily after the initial introduction. When related to instructions (i.e. an algorithm as a list or series of instructions) this concept related to their everyday lives. The pupils found this concept most difficult when it was only related to 'Scratch'; although Scratch is a very capable programming language this may therefore make the delivery of basic concepts more difficult.

Making logical predictions: This was without a doubt the hardest concept for the young people to understand. In most cases even with prompting the pupils were unable to define this concept. Some pupils seemed to be beginning to understand this concept after having Scratch as a third session. This may be because the abstraction that occurs in Scratch makes 'making predictions' an implicit part of the process. Cubelets on the other hand seem to encourage a 'learn through play/exploration' response; this is highly engaging but does not encourage 'making predictions'.

Debugging: In School A and School C, which started with Cubelets and unplugged respectively, the pupils were able to understand the concept of debugging generally after the second session. When starting with Scratch the pupils took an additional session to build up a good understanding of debugging. Although this concept was new to the pupils and did not readily relate to any concept that they were already familiar with, the physicality of 'something going wrong' in the unplugged or Cubelet session allowed the pupils to quickly build up an understanding. Often in Scratch the errors they produced meant the program just did not work (made nothing happen) whereas with unplugged and Cubelets the debugging involved the unexpected happening and the pupils figuring out why.

5.2 Learning coded on concepts vs. tools

Based on the coded observations there were two clear trends that show the value of the different tools. After the unplugged session the pupils were most engaged in the concepts of computer science. The unplugged session seemed to demonstrate that young children can be introduced to and engaged in relatively complex ideas. However after the Scratch session the pupils had the most relevant ideas for things to try next. Although Scratch was one of the most difficult tools to implement as a teaching method (and in our experience proved to be the most intimidating method for teachers) it seemed to uniquely engage the pupils' creativity.

5.3 Observations based on paper models

The creation of the paper models delivered valuable insights and created a useful break in the session. However, motor skills of younger pupils impaired the creation of these models – the youngest children found the cutting and pasting more difficult and tended to need more than the 15 minutes provided to finish their paper model. Using the paper modeling exercise as a proxy for a level of understanding about the workings of the tool revealed a number of insights. In particular, categorisation of the paper models for learning levels suggested that for girls and younger pupils, the unplugged session was particularly effective in promoting understanding of the concepts.

5.4 Pupil preference

Based on observations and teacher feedback the pupils appeared to enjoy all sessions equally and at the beginning of each session were excited to find out what would be happening. However, the interviews did highlight a difference here such that the strongest determinant of the 'fun score' was the session order. The level of 'technology' involved in the session was also a strong determining factor; when it was not the first session many pupils did not rank the un-

plugged session as particularly 'fun'. The pupils also seemed to greatly enjoy the physicality of the Cubelets, relating to them much like toys. In fact the toy-like presentation of the Cubelets may well have made them seem more like fun and less like learning. Pupils' previous experience/capability for using a computer (mouse and keyboard) seemed to impact on their enjoyment of Scratch. When directly asked to pick a favourite session the pupils tended to pick Cubelets over the other sessions; when Cubelets was not an option their preference was Scratch.

6. CONCLUSION

This paper has presented an initial comparative study of three different methods of teaching computer science concepts to ages 5–7: Scratch, Cubelets and unplugged. Student comprehension was measured by coded interviews and by analysis of paper models. Overall, unplugged appeared to generate the highest level of understanding of the concepts of algorithms, logical predictions and debugging; while Cubelets proved one of the most engaging methods; and Scratch generated the most 'tool'-based questions.

The order in which these methods were used does not show a particularly high impact on comprehension in this study, though using Scratch first may have had a slight negative effect. In terms of 'fun' ratings we observe a general downward trend as each method is introduced to the same group, with the most notable negative effect observed in using unplugged as the last teaching method.

In future work we hope to expand the scale of our study and also apply more rigorous tests of the causes of higher or lower student comprehension in relation to the different levels of technology to which they are introduced.

7. ACKNOWLEDGMENTS

This project was funded through the HighWire CDT which is part of the UKREs digital economy strand, and funded through the EPSRC. Lesson plans, evaluation tools and raw anonymised data are available to download at: http://dx.doi.org/10.6084/m9.figshare.1381820.

8. REFERENCES

[1] T. Bell, J. Alexander, I. Freeman, and M. Grimley. Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology*, 13(1):20–29, 2009.

[2] H. Bort and D. Brylow. CS4Impact: Measuring computational thinking concepts present in CS4HS participant lesson plans. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE '13, pages 427–432, New York, NY, USA, 2013. ACM.

[3] T. Crick and S. Sentance. Computing at school: Stimulating computing education in the UK. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, Koli Calling '11, pages 122–123, New York, NY, USA, 2011. ACM.

[4] CSUnplugged.org. Computer Science Unplugged. http://csunplugged.org/ [Accessed 14 Aug. 2014], 2014.

[5] J. A. Fails, A. Druin, M. L. Guha, G. Chipman, S. Simms, and W. Churaman. Child's play: A comparison of desktop and physical interactive environments. In *Proceedings of the 2005 Conference on Interaction Design and Children*, IDC '05, pages 48–55, New York, NY, USA, 2005. ACM.

[6] Y. Feaster, L. Segars, S. K. Wahba, and J. O. Hallstrom. Teaching CS Unplugged in the high school (with limited success). In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, pages 248–252, New York, NY, USA, 2011. ACM.

[7] G. Fessakis, E. Gouli, and E. Mavroudi. Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Comput. Educ.*, 63:87–97, Apr. 2013.

[8] M. S. Horn, R. J. Crouser, and M. U. Bers. Tangible interaction and learning: The case for a hybrid approach. *Personal Ubiquitous Comput.*, 16(4):379–389, Apr. 2012.

[9] M. S. Horn, E. T. Solovey, R. J. Crouser, and R. J. Jacob. Comparing the use of tangible and graphical programming languages for informal science education. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 975–984, New York, NY, USA, 2009. ACM.

[10] B. T. Kirby, M. Ashley-Rollman, and S. C. Goldstein. Blinky blocks: A physical ensemble programming platform. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '11, pages 1111–1116, New York, NY, USA, 2011. ACM.

[11] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. The Scratch programming language and environment. *Trans. Comput. Educ.*, 10(4):16:1–16:15, Nov. 2010.

[12] O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari. Learning computer science concepts with Scratch. *Computer Science Education*, 23(3):239–264, 2013.

[13] S. Pruchnicky. Cubelets and inquiry based learning by stepan pruchnicky, the first FE contest winner post. http://flexibilityenvelope.com/cubelets-and-inquiry-based-learning-by-stepan-pruchnicky-the-first-fe-contest-winner-post [Accessed 14 Aug 2014], 2012.

[14] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. Scratch: Programming for all. *Communications of the ACM*, 52(11):60–67, Nov. 2009.

[15] T. Sapounidis and S. Demetriadis. Tangible versus graphical user interfaces for robot programming: Exploring cross-age children's preferences. *Personal Ubiquitous Comput.*, 17(8):1775–1786, Dec. 2013.

[16] E. Schweikardt. Modular robotics studio. In *Proceedings of the 5th International Conference on Tangible and Embedded Interaction 2011, Funchal, Madeira, Portugal, January 22-26, 2011*, pages 353–356, 2011.

[17] UK Department of Education. The National Curriculum in England Framework Document. Technical report, London, 2013.

[18] J. M. Wing. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881):3717–3725, 2008.