K-5 Teachers' Uses of Levels of Abstraction Focusing on Design

Jane Waite
Queen Mary University of London
London
j.l.waite@qmul.ac.uk

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

William Marsh
Queen Mary University of London
London
d.w.marsh@qmul.ac.uk

Sue Sentance
King's College London
London
sue.sentance@kcl.ac.uk

ABSTRACT

Recent research with middle school and university students highlights two factors that contribute to programming success: 1) understanding the level of abstraction that you are working at, and 2) being able to move between levels. In this qualitative study we explored levels of abstraction, and particularly the design level, with five K-5 teachers. Here we outline 11 main findings. The teachers interviewed use the design level for both programming and writing. However, the two expert computing teachers have a far greater depth of understanding of the opportunities for the use of the design level, supporting pupils to understand the level they are working at and helping them move between levels of abstraction by using designs in novel ways. Further work is needed to investigate whether our results are generalisable. Further exploration of levels of abstraction and particularly how the design level helps K-5 learners learn to program, in the same way that planning supports novices learning to write, is warranted.

ACM Reference format:
Jane Waite, Paul Curzon, William Marsh, and Sue Sentance. 2017. K-5 Teachers' Uses of Levels of Abstraction Focusing on Design. In *Proceedings of WiPSCE '17, Nijmegen, Netherlands, November 8–10, 2017,* 2 pages.
https://doi.org/10.1145/3137065.3137068

1 INTRODUCTION

Despite a lack of consensus on exactly what computational thinking is, proponents of computational thinking, surveys of computational thinking, and emerging curriculum frameworks propose that abstraction forms a core component [5, 10, 15] with ambitious potential and possibly exaggerated claims [12]. We investigate K-5 teachers use of abstraction through the lens of one scenario of abstraction: that of the levels of abstraction hierarchy, particularly focusing on the design level.

WiPSCE '17, November 8–10, 2017, Nijmegen, Netherlands

ACM ISBN 978-1-4503-5428-8/17/11…$15.00
https://doi.org/10.1145/3137065.3137068

2 RELATED WORK

Perrenet, Groote and Kassenbrood [8] proposed a levels of abstraction hierarchy explaining university students' thinking about algorithms. Armoni [1] suggested a framework for its use and to support understanding renamed the object level the algorithm level. Statter & Armoni [11] reported promising findings on using the hierarchy in middle school programming. In earlier work [14] we aligned the hierarchy with the work of others and renamed the object level to the design level to support K-5 teachers understanding naming the levels: problem, design, code and running the code.

3 AIMS AND APPROACH

Our aim was to better understand the opportunities for use of the levels of abstraction hierarchy, particularly the design level, with K-5 teachers. Using semi-structured interviews augmented with unplugged activities we conducted in-depth interviews with five K-5 teachers. A thematic qualitative data analysis approach was used to analyse the transcriptions [6].

4 FINDINGS & DISCUSSION

We outline *eleven* of our most interesting findings. These are loosely grouped by Magnusson, Krajcik & Borko's **PCK elements** [7].

Goals & Objectives: *Use of terms:* Our five teachers used a variety of conflicting terms for design, algorithm and code, and had a limited vocabulary to describe running the code. The terms algorithm and code, used as labels for different levels of abstraction, were used interchangeably. Progression is aligned to learners building understanding based on precise vocabulary [2, 9]. Teachers understanding of levels of abstraction may be limited and pupils' progression in programming may be being compromised by a lack of teachers' shared understanding. *Level of detail:* Both novice and expert teachers mentioned the level of detail included, or omitted, by pupils at the design level in literacy and in programming. One expert teacher explained that he demonstrated to pupils how to include 'less detail' in computing design than might be expected in plans for other subjects. Teachers' understanding of the amount of detail needed for each level may impact on teaching and learning of levels of abstraction. *Familiarity with and using different design types:* All teachers showed familiarity, confidence and a depth of understanding on the use of the design level in a variety of subjects, such as Maths, Music and History. Teachers cited storyboards as being good for sequencing, and concept maps being flexible to add new ideas. All teachers said they used storyboards in English and

labelled diagrams in Science. Concept maps, labelled diagrams and storyboards were mentioned as being used across a wide range of subjects including computing. The expert computing teachers showed a more in-depth understanding of the design level of abstraction for computing. They explained how storyboards were good for animations as they show sequence clearly, and how mind maps were better for games as common features can be easily shown. How different design types exemplify the level of abstraction at which one is working at, or are effective to support transition, requires further investigation.

Students' understanding: *Synergy with teaching writing & Self regulation:* Novice and expert computing teachers, mentioned the importance of, and utility of, using planning, a form of design, to support self-regulation for novice writers as they learn to write. Self-regulation, planning, revising and editing compositions is recommended as having high impact on improvement of writing[3, 4]. Novice writers use their plan to focus on one part of it at a time, transition levels of abstraction as they write that part, revise and edit, and check back to the plan. Similarly, a design in programming becomes a personalised scaffolding map that supports traversing the levels of abstraction as learners decompose their problem, implement each component, debug each part as they run the code and check back to the problem level, before moving onto the next component of their design.

Instructional Strategies: *Aide memoire:* All teachers mentioned the role of a design in managing the process of pupils progressing their programming project. Design served as an aide memoire of what was to be done, what had been done and what to do next. *Completeness, coverage, cohesion:* One teacher drew attention to the use of design to help pupils finish work improving completeness, another to the design helping pupils check coverage, and another to how design kept pupils on track improving cohesion. Here the quality of the finished piece is improved by using the design level to manage a complex task. There are links to cognitive load theory [13] which asserts that if a complex scenario is broken down into discrete units then the complex task becomes easier to solve. *Annotation:* The expert teachers mentioned adding notes, such as code constructs, to designs, both before and after coding had started. Annotations not only transitioned the levels of abstraction, but were also used for differentiation and to provide a record of the original and changed ideas supporting a growth mindset. *Pair programming:* One expert teacher raised the idea of using a design as a contract between pupils when working in pairs. *What next?:* An expert teacher also pointed out design was useful for him as well as pupils, as he could see what was needed to be taught to implement ideas planned. Each strategy may draw attention to the level of abstraction one is working at or support transition across levels. However, further investigation is needed to explore this.

Assessment: *Do-ability:* The expert teachers required pupils to consider 'do-ability'.'Do-ability' is understanding whether one can, at one's current and anticipated level of experience and within the time frame of a project, implement a design, within the constraints of the programming language being used. *Self-assessment:* The expert teachers required pupils to mark their design with a self-assessment of their confidence to implement its components. Both these assessment activities straddle the design and code level.

5 CONCLUSIONS

Despite a limited population of participants, our findings suggest that the levels of abstraction hierarchy may be useful for reviewing pedagogy for programming. Our novice and expert teachers situate work at the design level in programming as well as in other subjects. Our expert teachers use design in novel and interesting ways including using it to facilitate movement across the levels of abstraction. However, our findings require further investigation to assess whether they are generalisable. We suggest there is particular merit in investigating the use of design as a self-regulation tool to develop independence for novice and struggling programmers in the same way that planning is used to support novice and struggling writers.

6 FURTHER WORK

We plan to explore in more detail the relationship between abstraction, design and levels of abstraction. Our next steps also include: further literature review and survey of experts; a review of curricula material for incorporation of design and other levels of abstraction; a survey of teachers to verify the findings presented here with a wider audience; work with a focus group of teachers to create guidance on the practical application of the levels of abstraction particularly the design level, in K-5 programming teaching.

REFERENCES

[1] Michal Armoni. 2013. On Teaching Abstraction in Computer Science to Novices. *Journal of Computers in Mathematics and Science Teaching* 32, 3 (2013), 265–284.

[2] Joanne Carlisle, Jane Fleming, and Beth Gudbrandsen. 2000. Incidental word learning in science classes. *Contemporary Educational Psychology* 25, 2 (2000), 184–211.

[3] Steve Graham, Alisha Bollinger, C Olson, Catherine DÁoust, Charles MacArthur, Deborah McCutchen, and Natalie Olinghouse. 2012. Teaching elementary school students to be effective writers. *What Works Clearinghouse, US Department of Education* (2012).

[4] Steve Higgins, Maria Katsipataki, Dimitra Kokotsaki, Robert Coe, Lee Elliot Major, and Robbie Coleman. 2013. The Sutton Trust-Education Endowment Foundation Teaching and Learning Toolkit: Technical Appendices. *Education Endowment Foundation, London* (2013).

[5] Jeff Kramer. 2007. Is abstraction the key to computing? *Commun. ACM* 50, 4 (2007), 36–42. https://doi.org/10.1145/1232743.1232745

[6] Udo Kuckartz. 2014. *Qualitative text analysis: A guide to methods, practice and using software.* Sage.

[7] Shirley Magnusson, Joseph Krajcik, and Hilda Borko. 1999. Nature, sources, and development of pedagogical content knowledge for science teaching. In *Examining pedagogical content knowledge.* Springer, 95–132.

[8] Jacob Perrenet, Jan Friso Groote, and Eric Kaasenbrood. 2005. Exploring students' understanding of the concept of algorithm: levels of abstraction. *ACM SIGCSE Bulletin* 37, 3 (2005), 64–68. https://doi.org/10.1145/1067445.1067467

[9] Edward Sapir. 1921. An introduction to the study of speech. *Language* (1921).

[10] Deborah Seehorn, Tammy Pirmann, Todd Lash, Letici Batista, Dylan Ryder, Vicky Sedgwich, Irene Lee, Dianne OGrady-Cunniff, Bryan Twarej, Daniel Mox, Julia Bell, Laura Blankenship, Lori Pollock, and Uche Chinma. 2016. Interim CSTA K-12 Computer Science Standards. (2016).

[11] David Statter and Michal Armoni. 2016. Teaching Abstract Thinking in Introduction to Computer Science for 7th Graders. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education.* ACM, 80–83.

[12] Matti Tedre and Peter J Denning. 2016. The long quest for computational thinking. In *Proceedings of the 16th Koli Calling Conference on Computing Education Research.* 24–27.

[13] Jeroen JG Van Merrienboer and John Sweller. 2005. Cognitive load theory and complex learning: Recent developments and future directions. *Educational psychology review* 17, 2 (2005), 147–177.

[14] Jane Waite, Paul Curzon, William Marsh, and Sue Sentance. 2016. Abstraction and common classroom activities. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education.* ACM, 112–113.

[15] Jeannette M Wing and Valerie Barr. 2011. {Jeannette M. Wing@ PCAST; Barbara Liskov Keynote}. *Commun. ACM* 54, 9 (2011), 10–11.