Abstraction and common classroom activities

Jane Waite Paul Curzon, William Marsh
Queen Mary University of London
Mile End Road
London
044 020 7882 5555
{j.l.waite p.curzon d.w.r.marsh}@qmul.ac.uk

Sue Sentance
King's College London
Stamford Street
London
044 020 7836 5454
sue.sentance@kcl.ac.uk

ABSTRACT
In popularizing computational thinking, Wing notes that 'abstraction is described as underlying computational thinking and computational thinking is described as fundamental to computing.' Emerging curricular now require educators to incorporate computational thinking and abstraction into their teaching. Many refer to Piaget's work as evidence of an age-related ceiling preventing younger pupils from being able to abstract. However, more recent evidence suggests that pupils use elements of abstraction in their general process of learning, and that the skill of abstraction can be explicitly taught. We draw on personal classroom experience to illustrate the points made in the literature. Common classroom activities such as using labelled diagrams, concept maps and storyboards are aligned to features of abstraction. We argue that abstraction can and should be taught to young pupils.

CCS Concepts
• Social and professional topics~Computational thinking • Social and professional topics~K-12 education

Keywords
Abstraction, visualisation, storyboard, graph, design.

1. COMPUTATIONAL THINKING AND ABSTRACTION
The concept of abstraction within computer science is, as with computational thinking, not new. Those learning to solve problems using computers, are required to abstract as they develop designs at different levels of detail, write programs and run code; algorithms and programs themselves are abstractions [17, 21].

Abstraction, has been widely accepted to be included as a component, if not the cornerstone, of computational thinking [8, 12, 20, 21] and there appears to be general consensus that abstraction relates to hiding detail, and removing unnecessary complexity with respect to the problem at hand [4]. Some also include generalisation, such as Barr and Stephenson's definition 'abstraction - simplifying from the concrete to the general as solutions are developed' [2].

How abstraction relates to other computational thinking terms is sometimes mentioned in emerging lists of what abstraction might look like in class [2, 4, 5], and curriculum material is starting to be developed that suggest lists of behaviours and activities that might demonstrate progression in abstraction [4, 9].

WiPSCE '16, October 13-15, 2016, Münster, Germany
ACM 978-1-4503-4223-0/16/10.
DOI:http://dx.doi.org/10.1145/2978249.2978272

2. ABSTRACTION CEILING OR PROGRESSION IN ABSTRACTION?
Piaget's seminal educational theories on learning, written some thirty years ago are often referenced as evidence of an age-related developmental ceiling for abstraction [13, 14]. However, Piaget's, rarely cited, late work, 'Recherches sur l 'abstraction réfléchissante', first translated to English in 2001, seems to counter these claims [16]. Campbell translates key terms used by Piaget to describe abstraction as empirical abstraction, projection, reflected abstraction and meta reflection. Piaget, describes abstraction as a spiral of learning 'without end and especially without an absolute beginning' (page 306 of [16]) not bound by operational stages. He also details tests of children's ability to abstract from 18 months.

Recently, Syslo & Kwiatkowska concluded 5 to 12 year olds 'demonstrate that they are capable to work with abstraction' [18]. Similarly, Gibson showed he could introduce children to abstraction rich computer science material normally associated with older pupils [11]. Curzon *et al*'s unplugged cs4fn activities, taught to younger pupils, include teaching abstraction [6].

3. LEVELS OF ABSTRACTION
Armoni states 'there are no validated tools that assess abstraction ability'[1]. However, she points to the PGK model's levels of abstraction 1. Execution, 2. Program, 3. Object (redefined to Algorithm by Armoni), 4. Problem [1, 15] and defines sub-skills , simplified here as being able to say what level you are working at; move between levels; say what level you should be using; add more detail, or remove detail as needed within a level

In a similar vein, Cutts *et al* assign abstraction to levels, when investigating how students were asked to express answers to questions. The levels were 1. English, 2. CS Speak, 3. Code [7]. They found that much assessment is focused on levels 2 and 3.

Taub *et al*'s study considered how abstraction might be taught through physics games, concluding that students, in terms of computer science learning, moved from the high level of what the simulation should do to the low level of how it is done [19].

In order to situate classroom activities, we combine and add to the work of these authors to create a simple framework of the levels of abstraction for problem solving in programming projects. 1. Problem – English – What is needed 2. Algorithm – CS Speak – What it should do. 3. Program – Code - How it is done. 4. Runtime – Results – What it does.

4. CLASSROOM ACTIVITIES
We write now based not on empirical work but on the experience of one of the paper's authors, as a primary teacher. We discuss several classroom activities, each in some way a summary, an

abstraction. We ask how each might be investigated, in a sociocultural situated framework, with pupils, teachers, toolset and environment and so outline a programme of work to investigate abstraction in a primary school context.

Labelled diagrams: Across the curriculum, from emergent writing onwards, the labelled diagram provides pupils with the opportunity to show what is most important about an object, process or system. Teachers exemplify what can be ignored as they provide model examples, and pupils demonstrate understanding in independent work. Can Piaget's late work on abstraction be aligned to Solo [3] or Blooms [10] to aid teachers understanding of pupils' progression in abstraction as they use labelled diagrams?

Concept maps: These visualisations are used across the curriculum to teach classification and grouping. The property used to create a new group implies it is important. Do teachers point this out? Are pupils given opportunities to work out new classifications and groupings and compare their ideas to that of others, across different scenarios? Would doing so provide a foundation for understanding levels of abstraction in programming concepts.

Storyboards: As pupils plan writing they must decide what must be included, and what they can ignore. Teachers provide examples that children first copy, but with experience they adapt and invent. How might the storyboard format, and how storyboards are used, impact progression of abstraction? When writing a story, the brief is the problem, the storyboard like an algorithm and the story itself like a program, conforming to syntax and grammar rules, debugged at write time. Do teachers draw attention to the level of abstraction being used during use?

5. CONCLUSION

As well as the classroom activities above there are numerous situations where pupils are asked to summarise or create and then work from a plan. Similarly, as pupils embark on programming projects perhaps these visualisations can be used at various levels of abstraction. There appear to be rich far reaching opportunities to utilise planning and summarising visualisations to make progression in abstracting and to work across the levels of abstraction. However, evidence is limited and research is needed to determine the potential of, and effectiveness of such approaches.

6. REFERENCES

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

[2] Barr, V. and Stephenson, C. 2011. Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community? *ACM Inroads*. 2, 1 (2011), 48–54.

[3] Biggs, J. and Collis, K. 1982. Origin and description of the SOLO taxonomy. *Evaluating the quality of learning: The SOLO Taxonomy. New York: Academic Press Inc*. 17–30.

[4] Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. and Woollard, J. 2015. Computational Thinking a Guide for Teachers. Retrieved May 5, 2016 http://community.computingatschool.org.uk/files/6695/original.pdf

[5] CSTA 2011. Computational Thinking Teacher Resources 2nd Edition. (2011). Retrieved May 5, 2016 from http://csta.acm.org/Curriculum/sub/CurrFiles/472.11CTTeacherResources_2ed-SP-vF.pdf

[6] Curzon, P., McOwan, P.W. and Plant, N. 2014. Introducing teachers to computational thinking using unplugged storytelling. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (2014), 89–92.

[7] Cutts, Q., Esper, S.,Fecho, M., Foster, S.R. and Beth, S. 2012. The abstraction transition taxonomy: developing desired learning outcomes through the lens of situated cognition. *Proceedings of the ninth annual international conference on International computing education research* (2012), 63–70.

[8] DfE 2013. *Computing programmes of study key stages 1 and 2 National Curriculum in England*. Department of Education. Retrieved May 5, 2016 from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study.

[9] Dorling, M. and Walker, M. 2015. Computing Progression Pathways. (2015). Retrieved May 5, 2016 from http://community.computingatschool.org.uk/files/5098/original.xlsx

[10] Fuller, U., Johnson, C.G., Ahoniemi, T., Cukierman, D., Hernan-Losada, I., Jackova, J., Lahtinen, E., Lewis, T. L., Thompson, D. M. Riedesel, C. and Thompson, E. 2007. Developing a computer science-specific learning taxonomy. *ACM SIGCSE Bulletin* (2007), 152–170.

[11] Gibson, J.P. 2012. Teaching graph algorithms to children of all ages. *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education* (2012), 34–39.

[12] Grover, S. and Pea, R. 2013. Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*. 42, 1 (2013), 38–43.

[13] Kramer, J. 2007. Is abstraction the key to computing? *Communications of the ACM*. 50, 4 (2007), 36–42.

[14] Lister, R. 2011. Concrete and other neo-Piagetian forms of reasoning in the novice programmer. *Proceedings of the Thirteenth Australasian Computing Education Conference-Volume 114* (2011), 9–18.

[15] Perrenet, J.C., Groote, J.F. and Kaasenbrood. E. 2005. Exploring students' understanding of the concept of algorithm: levels of abstraction. *ACM SIGCSE Bulletin*. 37, 3 (2005), 64–68.

[16] Piaget, J. and Campell, R.L. 2001. *Studies in reflecting abstraction*. Psychology Press.

[17] Schwill, A. and Universität Paderborn. 1994. Fundamental ideas of computer science. *Bulletin-European Association for Theoretical Computer Science*. 53, (1994), 274–274.

[18] Syslo, M.M. and Kwiatkowska, A.B. 2014. Playing with computing at a children's university. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (2014), 104–107.

[19] Taub, R., Armoni, M. and Ben-Ari, M. 2014. Abstraction as a bridging concept between computer science and physics. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (2014), 16–19.

[20] The College Board 2016. *AP Computer Science Principles Curriculum Framework 2016-2017*. The College Board. Retrieved May 5, 2016 from https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-principles-course-and-exam-description.pdf

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