Evidence of Assessing Computational Thinking

Cynthia Selby
University of Southampton
Highfield
Southampton UK
C.Selby@
southampton.ac.uk

Mark Dorling
Computing At School
BCS, Chartered Institute for IT
Swindon UK
Mark.Dorling@
computingatschool.org.uk

John Woollard
University of Southampton
Highfield
Southampton UK
J.Woollard@
southampton.ac.uk

Abstract

Computational thinking is at the heart of the new English national curriculum for computing. There is a range of academic and pedagogic interpretations of the concept of computational thinking, a lack of understanding of the concepts and a close association of the subject with writing computer code using a programming language. Teachers might focus on a small aspect of the programme of study, thereby neglecting the breadth of content and the broader aims. In addition, the level descriptors associated with the curriculum have been removed creating a need for assessment guidance. In light of these changes, this paper explores the statutory requirements of the curriculum and the descriptions of computational thinking. It suggests a mechanism for assessment of achievement and progression for both computing and computational thinking.

Keywords: Computational thinking, assessment, secondary school, computer science education, curriculum design, pedagogy

Introduction

From September 2014, pupils in state-maintained schools will be expected to follow the programmes of study set out in the national curriculum document (Department for Education (DfE), 2013b). The subjects addressed in this document include computing. In addition, the statutory assessment framework is being removed and the system of assessment levels is not to be replaced (DfE, 2013a).

Computational thinking sits at the heart of the national curriculum programme of study for computing. The opening sentence states "A high quality computing education equips pupils to use computational thinking and creativity to understand and change the world" (DfE, 2013b, p. 188). The scope of computational thinking is described in the first aim – "understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data representation" (DfE, 2013b, p. 188). There are many different interpretations of the concept of computational thinking. Jeanette Wing, when she first used the term, defined computational thinking as including "… a range of mental tools that reflect the breadth of the field of Computer Science" (Wing, 2006, p. 33).

However, there is a strong emphasis, being led by the media, implying that the new computing curriculum focuses on "coding" (Crow, 2014; Nettleford, 2013). This misleading message, received by teachers and parents, could have a negative impact in the classroom. There is a danger of teachers focusing on a small aspect of the programme of study, thereby neglecting the breadth of the subject content and the broader aims.

Computational thinking is itself in danger of becoming a "buzz word" in the teaching of computing. Teachers acknowledge the need to teach computational thinking but may struggle with the various and conflicting interpretations of its nature. This may be the result of debate by individuals and groups (Computer Science Teachers Association (CSTA), 2011; Henderson, et al., 2007; Lu, et al., 2009; Naughton, 2012; Wing, 2006; Wing, 2008; Yadav, et al., 2011) concerning what is and is not computational thinking. Some of these definitions are broad, overlapping other subjects (Bundy, 2007; CSTA, 2011). In order to facilitate incorporation of computational thinking into classroom practices, a narrower definition is required. Once computational thinking is defined adequately, appropriate assessment instruments can be designed (National Research Council, 2010).

Recent developments in pedagogy have focussed upon thinking skills (Department for Education and Employment (DfEE), 1999; Department for Education and Skills (DfES), 2002; Wickens, 2007) as underpinning areas of the curriculum. 'Thinking Hats', based on de Bono's work (de Bono, 2000; de Bono, 2007), is a popular approach in which pupils are encouraged to think about the way they think. The computing curriculum is now challenging pupils to think using particular strategies for solving problems and understanding situations, referred to as computational thinking.

There are a number of stages towards establishing a curriculum in which computational thinking can be taught and then assessed. These stages are:

- to establish an understanding of the current computing curriculum,
- to establish the meaning of computational thinking,
- to establish an assessment framework for the current computing curriculum, and
- to develop a method for evidencing the assessment of computational thinking.

Current computing curriculum

The programme of study has high-level aims in terms of the introduction of computer science (DfE, 2013b). The following extracts illustrate learner capabilities at different stages of primary and secondary education.

- At key stage 1 (ages 5-7), pupils should be able to "understand what algorithms are; how they are implemented as programs on digital devices; and that programs execute by following precise and unambiguous instructions" (DfE, 2013b, p. 189).
- At key stage 2 (age 7-11), pupils should be able to (among other things): "solve problems by decomposing them into smaller parts" and also "use logical reasoning to explain how some simple algorithms work and to detect and correct errors in algorithms and programs" (DfE, 2013b, p. 189).
- At key stage 3 (ages 11-14), pupils should be able to: "design, use and evaluate computational abstractions …" and "use logical reasoning to compare the utility of alternative algorithms for the same problem" (DfE, 2013b, p. 190).
- At key stage 4 (ages 14-16), pupils should be able to "develop and apply their analytic, problem-solving, design, and computational thinking skills" (DfE, 2013b, p. 191).

These extracts demonstrate an emphasis on the progressive development of computational thinking skills. Teachers in England, engaging with the new programme of study, are now frequently hearing the term computational thinking and may question what it means to them as classroom practitioners.

Along with the move toward computational thinking, there is a withdrawal from the use of national standardised levels and level descriptors. Under the auspices of the Department for Education (2013a), schools are now free to design their own assessment models. There are many reasons for this move, including the suggestion that assessment leads the teaching (Barker, 2013; Passmore, 2007; Warner, 2008).

Computational thinking concepts

Jeanette Wing broadly defines computational thinking as "… the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Cuny, Snyder, Wing, 2010, cited in Wing, 2011, p. 20). Wing indicates that these solutions can be carried out by any processing agent, whether human, computer, or a combination of both (Wing, 2006). The emphasis in this statement is on thought processes, not the production of artefacts or evidences.

Given Wing's description of computational thinking, the next step is to decompose that definition into a set of concepts. This work has been undertaken by Selby and Woollard (2013). The result refines the definition of computational thinking to six concepts: a thought process, abstraction, decomposition, algorithmic design, evaluation, and generalisation. All of these concepts are employed in problem-solving processes. Again, the emphasis in this list of concepts is on thought processes, not the production of artefacts or evidences.

Computing progression pathways

Although there is some disagreement concerning at what level a computing assessment framework should be developed, from a classroom practitioner's perspective, there is definitely a need for one. This section introduces the Computing Progression Pathways and describes how it can be used to acknowledge progression and reward performance in mastering both the computing programme of study content and computational thinking skills.

There is some debate about whether it is important that the arbitrary values of progression be standardised across schools. Naace (Harrison, 2014), in their guidance, indicate "…a school approach to assessment will need to be tailored to match their approach to the curriculum" (p. 1). Alternatively, the National Association of Head Teachers (NAHT) propose when translating the national curriculum into assessment criteria "… there is little room for meaningful variety, we suggest this job be shared between schools" (2014, p. 10). Whether it is designed by a single school or a collection of interested parties, an assessment framework is required by classroom practitioners.

The Computing Progression Pathways (Dorling and Walker, 2014) is an example of a non-statutory assessment framework. It was produced by a small team of authors and reviewers, all teachers, based on their classroom experiences. It is an interpretation of the breadth and depth of the content in the 2014 national curriculum for computing programme of study. It includes the dependencies and interdependencies between concepts and principles. This may help non-specialist teachers and inexperienced teachers to understand what should be taught in the classroom. It is publically available at this link:

https://www.hoddereducation.co.uk/Subjects/ICT/Series-pages/Compute-IT/Series-Box/Progression-Pathways/Progression-Pathways-Grid.aspx.

The framework is grid-based. Five of the six strands, represented as columns, are aligned with the range and content categories from the Computing at School curriculum (Computing at School, 2012) and the requirements of applicants to initial teacher training courses (DfE, 2012). These include algorithms, programming and development, data and data representation, hardware and processing, communication and networks. The sixth strand incorporates the more traditional concept of information technology. This breadth affords an opportunity to view the subject of computing as a whole, rather than the separate subjects of Computer Science, Digital Literacy, and Information Technology. Each row represents a level of pupil progression. Annotation of the framework suggests that key stages 1-2 cover the first four levels (pink, yellow, orange, and blue), that key stages 3-4 cover the next four levels (purple, red, and black), and that GCSE covers the final level (white). As an example, the purple cell under the "Hardware and Processing" strand states that a pupil "Recognises and understands the function of the main internal parts of basic computer architecture" (Dorling and Walker, 2014).

The colour-coded rows may aid teachers in assessing whether pupils are exhibiting competences at different levels and in recognising achievement and attainment. In addition, adherence to the colour-coded statements can provide standardisation across schools as identified by the NAHT (2014). Institutions planning to use this assessment framework with existing assessment or reporting systems may:

- assign values or levels to the coloured rows,
- agree the benchmark value, level, or entry point for a particular key stage,
- assign the benchmark value or level to the appropriate progression statements.

The Computing Progression Pathways also affords opportunities to celebrate achievement in computing. There is a growing interest in badges as an informal recognition of skill, knowledge, understanding, or attitude. They are made and awarded by commercial organisations, educational suppliers, websites, schools, teachers, and pupils (Hamilton and Henderson, 2013; Mozilla, 2014; Radiowaves Schools, 2014). Recognising and rewarding pupil achievement in each strand can be accomplished via coloured digital badges. Each strand can be assigned a separate digital badge. There may be two-tone badges for pupils working between coloured progression levels. Currently, there are no digital badge designs for the strands. Teachers and pupils who will be using the digital badge system are better placed to design and create them. The process of designing and creating the digital badges might promote learner ownership and student-centeredness (Reigeluth, 2013).

Evidence of assessing computational thinking

Given that computational thinking concepts have been defined (Selby and Woollard, 2013) and an assessment framework for the computing programme of study has been proposed (Dorling and Walker, 2014), a mapping can be developed to illustrate how computational thinking can be assessed over the full breadth and depth of the computing programme of study.

The key to developing this mapping lies in understanding that computational thinking concepts can be demonstrated in multiple ways. For example, decomposition is demonstrated by pupils breaking game logic down into levels (avoid traps, climb mountain, guess password). This can be mapped to the "Programming & Development" strand, blue row. However, it can also be demonstrated by pupils designing a library inventory (an inventory grid for DVDs, a different grid for books). This can be mapped to the "Data & Representation" strand, yellow row. These examples illustrate decomposition in terms of functionality and data structures, across strands (breadth) and across rows (depth).

Rather than provide specific examples, tied to activities, for each statement in the Computing Progression Pathways that illustrate one or more computational thinking concepts, consider the meaning of the computational thinking concept and how it might apply to the pathways' statement. This affords the opportunity for classroom practitioners to contextualise the pathways and computational thinking concepts in any way they see fit.

As an example of this approach, consider the purple cell of the "Hardware & Processing" strand of the Computing Progression Pathways. It requires that a pupil "Understands the concepts behind the fetch-execute cycle" (Dorling and Walker, 2014). The fetch-execute cycle can be viewed as an algorithm. Understanding of this demonstrates the computational thinking concept of algorithmic thinking. Therefore, at a minimum, this pathways' statement maps to the computational thinking concept of algorithmic thinking. Once this mapping is complete, it is possible to identify, across the breadth and depth of the programme of study, all those activities with potential to enhance computational thinking skills.

The following table is a reproduction of the blue row (mid-range of key stage 3) of the Computing Progression Pathways (Dorling and Walker, 2014). Each statement has been numbered. Where applicable, the computational thinking concepts associated with that statement have been indicated in the last column. The computational thinking concepts of abstraction, decomposition, algorithmic design, evaluation, and generalisation have been abbreviated to the first two letters. Care has been taken by 3 iterations of expert evaluation of the statements to avoid making assumptions about how the teaching might afford opportunities for computational thinking rather than strictly interpreting what is explicitly stated in the Computing Curriculum Pathways. For example, an exercise in a classroom might afford opportunities to identify suitability for purpose and efficiency of input and output devices. That would

fall into the yellow cell of the “Hardware & Processing” strand, where a pupil “Recognises and can use a range of input and output devices” (Dorling and Walker, 2014). The teaching affords the opportunity for evaluation, although the statement from the pathways does not indicate that it would be an evaluation-based exercise. The teaching of the fetch-execute cycle, previously mapped to algorithmic thinking, usually incorporates the ideas of instructions and data, which correspond to the concept of abstraction. The teaching affords the opportunity for abstraction, although the pathways statement does not explicitly anticipate this.

Strand	**Statement from the Computing Curriculum Pathway**	**CT Concept**
A	• Shows an awareness of tasks best completed by humans or computers. • Designs solutions by decomposing a problem and creates a sub-solution for each of these parts. • Recognises that different solutions exist for the same problem.	EV DE, AL, AB AL, AB
P&D	• Understands the difference between, and appropriately uses if and if, then and else statements. • Uses a variable and relational operators within a loop to govern termination. • Designs, writes and debugs modular programs using procedures. • Knows that a procedure can be used to hide the detail with sub-solution.	AL, DE AL, AB AL, DE, AB, GE AL, DE, AB
D&DR	• Performs more complex searches for information e.g. using Boolean and relational operators. • Analyses and evaluates data and information, and recognises that poor quality data leads to unreliable results, and inaccurate conclusions.	AL, EV EV
H&P	• Understands why and when computers are used. • Understands the main functions of the operating system. • Knows the difference between physical, wireless and mobile networks.	DE, AB AB

C&N	• Understands how to effectively use search engines, and knows how search results are selected, including that search engines use 'web crawler programs'. • Selects, combines and uses internet services. • Demonstrates responsible use of technologies and online services, and knows a range of ways to report concerns.	AB, EV AL, EV
IT	• Makes judgements about digital content when evaluating and repurposing it for a given audience. • Recognises the audience when designing and creating digital content. • Understands the potential of information technology for collaboration when computers are networked. • Uses criteria to evaluate the quality of solutions, can identify improvements making some refinements to the solution, and future solutions.	EV EV EV EV
Key	**Computing Curriculum Pathways** A Algorithms; P&D Programming and Development; D&DR Data and Data Representation; H&P Hardware and Processing; C&N Communication and Networks; IT Information Technology	**Computational Thinking Concept** AB Abstraction; DE Decomposition; AL Algorithmic Thinking; EV Evaluation; GE Generalisation

Table 1: Computational thinking and progression pathways in computing (Based on Dorling and Walker, 2014)

Using this strategy of identifying computational thinking concepts associated with the pathways' statements enables computational thinking to be assessed using the same framework as the programme of study. From a practitioner's perspective, there is no additional assessment or progression tracking required to fulfil the broad aim of the computing programme of study to incorporate computational thinking.

Conclusion

The computing programme of study (DfE, 2013b) includes the broad aim of incorporating computational thinking into the classroom. The subject content is detailed in the document, but the connection to computational thinking and its meaning is not. Removal of the statutory assessment frameworks, which did not assess computational thinking, leaves a void in assessing pupils' attainment. Both of these shortcomings have been addressed in this paper. An understanding of

computational thinking, based on the work of Selby and Woollard (2013), has been established. An assessment framework, the Computing Progression Pathways, has been used to illustrate the dependencies and interdependencies between the concepts and principles of the programme of study (Dorling and Walker, 2014). This work has demonstrated how the Computing Progression Pathways can be used to evidence the assessment of computational thinking directly. By using the assessment framework to evidence progression, with its underlying support for computational thinking concepts, it is possible for the classroom practitioner to assess computational thinking without introducing additional complexity to the assessment process. However, this does raise questions around the provision for teachers of a framework for the pedagogy of computational thinking that aligns to this assessment approach.

References

Barker, I. 2013. 'See-Sawing' Rankings Are Shaky Ground to Build On. *Times Educational Supplement* [Online]. Available: http://www.tes.co.uk/article.aspx?storycode=6317642 [Accessed 10-04-2014].

Bundy, A. 2007. Computational Thinking Is Pervasive. *Journal of Scientific and Practical Computing,* 1**,** 67-69.

Computer Science Teachers Association Task Force 2011. *K–12 Computer Science Standards,* New York, ACM.

Computing at School Working Group 2012. Computer Science: A Curriculum for Schools. Computing At School.

Crow, D. 2014. Why Every Child Should Learn to Code. *The Guardian* [Online]. Available: http://www.theguardian.com/technology/2014/feb/07/year-of-code-dan-crow-songkick.

de Bono, E. 2006a. *6 Thinking Hats (Revised),* London, UK, Penguin.

de Bono, E. 2006b. *How to Have Creative Ideas,* London, UK, Vermillion.

Department for Education. 2012. Subject Knowledge Requirements for Entry into Computer Science Teacher Training. Available: http://academy.bcs.org/sites/academy.bcs.org/files/subject%20knowledge%20requirements%20for%20entry%20into%20cs%20teacher%20training.pdf [Accessed 15-03-14].

Department for Education. 2013a. Assessing without Levels. Available: http://webarchive.nationalarchives.gov.uk/20130904084116/https://www.educ

ation.gov.uk/schools/teachingandlearning/curriculum/nationalcurriculum2014/a00225864/assessing-without-levels [Accessed 27-02-14].

Department for Education. 2013b. The National Curriculum in England, Framework Document. Available: www.education.gov.uk/nationalcurriculum [Accessed 13-08-2013].

Department for Education and Employment 1999. *Information and Communication Technology - the National Curriculum for England,* London.

Department For Education and Skills 2002. *Framework for Teaching ICT Capability: Years 7, 8 and 9,* London, UK.

Dorling, M. & Walker, M. 2014. Computing Progression Pathways. Available: https://www.hoddereducation.co.uk/Subjects/ICT/Series-pages/Compute-IT/Series-Box/Progression-Pathways/Progression-Pathways-Grid.aspx. [Accessed 28-02-14].

Hamilton, G. & Henderson, B. 2013. *So What Are Open Badges?* [Online]. Jisc. Available: http://www.jisc.ac.uk/blog/so-what-are-open-badges-28-aug-2013 [Accessed 01-04-2014].

Harrison, J. 2014. Naace Assessment Panel Guidance Available: http://www.naace.co.uk/curriculum/assessment [Accessed 15-03-14].

Henderson, P. B., Cortina, T. J. & Wing, J. M. 2007. Computational Thinking. *Proceedings of the 38th SIGCSE technical symposium on Computer science education.* Covington, Kentucky, USA: ACM.

Lu, J. J. & Fletcher, G. H. L. Year. Thinking About Computational Thinking. *In:* Proceedings of the 40th ACM Technical Symposium on Computer Science Education, 2009 Chattanooga, TN, USA. New York: ACM, 260-264.

Mozilla. *Open Badges* [Online]. Available: http://openbadges.org/about/ [Accessed 01-04-2014].

National Association of Head Teachers. 2014. Report of the Naht Commission on Assessment. Available: http://www.naht.org.uk/welcome/news-and-media/key-topics/assessment/profession-takes-lead-on-assessment-after-the-end-of-levels/ [Accessed 15-03-14].

National Research Council 2010. Report of a Workshop on the Scope and Nature of Computational Thinking. The National Academies Press.

Naughton, J. 2012. Why All Our Kids Should Be Taught How to Code. *The Guardian* [Online]. Available: http://www.theguardian.com/education/2012/mar/31/why-kids-should-be-taught-code [Accessed 01-04-2014].

Nettleford, W. 2013. Primary School Children Learn to Write Computer Code. Available: http://www.bbc.co.uk/news/uk-england-london-23261504 [Accessed 10-07-2013].

Passmore, B. 2007. Testing, Testing, Testing. *Times Educational Supplement* [Online]. Available: http://search.ebscohost.com/login.aspx?direct=true&db=trh&AN=24943568&site=eds-live.

Radiowaves Schools Ltd. *Makewaves* [Online]. Leeds. Available: https://www.makewav.es/ [Accessed 01-04-2014].

Reigeluth, C. M. 2013. *Instructional-Design Theories and Models: A New Paradigm of Instructional Theory*, Routledge.

Selby, C. & Woollard, J. 2013. Computational Thinking: The Developing Definition Available: http://eprints.soton.ac.uk/356481/ [Accessed 01-04-2014].

Warner, L. 2008. The 'Freedom to Frame Questions ... Worth Asking', or Three Stories and Three (Other) Fragments of Research. *English in Education,* 42, 1**,** 88.

Wickens, C. 2007. Creativity. *In:* Kennewell, S., Connell, A., Edwards, A., Hammond, M. & Wickens, C. (eds.) *A Practical Guide to Teaching ICT in the Secondary School.* Oxford, UK: Routledge.

Wing, J. 2006. Computational Thinking. *Commun. ACM,* 49, 3**,** 33-35.

Wing, J. 2008. Computational Thinking and Thinking About Computing. *Philosophical Transactions of The Royal Society A,* 366**,** 3717-3725.

Wing, J. 2011. Research Notebook: Computational Thinking - What and Why? *The Link.* Pittsburgh, PA: Carneige Mellon.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S. & Korb, J. T. 2011. Introducing Computational Thinking in Education Courses. *Proceedings of the 42nd ACM technical symposium on Computer science education.* Dallas, TX, USA: ACM.