Computing at school

Miles Berry

October 2016

2014 saw some radical changes to the content of England's national curriculum. Perhaps one of the most significant changes was the replacement of the subject "Information and communication technology", with a focus on end-user skills using a range of office and media applications, with a new subject, "Computing", where the emphasis was rather on the conceptual understanding and knowledge of the principles of computer science, the applications of computational thinking and the craft of computer programming. There has been much international interest in these changes, with many countries following England's lead in making changes to their own curricula to incorporate programming and other aspects of computer science. Here, I discuss a number of the rationales offered for such a radical change, offer a brief narrative of the process of moving from one curriculum to another, survey some of the content of the new curriculum and conclude with an outline of the role of the Computing At School community in implementing this curriculum.

Some rationales

Whilst many would argue that elementary, or indeed secondary, education affords no place for a discipline as specialised as computer science, there are a number of convincing rationales for including this subject, alongside music, poetry and physics, as part of the educational entitlement for all children. Such rationales might include one or more of the following.

- **Children's inherent curiosity in the world around them.** It is hard to deny that digital technology is now almost ubiquitous. A typical infant is likely to encounter touch screen devices such as smartphones or tablet computers on her mother's lap, before she even learns to speak. Children are naturally curious about the world in which they grow up, and thus are curious about the technology which is, for them, an inherent part of that world. The child grows up, yes, using digital technology, but also wonder how it is made and how it works. Part of the school's role is to nurture this curiosity and help the child learn for herself the answer to such questions:

  > "One of the main educational tasks of the primary school is to build on and strengthen children's intrinsic interest in learning and lead them to learn for themselves" [1]

- **Computer science as part of scientific understanding.** There is more to computer science as a discipline than 'merely' understanding how technological artefacts

---

[1] Plowden, B., 1967. Children and their primary schools: A report of the Central Advisory Council for Education (England). London: HM Stationery Office.

function and are developed. Computer science, like the other, older, sciences seeks to provide insight into the nature of reality itself. Computability is one example of this: it seems in the nature of our universe that problems fall into three categories: those which are easy to solve, those that are hard to solve and those which can never be solved. Theoretical computer science considers ideas such as these, and provides a means to build useful systems using them. [2]

- **Economic benefits.** For ministers, the need to sustain England's vibrant software industries must undoubtedly have been a priority. In order to maintain successful games, visual effects, fintech and cybersecurity sectors, a steady stream of high calibre computer science graduates is needed, and the decline in the number of students studying such disciplines was thought best addressed through ensuring exposure to computer science for all as part of school education. In addition to this, computing is a largely meritocratic field with much potential for social mobility - home background and educational privilege are less important than the quality of the code a candidate can write.

- **Preparation for an uncertain future.** Few would doubt that computers are likely to play a more significant role in our lives, and the life of our society, in our future than in our past or present. It can be argued that the best way to prepare for such a future is to through acquiring an understanding of how computers work and of how they are programmed: this seems more likely to be useful in the long term than skills of current relevance such as centering text on presentation slides. In a society in which computers will play an increasingly significant role, our duty as educators is to empower our pupils to take charge of the machines: as Douglas Rushkoff memorably puts it:

  > "Program or be programmed." [3]

- **As part of a rounded, liberal education.** Much of what is included in school curricula is less about vocational training for the world of work and more about providing a broad and balanced education, in which pupils develop an understanding of their world and are empowered to make positive changes to that world through expressing themselves creatively. Froebel's gifts[4] allow young children to learn such fundamentals as the conservation of number, the conservation of volume, the conservation of shape and the direction of gravity, but they also allow kindergarteners to problem solve and build something of their own choosing. The same is true of programming languages: they are a means to understanding technology and reality,

---

[2] Bentley, P.J., (2012) Digitized: The Science of Computers and How it Shapes our World. Oxford: Oxford University Press.

[3] Rushkoff, D., 2010. Program or be programmed: Ten commands for a digital age. Or Books.

[4] See, e.g, Resnick, M., 1998. Technologies for lifelong kindergarten. Educational technology research and development, 46(4), pp.43-55.

and a medium for creative expression. Just as Frank Lloyd-Wright's later work as an architect was coloured by his kindergarten experience of playing with Froebel's gifts [5], so too will the software engineers and computer scientists of the future draw on their early experience of thinking computationally and coding with blocks and floor turtles.

From ICT to computing

Much of the inspiration for computing, particularly computer programming, in school education comes from Seymour Papert's work in the 70s, 80s and 90s, particularly in relation to the educationally focussed programming language Logo, its implementation of turtle graphics, and the constructionist theory of learning. Back in 1980, Papert wrote:

> In many schools today, the phrase "computer-aided instruction" means making the computer teach the child. One might say the computer is being used to program the child. In my vision, the child programs the computer and, in doing so, both acquires a sense of mastery over a piece of the most modern and powerful technology and establishes an intimate contact with some of the deepest ideas from science, from mathematics, and from the art of intellectual model building. [6]

Little seems to have changed in the intervening time. Much of the educational use of computers offers little more than drill-and-practice behaviourist learning, which undoubtedly has its place but seems a poor alternative to children taking charge of the computer and using it as a tool for a far more creative, exploratory mode of learning. Crucially for Papert, programming was never an end in itself, but the means through which children connect to deep ideas in mathematics, science and philosophy. The same vision motivates much of what has been included in England's computing curriculum.

For many of those involved in drafting and then implementing this curriculum, there was a feeling of needing to move children on, beyond what they had already achieved, or might achieve for themselves without instruction from teachers. By and large, children moving from primary to secondary education in England (i.e. at the age of eleven), were already:

- effective users of technology;
- good at communicating digitally, even if they would not always observe the terms and conditions of the services they used;
- able to do the equivalents of reading and writing in the digital domain;
- able to keep themselves reasonably safe when using online technologies; and
- able to demonstrate a sound portfolio of digital skills.

---

[5] Lloyd-Wright, F., 2005. Frank Lloyd Wright: An Autobiography. Pomegranate.

[6] Papert, S., (1980) Mindstorms: Children, Computers, and Powerful Ideas. New York: Basic Books

However, there was some sense that, for all they knew, the technologies they were using might as well be based on magic[7].

The ambition was to move children's learning on, beyond this level, so that they would also be:

- able to make technological artefacts, and potentially new technologies;
- able to collaborate productively online;
- critical of content, tools and assumptions when working online;
- responsible in their own use of technology, with some sense of the consequences for others of their actions; and
- empowered by some degree of understanding of how technologies worked and were made.

Thus replacing the sense of technology being almost magical with some concrete knowledge of the principles on which it depends.

Despite the reference to technology above, the new curriculum concerns itself far less with technology than it does with the fundamental principles and concepts of computer science. In the introduction to the earlier, non-statutory CAS Computer Science Curriculum [8], it was stated:

> Computer science encompasses foundational principles and widely applicable ideas and concepts. It incorporates techniques and methods for solving problems and advancing knowledge, and a distinct way of thinking and working that sets it apart from other disciplines. It has longevity (most of the ideas and concepts that were current 20 or more years ago are still applicable today), and every core principle can be taught or illustrated without relying on the use of a specific technology.

This concept of computer science as a foundational discipline like physics or history undoubtedly coloured the development of the curriculum.

In August 2012, the British Computer Society and the Royal Academy of Engineering were approached by the government Department for Education to draft a new programme of study for computer (or ICT as it was still called at the time), as expert advice to ministers. Under the chairmanship of Simon Peyton Jones, a diverse group of stakeholder representatives were assembled, drawing on sector representative bodies, multinational technology companies, subject associations, universities and schools.

---

[7] qv Clarke, A. C. (1973). Profiles of the Future: An Inquiry into the Limits of the Possible. Popular Library.

[8] CAS (2012). Computer science: a curriculum for schools. Available at: https://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf

The drafting panel took inspiration from the Royal Society's report on computing in schools [9], which recommended that ICT as a subject be replaced with a new subject, computing, recognised as having three distinct but inter-related components: computer science, information technology and digital literacy. These can be thought of as the foundations, applications and implications of the discipline.

In the context of Google search, we might ask at the foundation, computer science level, how does the Page Rank algorithm determine the relevance of a page and how are search queries served so quickly irrespective of geographic location; at the application, IT level, how might results be filtered more effectively to show those only in my language or from a particular range of dates; and at the implication or digital literacy level, what data does Google store on me so that it can better target advertising for me, and is this a price worth paying to use their search engine for free?

The first sentence of the new curriculum sets out it's vision, and how computing forms part of a liberal education designed to provide children with an understanding of their world and the tools needed to contribute creatively to it:

> A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world. [10]

Whilst creativity has long been an element of elementary and secondary education, the 'computational thinking' referred to here is relative new. Although Seymour Papert mentioned the term in Mindstorms[11], computer scientist Jeanette Wing is widely credited with popularising the idea and has been instrumental in the emphasis that this concept has received in computing curricula worldwide. For Wing, computational thinking is:

> "… 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" [12]

---

[9] Furber, S. (2012) Shut down or restart. London: The Royal Society

[10] Department for Education (2013) The national curriculum in England Framework document. London: DfE.

[11] Papert, S., (1980) Mindstorms: Children, Computers, and Powerful Ideas. New York: Basic Books

[12] Wing, J. 2011. Research Notebook: Computational Thinking - What and Why? The Link. Pittsburgh, PA: Carneige Mellon. Available at: http://www.cs.cmu.edu/sites/default/files/11-399_The_Link_Newsletter-3.pdf

In helping teachers understand what this might mean in practice, the Computing At School Barefoot Computing project[13] outlined six concepts of computational thinking and five approaches. The concepts are:

- **Logical reasoning** - computers are deterministic machines and we can reason logically about how they will behave. Pupils learn to predict confidently the output from a program, to detect and correct errors in their own and others' programs, and the basics of Boolean logic.
- **Algorithms** - computer programming requires that the programmer has a clear idea of how to achieve a particular objective before she begins coding, and that some approaches to solving a problem are inherently more efficient than others.
- **Decomposition** - when faced with large problems, engineers break these up into smaller parts, and address each of these separately.
- **Generalisation** - engineers will often look for solutions to more general problems, borrowing liberally on the work and ideas of others who have been successful in solving equivalent or related problems.
- **Abstraction** - programs and computers are complex systems, and a multilayered model of abstraction has been necessary to manage the complexity of these systems, in which the detail of implementation can be effectively hidden. In tackling complex problems, a similar approach is helpful, in which the solver's focus rests on the necessary detail.
- **Evaluation** - it is necessary to review whether potential solutions are correct, suitable, reliable, efficient and elegant.

Whilst all these can be effectively learnt in the context of computer programming, they also have wide applications across and beyond the school curriculum, and might provide a set of core competencies in any problem solving work, particularly when this involves using computers.

Alongside these concepts are a number of approaches, which might describe how software engineers and others tackle problems, and provide a foundational model for pedagogic practice in computing education:

- **Tinkering** - a certain playfulness, or willingness to explore and experiment seems to characterise the work of many leading software engineers, and links closely with play as a powerful pedagogic approach in early years education.
- **Creating** - computing is an inherently creative discipline, and Papert's constructionist insights suggest that a learner's conceptual understanding might best be developed through the conscious construction of knowledge artefacts designed to be shared with others.

---

[13] Barefoot Computing (2014) Computational Thinking. Available at http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking/

- **Persevering** - programming is undeniably hard, but for many this is where its joy lies; encouraging pupils to develop a 'growth mindset'[14] and an associated willingness to persevere in the face of challenges is important, not just for success in computing[15].
- **Debugging** - One particularly powerful way to help pupils develop such a growth mindset is through the repeated experience of detecting and correcting the mistakes (or 'bugs') in their own algorithms and programs.
- **Collaborating** - Large computer programs are written by large, well-coordinated teams of people, often widely distributed. There's evidence that pair-programming is an effective methodology for agile software development[16], and having pupils work with a partner on programming tasks seems to be motivating and pedagogical effective.

Curriculum content

The revised national curriculum includes relatively brief requirements for what should be taught to 5-16 year olds - the content itself fits comfortably on three sides of A4 paper. For younger pupils, provision is governed by the 'Early Years Foundation Stage Framework'[17].

Before the age of five

For the youngest pupils in nursery schools and school reception classes, there's, not surprisingly, no requirement that pupils be taught to use digital technology or learn to program. However, the framework for these children's education does list a number of 'characteristics of effective learning', including 'creating and thinking critically'. The non-statutory Development Matters guidance[18] lists a number of aspects of this, such as 'finding ways to solve problems', 'making links and noticing patterns in their experience' and 'planning, making decisions about how to approach a task, solve a problem and reach a goal'. There are strong parallels here with the concepts and approaches of computational thinking.

---

[14] Dweck, C (2012) Mindset. New York NY, Random House.

[15] Cutts, Q., Cutts, E., Draper, S., O'Donnell, P., and Saffrey, P. (2010) Manipulating mindset to positively influence introductory programming performance. In: SIGCSE '10 Proceedings of the 41st ACM Technical Symposium on Computer Science education, Milwaukee, USA, 10-13 Mar 2010, pp. 431-435.

[16] Williams, L. A., & Kessler, R. R. (2000). All I really need to know about pair programming I learned in kindergarten. Communications of the ACM, 43(5), 108-114.

[17] DfE (2014) Statutory framework for the early years foundation stage. London: DfE

[18] The British Association for Early Childhood Education (2012) Development Matters in the Early Years Foundation Stage (EYFS). London: Early Education

It seems more appropriate in early years education to encourage young children to plan systematically, develop resilience and to make predictions, thus laying the foundations of computational thinking, than to introduce them to programming per se, although products such as Bee-Bots and Scratch Jr have done much to make this accessible.

Ages five to seven

The curriculum requirements for ages five to seven include that pupils be taught to understand what algorithms are. Pupils will learn about algorithms as sequences of steps or sets of rules. They'll typically do this away from computers, through activities which involve them following a teacher's instructions, giving instructions to their peers or working out their own instructions for practical activities such as sharing a pile of sweets[19].

Pupils are also taught how algorithms are implemented as programs on 'digital devices'. The phrasing here is deliberate, so as to encourage teachers to use floor turtles and other robots with young learners rather than going directly to on screen coding. It seems easier for pupils to put themselves in the place of a floor turtle to step through programs themselves (cf Papert's 'body syntonic' reasoning[20]), thus making it easier for them to reason about their programs. Reasoning logically about programs and algorithms is a recurring theme in the curriculum - it's considered crucial for pupils to be able to think about their code rather than just code.

Ages seven to eleven

Between the ages of seven and eleven, pupils' programming becomes more formal, as they learn to use a greater variety of structures in their code, beyond simple sequences of steps. They're taught about selection and repetition, as well as the use of input and output and are introduced to variables as a data structure. Typically their programs will be written in MIT's block-based toolkit Scratch[21], although this is not explicitly specified in the programme of study.

Logical reasoning is again emphasised, with pupils expected to explain how simple algorithms work, as well as detecting and correcting errors in algorithms and programs.

Other elements of computer science are covered too: pupils are taught how computer networks including the internet work, and how they can provide services such as the World Wide Web. Typically, this will draw on 'unplugged' approaches involving classroom based

---

[19] Barefoot Computing (2014). Sharing sweets activity. Available at: http://barefootcas.org.uk/programme-of-study/understand-algorithms/ks1-sharing-sweets-activity/ (free registration required)

[20] Papert, S., (1980) Mindstorms: Children, Computers, and Powerful Ideas. New York: Basic Books

[21] Available at http://scratch.mit.edu

simulations, although adventurous schools might introduce pupils to command line networking tools such as `ping`, `nslookup` and `traceroute`.

Ages 11-14

In the first years of secondary education, pupils are introduced to the idea of a computational abstraction which models the state and behaviour of a system. This might be a simple game, although computational models for the spread of epidemics or the growth of a culture of cells are interesting alternatives.

Pupils' knowledge of algorithms is now extended to include 'key algorithms that reflect computational thinking'. Typically schools interpret this as algorithms for search and sort, but it might also include some mathematical algorithms like Euclid's algorithm for highest common factors and the Sieve of Erastothenes. Pupils learn that some algorithms provide more efficient solutions to the same problem.

Pupils extend their knowledge of how computers work, learning about the hardware and software components of computer systems and how these interoperate. Teachers have found that simpler systems, such as the BBC micro:bit[22] make it easier to scaffold this understanding.

Ages 14-16

There are minimal statutory requirements for computing in the national curriculum beyond the age of 14, but computing does remain a compulsory subject for pupils at this age. The requirements can be satisfied through embedding computer science, information technology and digital literacy across the rest of the curriculum, or through providing a number of focus days or independent project work over the course of these two school years.

In addition to this, there's a requirement that pupils "must have the opportunity to study" IT and computer science in greater depth - this is interpreted as meaning that schools should offer courses leading to public exams in IT or computing at 16+, but not require pupils to enter such exams.

Detailed specifications have been developed by English exam boards for General Certificate of Secondary Education (GCSE) qualifications in computer science[23]. These include aspects of computational thinking including algorithm design, knowledge of Boolean logic and binary representation and an extended practical programming project, most typically undertaken in Python.

---

[22] See http://microbit.org

[23] See, for example, OCR (2016) Specification (Accredited) - GCSE Computer Science - J276. Available at http://ocr.org.uk/qualifications/gcse-computer-science-j276-from-2016/

Ages 16-18

Between the ages of 16 and 18 computer science becomes an elective subject, with relative small, but growing, numbers of pupils continuing to study the subject at this age.

Again, English exam boards have developed detailed specifications for qualifications at this level[24]. These include some deliberately ambitious material, such as an introduction to functional programming, often taught in Haskell, graphs and algorithms for graph traversals and shortest paths and big-O notation. They also include a significant, more extensive project which might include the development of a computer game or simulation with a graphical user interface or an investigation of topics such as machine learning or 3-d graphics.

Whilst such qualifications naturally lead on to undergraduate degrees in computer science, they are increasingly recognised as useful preparation for degrees in many other disciplines which make use of computing, such as engineering, natural and social sciences, teaching and medicine.

Computing At School

This significant change in curriculum content and the associated assessment framework has taken place at a time when government has deliberately stepped back from the implementation of such changes. Currently in the UK, the view is that:

> Government should only do what only government can do.[25]

Much of the implementation of computing as a curriculum subject in England had been achieved through the Computing At School (CAS) group[26], the UK subject association for computer science, formally part of BCS, the Chartered Institute for IT.

CAS has seen rapid growth in its membership since its beginnings as a small working group of computing teachers and academics. Membership currently stands at 25,653 with some 223 local hubs.

The main challenge for implementing the computing curriculum has been the need to ensure sufficient numbers of teachers in schools with sufficient confidence to make the new subject a success:

---

[24] See, for example, AQA (2015) AS and A-level Computer Science. Available at http://www.aqa.org.uk/subjects/computer-science-and-it/as-and-a-level/computer-science-7516-7517

[25] GDS (n.d.) Government Digital Service Design Principles. Available at https://www.gov.uk/design-principles

[26] See http://computingatschool.org.uk

> The hardest part of getting great computer science in every school is getting a great computer science teacher in every school.[27]

Providing great computer science teachers for every school can be done through two routes: initial teacher training and continuous professional development. The former is now addressed through the development of subject knowledge entry requirements for specialist computer science teachers[28] and through the provision of generous tax-free training bursaries and scholarships for those with good degrees in computer science who choose to train to become teachers[29].

Great computer science teaching demands three things: great teaching skills, great technology skills and great computer science subject knowledge[30]. The lattermost is a particularly challenge: in primary schools very few teachers have any background in computer science or software development; even in secondary a minority of those who were teaching the former ICT curriculum had a degree in computer science[31].

In developing continuing professional development programmes for computing, Computing At School has prioritised addressing teachers' subject knowledge. The most significant programme in this area has been the Network of Excellence in Computer Science Teaching, comprising around 400 'Master Teachers', serving classroom teachers supported, and partly funded, to provide continuing professional development training and peer-support to those teaching computer science in their area. More recently, the Network of Excellence has been re-configured on a regional basis, with ten universities acting as regional centres, typically drawing on the combined expertise of their computer science and education faculties.

CAS has also developed a range of resources to support teachers, including introductory guides to the curriculum, QuickStart handbooks with associated CPD resources[32], the

---

[27] Quote taken CS for All summit, White House, Washington DC, 14/9/16.

[28] Teaching Agency (2012) Subject knowledge requirements for entry into computer science teacher training. Available at http://www.computingatschool.org.uk/data/uploads/CSSubjectKnowledgeRequirements.pdf

[29] See https://getintoteaching.education.gov.uk/funding-and-salary/overview/funding-by-subject/funding-for-training-to-teach-computing

[30] Mishra, P. and Koehler, M.J., 2006. Technological pedagogical content knowledge: A framework for teacher knowledge. Teachers college record, 108(6), p.1017.

[31] Furber, S. (2012) Shut down or restart. London: The Royal Society

[32] See http://www.quickstartcomputing.org/

Barefoot Computing programme for primary computing[33], and Tenderfoot Computing for lower secondary teachers[34]. CAS members. CAS members have developed further computing resources themselves, and CAS has an active, gift economy culture of members sharing their resources under liberal licences with the rest of the community: at present, 3,863 resources have been shared in this way.

Concluding remarks

It is still too early to judge whether the changes to England's curriculum have been a success. There is undoubtedly much interest in programming and coding amongst pupils in English schools, and some surveys suggest that implementation has, given that there has been little involvement in this process from central government, been really quite successful: BT commissioned an Ipsos MORI survey of 400 primary teachers reporting that 81% are now confident in teaching computing[35]. Britain's Royal Society are currently conducting wider ranging research and their report is eagerly anticipated[36].

Exam entries at GCSE for computing have significantly increased in the period since the introduction of this qualification, from 4,253 in 2013 to 62,454 in 2016. At A Level, there has been a more modest increase, from 3,758 to 6,242 in the same period[37]. There's also evidence of an increase in recruitment to university computer science courses over this time frame[38], and many are optimistic that these trends will continue.

Perhaps the most significant outcome of England's decision to include computer science in its curriculum is that this subject now becomes an entitlement *for all*, which, it is hoped, will go a long way to address inclusion and equity in this domain.

---

[33] Barefoot Computing (2014) Computational Thinking. Available at http://barefootcas.org.uk/barefoot-primary-computing-resources/concepts/computational-thinking/

[34] See https://www.computingatschool.org.uk/custom_pages/56-tenderfoot

[35] BT and Ipsos MORI (2016). A new cornerstone of modern primary school education. Available at https://techliteracy.co.uk/IPSOS-full.pdf

[36] See https://royalsociety.org/topics-policy/projects/computing-education/

[37] Data from http://www.jcq.org.uk/examination-results

[38] Shadbolt, N. (2016) Shadbolt Review of Computer Sciences Degree Accreditation and Graduate Employability. Available at https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/518575/ind-16-5-shadbolt-review-computer-science-graduate-employability.pdf