Implementing the national curriculum in England

Miles Berry, University of Roehampton

February 2018

Introduction

In September 2014, England introduced a new curriculum subject, Computing. This included programming and other elements of computer science in the curriculum for pupils from age 5 onwards, replacing Information and Communication Technology (ICT) which many characterised as focussing almost entirely on end-user application skills. Over three years on, we can look back at how successful the implementation of the new curriculum has been, and perhaps draw out some lessons for those seeking to implement a similar change.

It's worth recognising that the English coalition government that introduced the computing curriculum alongside reform of the other subjects in the national curriculum also deliberately pulled back from the details of implementation. Amongst his first acts as education minister, Michael Gove abolished the English qualifications and curriculum development agency and the British educational communications and technology agency. The British government's stated policy was that 'government should only do what only government can do'.

Oates (2011a), who chaired the panel who designed the overall framework for the English national curriculum, argues that a coherent curriculum should have content arranged in an order associated with age-related progression, and that elements of content, assessment, pedagogy, teacher training, resource materials and incentives should all line up and act in a concerted way. Below, I examine each of these elements in turn, to explore the extent to which England's implementation of its computing curriculum has been coherent.

Content

Unlike the other subjects of England's national curriculum, the computing programmes of study were developed through an open consultation led by the British Computer Society and the Royal Academy of Engineering. The curriculum encompasses computer science, information technology and digital literacy, perhaps best thought of as the foundations, applications and implications of digital technology.

The curriculum places computational thinking and creativity at the heart of computing, as the twin, golden threads which run throughout the programmes of study:

> "A high-quality computing education equips pupils to use computational thinking and creativity to understand and change the world." (DfE, 2013)

The programmes of study themselves are remarkably concise, running to just two and bit pages to outline all the computing which pupils should be taught from age five to sixteen. Whilst many may feel this leaves too much open to the interpretation of individual schools and teachers, others welcome the flexibility and freedom that such brevity affords.

Alongside the curriculum requirements, the Department for Education also developed the content requirements for the elective GCSE and A Level computer science qualifications: it is these specifications which typically drive the content of computer science lessons for 14-18 year olds.

A particular challenge for those teaching computing was the 'all at once' way in which the computing curriculum and, to a lesser extent, new qualifications were put in place: the curriculum for lower secondary pupils assumes that they have studied the primary curriculum, and the GCSE and A Level specifications similarly assume that students have studied the lower secondary and GCSE courses respectively. Yet, in these first years of implementing computing, this could never be the case. A more strategic approach might see a gradual phasing in of computer science, starting with younger pupils and making changes to curriculum and qualifications as the initial cohort gradually make their way through the education system.

Assessment

Very little detail was provided to teachers on how to assess the new national curriculum. The national curriculum review commission (Oates, 2011b) were clear in their advice that the old system of levels of attainment should be removed and not replaced:

> We thus emphasise the importance of establishing a very direct and clear relationship between 'that which is to be learned' and all assessment (both formative and ongoing, through to periodic and summative). Imprecise Attainment Targets and the current abstracted, descriptive 'levels' are of concern, since they reduce the clarity of this relationship. (Oates, 2011b, p42)

This recommendation was accepted by ministers, and thus each key stage for subject in the new curriculum has a generic, statutory 'attainment target':

> By the end of each key stage, pupils are expected to know, apply and understand the matters, skills and processes specified in the relevant programme of study.

Many schools, however, seemed reluctant to move from a system that had been used to assess progress and attainment for the previous 24 years, and appeared keen to find some way to re-invent the old system and apply it to the new curriculum. Frameworks such as Dorling & Walker (2014)'s 'Progression Pathways' were developed, and initially received positively by teachers. More recently, the McIntosh Commission reiterated the importance of removing and not replacing levels:

> Levels distorted the purpose of in-school assessment, particularly day-to-day formative assessment. The Commission believes that this has had a profoundly negative impact on teaching. (Mcintosh, 2015, p5)

Interestingly, the McIntosh commission recommended the development of item banks to facilitate assessment:

> The Commission recommends the establishment of a national item bank of assessment questions to be used both for formative assessment in the classroom, to help teachers evaluate understanding of a topic or concept, and for summative assessment, by enabling teachers to create bespoke tests for assessment at the end of a topic or teaching period. (Mcintosh, 2015, p9)

Pragmatically, assessing pupils' capability in computing must use a combination of questions, to assess knowledge, and projects, or perhaps problems, to assess the application of that knowledge (qv Grover, Cooper, & Pea (2014)).

In assessing project work, teachers might well look for evidence of the individual constructs listed in the programmes of study, as well as qualities such as creativity, perseverance and collaboration - this approach continues to dominate at primary level.

In assessing knowledge, questions, including multiple choice questions, have a role to play. Computing At School (the UK subject association for CS teaching) has been leading work on crowdsourcing an item bank of multiple choice questions for computing, Project Quantum, with 7,488 questions freely available at the time of writing. These are, unsurprisingly, skewed towards the secondary curriculum, but a number have been curated into a quiz to provide a baseline for assessment at the beginning of secondary education.

At GCSE, assessment remains problematic. The exam boards included a non-exam assessment (NEA) within the specifications, counting for 20% of the final exam grade. This is a programming project undertaken by pupils individually in response to a detailed problem specification. Pupils are allowed twenty hours and must work on the task individually and without internet access. Regrettably, there appears to have been widespread misconduct, with pupils, and some

teachers, sharing the task specification, and even complete solutions in various online fora, in contravention of the exam regulations. After consultation, it was decided that scores from the non-exam assessment would no longer count towards final grades, but pupils would nevertheless have to complete the tasks, albeit under rather less strict conditions.

The A-level coursework is much more open-ended, with pupils developing a computing project of their own choice, in which their skills can be effectively demonstrated. Former A Level students have completed some very impressive projects, including in artificial intelligence, control systems and networking, before heading to university to read computer science and related degrees. Take up of this qualification remains very low (in 2017, there were just 7,607 entries), with a worryingly low proportion of entries from female students (under 10% in 2017).

Pedagogy

Compared to longer established subjects, there is little concrete knowledge within the profession, or indeed teacher training establishments, about how best to teach computing, particularly in schools. Whilst research was undertaken by those following Seymour Papert's lead in introducing Logo programming to primary and lower secondary education (see, e.g. the review by Yelland (1995)), and research into effective practice continues to be by academic computer scientists working with undergraduates, there remains a relative paucity of research into computing pedagogy in schools.

English teachers have taken a largely pragmatic line in deciding how best to teach computing, figuring out for themselves what works for their pupils in their schools. This is perhaps unsurprising, given the conclusions of Fossati & Guzdial (2011) that:

> Computer Science instructors rely mostly on intuition and anecdotal evidence to make decisions about changes in their daily teaching practice... instructors used little empirical data for deciding to make a change, and for deciding whether a change was successful or a failure.

English teachers have made use of social media, CAS hub meetings, and professional development run by CAS appointed 'Master Teachers' to share their classroom practice with one another. Sentance & Csizmadia (2015) present results of their survey into the strategies adopted by some 357 teachers. There seems surprisingly little consistency in these teachers' responses - for example only 12% reported scaffolding or modifying code as an approach, with only 8% relating programming tasks to the real world. 13% taught programming through unplugged activities, in which concepts are taught through examples and activities away from the computer, despite the rather mixed evidence for the efficacy of this approach (Taub, Ben-Ari, & Armoni, 2009).

The English curriculum places computational thinking at the centre of computing education, and this emphasis perhaps explains some teachers' preference for 'unplugged' pedagogies, as might non-specialist teachers' limited confidence in computer programming. There is, though, a lack of clarity over what is meant by computational thinking, with many seeing it as some generalised approach to problem solving:

> It is the process of recognising aspects of computation in the world that surrounds us and applying tools and techniques from computing to understand and reason about natural, social and artificial systems and processes. It allows pupils to tackle problems, to break them down into solvable chunks and to devise algorithms to solve them. (Csizmadia et al., 2015)

There is though little evidence that computational thinking can be applied effectively to problems outside of those involving computation. Indeed it could be argued that computational thinking without computation is merely thinking. Tedre & Denning (2016) argue that we should be honest about the scope of computational thinking:

> Computational thinking—the habits of mind that many of us have developed from designing programs, software packages, and computations performed by machines—offers very powerful mental tools for people who design computations.

On the other hand, it seems mistaken to assume that these habits of mind are automatically acquired through the teaching of programming. Rather, it appears that this must be accompanied by more or less explicit instruction in logical reasoning, algorithms, abstraction and generalisation. The popular, extra-curricular Code Club network conducted a control trial based evaluation of its impact, reporting, unsurprisingly, significant progress in coding skills amongst those who attended for a year, but on the computational thinking questions, outcomes were not significantly better than the control group (Straw, Bamford, & Styles, 2017).

Teacher training

Excellent computing education in schools demands that schools have excellent computing teachers. There remains a shortfall in the numbers of teachers qualified to teach computing in secondary education (Royal Society, 2017), very few primary school teachers have an academic or professional background in computer science. To address this shortage, more computing teachers can be recruited and trained, or existing teachers can undertake the professional development necessary to be able to teach computing with confidence and competence. England has used both approaches since the introduction of its new curriculum, with only limited success thus far.

The Teaching Agency developed a set of subject knowledge requirements for entry into computer science teacher training (Teaching Agency, 2012). These broadly encompass the subject knowledge deemed necessary for teaching primary and secondary levels, although it should be noted that they were developed before the national curriculum and the most recent qualification requirements were produced. These have, however, provided some benchmark for teacher training and professional development, as well as setting a standard for entry into secondary initial teacher training courses.

Before the introduction of the computing curriculum, secondary ICT also struggled to fill the allocated teacher training places, and the situation has not improved subsequently. Despite generous, tax-free bursaries and scholarships for those with good honours degrees or postgraduate qualifications who choose to train to become secondary computing teachers, only 66% of the allocated training places have been taken up for the current academic year (Department for Education & National College for Teaching & Leadership, 2017). It would seem that teaching is not yet the most appealing career choice for the majority of English computer science graduates.

Rather than waiting for a new generation of computing teachers to join the profession, efforts have been made to address the professional development of those already in the profession, particularly primary school teachers, who typically teach all subjects to their class, and the former teachers of ICT in secondary school. Thinking of teachers' knowledge using Koehler & Mishra (2005)'s TPCK framework, most professional development thus far has focussed on the need to equip teachers' with the content knowledge, and perhaps the technological knowledge, needed for teaching computing to their classes; it is only recently that attention has been paid to pedagogical knowledge.

A wide variety of professional development programmes have been offered. Notable amongst these is Computing At School's Network of Excellence in Computer Science Teaching (see, e.g. Sentence et al. (2012) and Boylan & Willis (2015)). This network, funded by grant from central government, includes a number of lead schools, more than 400 'master teachers' supporting their colleagues locally, and ten regional centres, typically based in university computer science and education departments. In primary education, the Barefoot Computing project has reached more than 45,000 teachers through workshops, conceptual guides and exemplar lesson plans (British Telecom & Accenture Strategy, 2017).

However, Royal Society (2017) called for a massive increase in professional development for computing teachers, focussing particularly on the need to retrain those teaching computing to GCSE who lack any post A-level qualification in computing. The 2017 budget saw the announcement of £84m of public funding to establish a National Centre of Computing Education and at least 40 hours of professional development for presently unqualified secondary computing teachers. It is to be hoped that this significant public investment will be put to good use.

Resource materials

Oates (2014) made a persuasive case for the importance of text books and other teaching resources in embodying learning theory, providing a clear delineation of content, ensuring coherent progression, offering stimulation and support for reflection and encouraging 'expansive application'. However, government has, perhaps out of fear for being seen as anti-competitive, not led, or funded, the development of text books for computing, but has rather left this to the publishing industry, not-for-profit organisations active in this domain and individual teachers.

The British Educational Suppliers' Association and the Publishers' Association jointly developed guidelines for computing textbook publishers (BESA & The Publishers Association, 2015), but there is little evidence that those developing materials have been influenced by these.

A range of commercial materials have been produced, with some translated into foreign editions for countries following England's lead in implementing computer science for all. Alongside these commercial offerings high quality, free and liberally licenced materials are available from the BBC, Barefoot Computing, the Raspberry Pi Foundation and Code Club, amongst others. Computing At School provides an online resource sharing portal, by members, for members (and others): at the time of writing this hosts some 4,417 teaching resources.

Whilst online coding environments such as Scratch and locally installed IDEs for languages such as Python, C# and Java provide sufficient scope for pupils to meet the requirements of the National Curriculum, there is interest in 'physical computing' amongst English computing teachers. The BBC's micro:bit platform provides a readily accessible introduction to programming a simple microcontroller using block- or text-based languages, and the Raspberry Pi might now be seen as the default platform for any developers wishing to integrate programming with control and monitoring interfaces.

Incentives

Whilst a top down approach to curriculum change can control content and assessment, and with funding ensure suitable training and resources are in place, the actual implementation of any such change is in the hands of head teachers and teachers.

The curriculum change attracted much media attention, with most of those involved in teaching computing, and their head teachers, aware of the planned changes. Whilst data on take-up is hard to come by - as yet no reliable evaluation of the curriculum change has been undertaken at scale - anecdotal evidence suggests most primary schools are now teaching computing, and the majority of state funded schools now enter at least some students for the qualification at GCSE, albeit for relatively small groups of students on the whole.

Ministers, advocates for computing education in the technology industries and organisations such as Computing At School and Code Club have given a consistent message on the need to prepare students for life and work in a future in which digital technology might be expected to play a significant role. In general education, computer science has been recognised as a foundational discipline alongside physics and music as something *all* students should learn, rather than as elective subject for those likely to study it at university or pursue careers in the technology sector. This moral argument is a somewhat persuasive one for school leaders and parents.

Whilst all schools funded and supported through local education authorities are legally required to teach the content of the national curriculum, including computing, independent schools, and those funded directly by central government ('academies' and 'free schools') are not. The majority of secondary schools, and a significant minority of primary schools are now directly funded academies, and thus exempt from the requirement to teach computing. It seems that many have chosen to do so, at least thus far (Kemp, Wong, & Berry, 2016). Recent statements by the head of England's education inspectorate, Ofsted, indicate that there will be greater focus on the breadth and balance of schools' curricula (Spielman, 2017), which may see wider implementation of computing in academies.

Schools are also incentivised by published league tables of their pupils' performance on public examinations. Computer Science at GCSE can be included alongside the older sciences in the crucial Achievement 8 and Performance 8 accountability measures, and in the 'English baccalaureate' (EBacc) set of academic subjects (BCS, 2012). In 2015, only 1,126 students took computing as one of three sciences, which satisfies the requirements for these accountability measures, although a further 10,849 took computing as one of *four* sciences. On the other hand, analysis suggests that Computer Science at GCSE is harder than most other subjects (Thomson, 2016), and this may disincline school leaders from entering all but the brightest and best of their students for the subject. It is hoped that once the computing curriculum between ages 5 and 14 is fully bedded down, GCSE performance will be comparable to other EBacc subjects.

Another key driver for schools, and students themselves, is university entrance. The Russell Group of leading UK universities recognises that A level computer science is a useful qualification for a number of science, engineering, medical and social science disciplines (Russell Group, 2016), which is unsurprising given the role that programming has come to play in these fields. However, computer science is not listed as a 'facilitating subject' - those which are regarded as opening doors 'to more degrees and more professions than others'. This may be because few computer science degrees currently require A level computer science as an entry requirement; indeed Cambridge University's Computer Laboratory appears to discourage applicants from offering this:

> The newer Computing A-Levels resemble much of what we teach in our first year, so it is worth remembering that studying the A-

> Level will currently result in repetition of material at University. Some students (and Directors of Studies) therefore prefer to broaden their horizons and study related subjects such as an extra science. (Computer Laboratory, 2015)

Concluding remarks

It is hard not to admire the courage of ministers and others who took the decision to make computer science part of the English national curriculum from age five to sixteen. Despite the relative haste with which the curriculum content was developed, and the reluctance of government to lead its implementation, schools, teachers and pupils have responded positively to this challenge, and the comparison with what pupils learnt in the former ICT curriculum is very favourable. Royal Society (2017) describe the current state of computing education as 'patchy and fragile'. They argue that

> future development and sustainability depend on swift and coordinated action by governments, industry, and non-profit organisations. Neglecting the opportunities to act would risk damaging both the education of future generations and our economic prosperity as a nation.

For other jurisdictions considering or about to embark on a similar change, a more coherent, joined-up approach to the *implementation* of a computer science curriculum, considering teaching, assessment, resources, training and incentives alongside the design of curriculum content, would result in a more widely adopted, and more robust computer science education for all.

References

BCS: *The case for Computer Science as an option in the English Baccalaureate*. Swindon: BCS, 2012.

BESA, The Publishers Association: *Guidance for the Publishing of Computing Teaching Resources*. London: BESA / The Publishers Association, 2015.

Boylan M., Willis B.: *Independent study of computing at School Master Teacher programme*. Sheffield: Sheffield Hallam University, 2015.

British Telecom, Accenture Strategy: *Tech know-how. The new way to get ahead for the next generation*. London: BT plc, 2017.

Computer Laboratory: Frequently Asked Questions. 2015:

Csizmadia A., Curzon P., Humphreys S., Ng T., Selby C., Woollard J.: *Computational thinking: A guide for teachers*. Cambridge: Computing At School, 2015.

Department for Education, National College for Teaching & Leadership: *Initial teacher training census for the academic year 2015 to 2016, England*. London: DfE, 2017.

DfE: *The national curriculum in England*. Lodnon: DfE; DfE, 2013.

Dorling M., Walker M.: *Computing Progression Pathways*. Cambridge: Computing At School, 2014.

Fossati D., Guzdial M.: The Use of Evidence in the Change Making Process of Computer Science Educators. Special Interest Group on Computer Science Education (SIGCSE) Conference 2011: 685–690.

Grover S., Cooper S., Pea R.: Assessing computational learning in K-12. Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE '14 2014: 57–62.

Kemp P.E., Wong B., Berry M.G.: *The Roehampton Annual Computing Education Report*. London: University of Roehampton, 2016.

Koehler M., Mishra P.: What happens when teachers design educational technology? The development of Technological Pedagogical Content Knowledge. Journal of Educational Computing Research 2005: 32: 131–152.

Mcintosh J.: *Final report of the Commission on Assessment without Levels*. London: DfE, 2015.

Oates T.: *Could do better: Using international comparisons to refine the National Curriculum in England*. Cambridge: Cambridge Assessment, 2011A.

Oates T.: The Framework for the National Curriculum A report the Expert Panel for the National Curriculum review. Deparment for Education 2011B: 1–77.

Oates T.: Why textbooks count. Cambridge Assessment 2014: 1–23.

Royal Society: *After the reboot : computing education in UK schools*. London: The Royal Society, 2017.

Russell Group: *Informed Choices*. London: Russell Group, 2016.

Sentance S., Csizmadia A.: Teachers ' perspectives on successful strategies for teaching Computing in school. Ifip Tc3 2015: 1–11.

Sentence S., McNicol A., Dorling M., Crick T.: Grand challenges for the UK: Upskilling teachers to teach computer science within the secondary curriculum. ACM International Conference Proceeding Series 2012: 82–85.

Spielman A.: Recent primary and secondary curriculum research. 2017:

Straw S., Bamford S., Styles B.: *Randomised Controlled Trial of Code Clubs*. Slough: NFER, 2017.

Taub R., Ben-Ari M., Armoni M.: The effect of CS unplugged on middle-school students' views of CS. ACM SIGCSE Bulletin 2009: 41: 99.

Teaching Agency: *Subject knowledge requirements for entry into computer science teacher training*. London: DfE, 2012.

Tedre M., Denning P.J.: The long quest for computational thinking. Proceedings of the 16th Koli Calling International Conference on Computing Education Research - Koli Calling '16 2016: 120–129.

Thomson D.: Which are the most difficult subjects at GCSE? 2016:

Yelland N.: Mindstorms or a storm in a teacup? A review of research with Logo. International Journal of Mathematical Education in Science and Technology 1995: 26: 853–869.