Computing at school in the UK

Simon Peyton Jones, Microsoft Research
Bill Mitchell, BCS Academy of Computing
Simon Humphreys, Computing At School

April 2013 v5
Under review by CACM

1. Introduction

In almost every country in the world the realisation has dawned that our education in computing is failing our children. Our young people should be educated not only in the *application* and *use* of digital technology, but also in *how it works,* and *its foundational principles*. Lacking such knowledge renders them powerless in the face of complex and opaque technology, disenfranchises them from making informed decisions about the digital society, and deprives our nations of a well-qualified stream of students enthusiastic and able to envision and design new digital systems.

Can anything be done, given the enormous inertia of our various countries' educational systems? Sometimes, yes, it can. After a decade of stasis, change has come to the UK. Over the last 18 months, there has been a wholesale reform of the English school computing curriculum, and substantial movement in Scotland and Wales, and it seems likely that computer science will, for the first time, become part of every child's education. This change has been driven not by institutions or by the government, but by a grass-roots movement of parents, teachers, university academics, software developers, and others. A key agent in this grass-roots movement — although not the only one, as Section 3 describes — is the Computing At School Working Group (CAS)[1].

In this paper we describe the recent developments in England. More importantly, we reflect on the lessons we have learned, in a conscious effort to develop ideas that may prove helpful in other countries.

(Throughout we use the term "school" in its UK sense, to mean primary and secondary school, ages 6-18, called K-12 in the USA.)

2. Computer science as a school subject

Here is our central thesis:

> **Computer science is a proper, rigorous school subject discipline, on a par with mathematics or chemistry, that every child should learn from primary school onwards.**

This is a pretty radical claim. Head teachers, parents, civil servants, and politicians all think of computer science, if they have heard of it at all, as a niche subject appropriate for socially-challenged male university students, but way beyond the scope of a school, let alone *primary* school. In so far as they think of computing as a school subject at all, it is as a technology subject equipping students with useful life or employment skills, and certainly not as a subject discipline like physics.

The key to making progress in the UK was to articulate the case for computer science as a foundational subject discipline in its own right.

[1] http://www.computingatschool.org.uk

2.1. Computer science as a discipline

What do we expect of a school education? Education should enhance pupils' lives as well as their life skills. It prepares young people for a world that doesn't yet exist, involving technologies that have not yet been invented, and that present technical and ethical challenges of which we are not yet aware.

To do this, education aspires primarily to teach **disciplines** with long-term value, rather than **skills** with short-term usefulness, although the latter are certainly useful. A "discipline" is characterised by:

- **A body of knowledge**, including widely-applicable ideas and concepts, and a theoretical framework into which these ideas and concepts fit.
- **A set of techniques and methods** that may be applied in the solution of problems, and in the advancement of knowledge.
- **A way of thinking and working** that provides a perspective on the world that is distinct from other disciplines.
- **Longevity**: a discipline does not "date" quickly, although the subject advances.
- **Independence from specific technologies**, especially those that have a short shelf-life.

Computer science is a discipline with all of these characteristics. It encompasses foundational principles (such as the theory of computation) and widely applicable ideas and concepts (such as the use of relational models to capture structure in data). It incorporates techniques and methods for solving problems and advancing knowledge (such as abstraction and logical reasoning), and a distinct way of thinking and working that sets it apart from other disciplines (computational thinking). 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.

Seen from this point of view, our critique of current education in digital technology is precisely that it focuses too much on *technology*. We teach our children how to use office productivity software, or other undeniably useful skills in using computer systems, but (at school level) we have lost sight of, or perhaps never even considered, the underlying discipline. In Britain the school subject is even called Information and Communication Technology, which focuses attention on technological artefacts rather than on principles and ideas. We have fatally entangled the message (ideas, principles) with the oh-so-seductive medium (devices, software, even programming).

2.2. Computer science at primary school?

It is no longer hard to make the case that computer science is a discipline, but one might still ask this: is computer science a discipline that **every** child should learn, or simply one in which some suitably-motivated young people should be able to specialise at some stage?

We emphatically believe the former. Why do we teach every child elementary physics at primary school? Not because most of these young people will become physicists! Only a tiny fraction will do so. Rather, because we live in a world governed by physical laws, and some knowledge of science is essential to being an empowered citizen.

It is just the same with computer science: adults who know nothing about how digital systems are designed, and the principles of their operation, are doomed to be the powerless slaves of a mysterious and opaque technology. So it is important for *every* child to have an elementary understanding of computer science, just as they have an elementary understanding of maths. Some will take to it better than others, but all will live more empowered lives as a result. In Douglas Rushkoff's famous phrase, the choice is simple: program or be programmed (Rushkoff, 2010).

Not only that, but there is good reason to believe that while every subject claims to teach students to think, computer science really does. The case for this goes back to Papert's "Mindstorms (Papert, 1980), whose worked with Piaget in understanding how young children learn, and advocated the use of computers in structuring understanding in maths and physics. This book has had a tremendous influence on teaching in primary and lower secondary schools, not just in the adoption of Logo in schools in the 1990s almost unilaterally, but also in the development of robot-like "toys" such as the Bee-Bot for early years children to learn simple programming and control structures.

Is it too ambitious to expect eight-year-olds to learn computer science? By no means. We have ample evidence that they have a great appetite for doing so. CS Unplugged[2] includes lots of materials that can be used with young children and evidence that they find these engaging. Code Club[3] offers extra-curricular opportunities in computer science for young children; although only launched in 2012, there are already over 600 Code Clubs in primary schools in the UK, all run by volunteers. There is an increasing amount of research emerging about how students will learn computational principles in primary school; work by Bitto and Minola (2013) and Serafini (2011) give just two examples.

2.3. A curriculum for computer science

If we believe that computer science should indeed be a school subject, it is incumbent on us to explain exactly what that might mean, by explaining the significance of the discipline for school pupils, and by laying out what material might constitute a computer science school curriculum. It is surprisingly hard to find articulate expositions of such a curriculum, so we made an attempt ourselves: "Computer science: a curriculum for schools". Here are its opening paragraphs:

> *Computer Science is the study of principles and practices that underpin an understanding and modelling of computation, and of their application in the development of computer systems. At its heart lies the notion of computational thinking: a mode of thought that goes well beyond software and hardware, and that provides a framework within which to reason about systems and problems. This mode of thinking is supported and complemented by a substantial body of theoretical and practical knowledge, and by a set of powerful techniques for analysing, modelling and solving problems.*
>
> *Computer Science is deeply concerned with how computers and computer systems work, and how they are designed and programmed. Pupils studying computing gain insight into computational systems of all kinds, whether or not they include computers. Computational thinking influences fields such as biology, chemistry, linguistics, psychology, economics and statistics. It allows us to solve problems, design systems and understand the power and limits of human and machine intelligence. It is a skill that empowers, and that all pupils should be aware of and have some competence in. Furthermore, pupils who can think computationally are better able to conceptualise and understand computer-based technology, and so are better equipped to function in modern society.*
>
> *Computer Science is a practical subject, where invention and resourcefulness are encouraged. Pupils are expected to apply the academic principles they have learned to the understanding of real-world systems, and to the creation of purposeful artefacts. This combination of principles, practice, and invention makes it an extraordinarily useful and an intensely creative subject, suffused with excitement, both visceral ("it works!") and intellectual ("that is so beautiful").*

[2] http://csunplugged.org/
[3] http://www.codeclub.org.uk/

By design, the CAS curriculum looks like a fairly traditional CS programme of study (data structures, algorithms, networks, architecture, abstraction, and so on). The whole point is that it is sufficiently independent of the present moment that it would have been recognisable ten years ago, and should still be recognisable in ten years time.

The CAS curriculum is careful to make two distinctions. First, **computer science is not primarily about computers**. The famous aphorism “Computer science is no more about computers than astronomy is about telescopes” , widely attributed to Dijkstra, slightly overstates the case, but it has the right idea. The CAS curriculum makes this distinction by devoting a chapter to developing Jeanette Wing’s influential idea of **computational thinking**, as something *people* do rather than something *computers* do (Wing, 2006). There is now a large literature on computational thinking ,(Bundy 2007, Lu and Fletcher, 2009, Barr and Stephenson, 2011, Perovic, 2010) being exemplars, and, although it is hard to find crisp, meaningful definitions, the term clearly has resonance. (The reader may enjoy looking our own attempt in the CAS Curriculum.) Another very insightful source of ideas is the CS Unplugged material from New Zealand[4]; by teaching computer science without using computers at all, it makes the distinction absolutely clear.

Second, **computer science is not the same as programming**. As CAS’s message has gained traction, the British newspapers have featured many articles with titles like “Why we must teach our kids to code” or “Coding is the new Latin”. The danger here is that teachers and politicians conflate the two, put in place Java courses for 14-year-olds, and declare the problem solved. We have found a helpful analogy in science. Physics without lab-work would be a mere shadow of itself, just as computer science without programming is almost an oxymoron. But no one would dream of turning students loose in a physics lab without teaching them physics! Programming is certainly where the rubber first hits the road, but teachers must be aware of the principles they are trying to convey through the medium of programming.

2.4. The economic argument

There is another popular argument for a good school computer science education, namely the economic imperative. Technology firms are crying out for graduates who are adept at computational thinking, are not afraid of programming, and who understand how the digital world is built (European Commission, 2010). Nations that teach their young people that computers are boring risk pinching off this supply pipeline at a very early stage, with long-lasting economic consequences.

This economic argument plays well with government, and has the great merit of being true, but we have found that it carries people’s *minds* rather than their *hearts*. Producing economically-productive citizens is not the sine qua non of education. We have found that the (powerful) economic imperative plays much better in a supporting role to the main educational argument of this section.

2.5. Summary

It is ambitious to present computer science as a foundational discipline for all, rather than as an economically-valuable specialism for the few. But we not only believe it to be true, but have also found that it is an idea that resonates strongly with almost everyone. It makes sense of our gut feel that the explosion of computation in our lives is of fundamental importance.

In focusing our attention broadly, on every level of school education including primary school, CAS has taken a very different approach to other countries. For example, in the USA, most attention is focused on the Computer Science AP exam, taken at the end of high school shortly before going to

[4] http://csunplugged.org

university. Outside the AP exam, which itself is taken only by a minority, there is little systematic attempt to teach Computer Science at school. It would be foolish to argue about which approach is "better" —the particularities in each country vary dramatically — but the differences may be instructive.

There is another important benefit to the "start early" approach. Every reader of CACM will be well aware of the horrendous gender imbalance in our discipline. Once gender stereotypes are established, it seems that they are extraordinarily difficult to dislodge. Starting computing early, in primary school, before these stereotypes have formed, seems to be the most plausible way to address this issue.

3. The birth of CAS

The Computing At School working group (CAS) was born at Microsoft Research Cambridge, at a meeting in 2008. Our fundamental goal is to establish computer science as a proper, rigorous, high status school subject discipline —the "fourth science" — and to build a network that supports teachers as they engage with computer science in their classrooms.

3.1. Partnership with BCS

CAS is, first and foremost, a grass-roots group of *individuals*, not of institutions or companies. Membership is free, is open to everyone, and is very broad, including teachers, parents, governors, exam boards, software professionals, members of professional societies, and university academics.

Serendipitously, at the very same time the BCS (the British Computer Society, the UK's equivalent of the ACM), was undergoing an internal renaissance, which led to the formation of the BCS Academy of Computing in 2009. The Academy plays the role of a learned society, and is intended to have deep engagement with research and education.

At this point the BCS Academy had legitimacy but relatively little activity, while CAS had a tremendous buzz of activity but no institutional legitimacy. The solution was obvious: with some trepidation, CAS formed a collaborative partnership with BCS. The partnership has turned out to be resoundingly beneficial for both parties. BCS has given CAS substantial resources and air cover, while CAS has become one of BCS's highest-impact and most visible activities.

3.2. A thousand flowers

We were not the only ones to feel frustrated about ICT. A number of other independent groups have also sprung up in the UK over the same period. Rather than try to absorb them all in one mega-movement, we have sought to work together in partnership rather than competition, for common goals. Prominent among them are:

- **Cs4fn**[5] is a print magazine and supporting web site about "computer science for fun". For example, magic tricks each with a computational component, feature prominently.
- **Apps for Good**[6] helps young people at school create apps, stressing entrepreneurial as well as technical skills.
- **Young Rewired State**[7] runs hack days for teenagers.
- **YouSrc**[8] is a web site aimed at supporting programming tuition, built by Paul Clarke, an IT professional. It is specifically adapted for the workflow of the UK education system.

[5] http://www.cs4fn.org/
[6] http://appsforgood.org/
[7] https://youngrewiredstate.org/
[8] http://www.yousrc.com/

- The **Raspberry Pi**[9] credit-card-sized computer, designed specifically for impecunious children, has been astonishingly successful. The team hoped to sell 10,000, but have now hit well over 1,000,000.
- **Code Club**[10], launched by Clare Sutcliffe and Linda Sandvik in April 2012, supports schools in running after-school programming clubs in primary schools. In the first seven months they have over 400 clubs running.
- **Technocamps**[11] is an initiative run from Swansea University, funded by an EU grant, to give computer science lessons to Welsh school pupils.

We also draw inspiration from similar groups in other countries, such as CS Unplugged[12] in New Zealand, in the USA the AP CS Principles project[13], Codeacademy[14]**,** Bootstrap[15], Georgia Computes![16], and Exploring Computer Science[17] in Los Angeles. All this creative volunteer-driven activity is a sign of a movement whose time has come.

3.3. The CAS community

CAS has developed into a real community. From four people in 2008, it now has 3,700 members and is growing at about 400 members/month. There are several sorts of glue that hold the group together:

- First, a clearly articulated purpose, described at the start of this section. It sounds obvious, but being absolutely clear what you stand for is powerful glue.

- Face to face meetings are much more effective than virtual ones for generating a shared sense of purpose and enthusiasm, so we started local “hubs” for (mainly) teachers to share ideas. Initially we started four hubs but there now 57, spread right across the country. Each is run by a volunteer leader, with very modest central support; they typically meet once a term.

- We held our first national CAS Teachers Conference in 2010, with much trepidation, but again the “buzz” generated by the face-to-face interaction of 200+ school teachers was overwhelming, and the conference has become an annual event (generously hosted by the Birmingham Computer Science department). Most of the day consists of dozens of workshops, many of them run by teachers themselves.

- Roger Davies (himself a teacher) launched a termly CAS Newsletter. Worthy causes often have newsletters full of descriptions of worthy meetings. Somehow Roger managed the trick of building a newsletter that you might actually want to read, and repeating the trick term after term[18]

- Until 2012 CAS was simply a Google email group. We had repeatedly discussed developing some kind of CAS-specific forum, where members could share ideas, upload resources, and comment on and improve those uploads. We investigated off the shelf solutions but none seemed right for us. Finally Michael K**ö**lling and Neil Brown, at the University of Kent,

[9] http://www.raspberrypi.org
[10] http://www.codeclub.org.uk/
[11] http://www.technocamps.com/
[12] http://csunplugged.org
[13] http://www.collegeboard.com/html/computerscience
[14] http://codecademy.com
[15] http://www.bootstrapworld.org/
[16] http://gacomputes.cc.gatech.edu/
[17] http://www.exploringcs.org/
[18] http://www.computingatschool.org.uk/

> decided to clone their existing Greenroom site (a community site for people using Greenfoot, a Java programming environment for schools). The result was CAS Online, which as well as forums now supports uploading resources, commenting on them and improving them, maps showing where members and hubs are, listings of events, and more.

CAS has grown explosively over the last four years. From four members in 2008, CAS had grown to 3,700 by April 2013. About two thirds are school teachers, but there are a substantial number of IT professionals, software engineers, academics, and others.

4. Breakthrough in the UK

The year 2012 was a breakthrough year in the UK, when computer science as a school subject moved out of the shadows into the media, the discourse of politicians, and the review of the national curriculum. Here is how events have unfolded.

In August 2011 Eric Schmidt, the chairman of Google, gave a now-famous speech in Edinburgh. *"I was flabbergasted to learn that today computer science isn't even taught as standard in UK schools,"* he said. "*Your IT curriculum focuses on teaching how to use software, but **gives no insight into how it's made**.*" A few sentences, publicly spoken by a powerful figure can catapult a topic into the public domain, and Schmidt's speech did precisely that. Within weeks the Prime Minister was saying "*I think Eric Schmidt is right... we're not doing enough to teach the next generation of programmers. One of the things you hear from the businesses here in Tech City is **"I don't just want people who are literate in technology, I want people who want to create programs"**, and I think that's a real wake up call for us in terms of our education system.*"

In February 2011 the NextGen Skills report [19], authored primarily by the games and creative technology industry, argued explicitly for Computer Science to form part of the school curriculum. Also during 2011 the Royal Society — Isaac Newton was a member — was working on a substantial report focused specifically on computing in UK high schools. The decision to run the project was led by Professor Steve Furber (of BBC Micro and ARM fame), inspired in part by CAS's activity and writings. The report was published in January 2012, and gave well-evidenced support to the analysis of Sections 2 and 3.

Then in January 2012 the submarine finally surfaced. Michael Gove, the Secretary of State for education, gave a major policy speech in which (to our astonishment and delight) he said this

> ***We're encouraging rigorous Computer Science courses.*** *The new Computer Science courses will reflect what you all know: that Computer Science is a rigorous, fascinating and intellectually challenging subject. Computer Science requires a thorough grounding in logic and set theory, and is merging with other scientific fields into new hybrid research subjects like computational biology.*
>
> *Although individual technologies change day by day, they are underpinned by foundational concepts and principles that have endured for decades. Long after today's pupils leave school and enter the workplace – long after the technologies they used at school are obsolete – the principles learnt in Computer Science will still hold true."*

These words could have been written by CAS. Even more eye-catchingly, he withdrew the Programme of Study for ICT, whilst keeping the subject a compulsory part of the National Curriculum to age 16, thus leaving schools free to teach whatever they liked. This triggered a year of

[19] http://www.nesta.org.uk/publications/assets/features/next_gen

uncertainty and debate about just what the future National Curriculum for ICT would look like, and whether there would be one at all.

4.1. School qualifications in Computer Science

February 2012 also saw a major step forward in qualifications. In Britain, GCSEs (General Certificate of Secondary Education) are qualifications taken by most school pupils at age 16. Students typically take between 5 and 12 GGCSEs, before specialising in their A-level subjects. The two-year period leading up the age-16 watershed (also called Key Stage 4) is entirely dominated by study for those qualifications. If there is no GCSE, you can't study it; it's that simple. And until 2010 *there was no GCSE whatsoever in Computer Science*, although over a dozen Key Stage 4 qualifications in ICT were available.

GCSEs are offered by so-called *awarding bodies*, independent non-profit organisations, policed by the Government's Ofqual regulator. CAS had been talking to all five awarding bodies, and OCR courageously launched a pilot Computing GCSE in September 2010. Then in February 2012 all four other awarding bodies also announced plans for a GCSE in Computing or Computer Science. This was a huge step forward, breaking the chicken-and-egg loop: no GCSEs means no students studying Computer Science, but no students studying Computer Science means that awarding bodies are unsure if the market is there.

4.2. The National Curriculum Review

Coincidentally but fortuitously, the Department for Education was in the midst of a full-scale review of the entire National Curriculum. In the latter half of 2012, the Department for Education (DfE) was engaged in writing a new Programme of Study for each subject, but for ICT they took an extremely unusual approach: they invited the BCS and Royal Academy of Engineering to coordinate a broad, multi-stakeholder process of writing the new Programme of Study for ICT. This invitation reflected a degree of trust between the DfE and BCS/CAS, that had built up over the previous year of meetings and informal policy papers.

The timescale was extremely tight: the first invitation arrived in early September, and the final version was handed over at the end of November, the result of two successive working parties and a good deal of community consultation. The DfE revised the draft somewhat, interestingly to strengthen the computer science component, and published the result February 2013 for public consultation. It will be revised in the light of that feedback, and published by September 2014, for first implementation a year later.

The aims of the draft Programme of Study for Computing are as follows: *"The National Curriculum for computing aims to ensure that all pupils:*

- *can understand and apply the fundamental principles of computer science, including logic, algorithms, data representation, and communication*
- *can analyse problems in computational terms, and have repeated practical experience of writing computer programs in order to solve such problems*
- *can evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems*
- *are responsible, competent, confident and creative users of information and communication technology."*

This is a major, enduring outcome: these aims clearly establish computer science in the school curriculum from primary school onwards, as part of a balanced curriculum involving both computer science and digital literacy. Moreover, the DfE proposes to re-title the subject from "ICT" to "Computing", again signalling a qualitative shift in content.

4.3. The English Baccalaureate

One particularly influential performance measure for schools is the English Baccalaureate. The EBacc is not a new qualification; a student achieves an EBacc if he or she gets a grade C or better in English, maths, two sciences, a humanity (history or geography), and a language. League tables showing what proportion of a school's cohort achieve an EBacc are now published annually.

Published performance measures exert a powerful influence over school behaviour. For example, when the EBacc was first introduced, enrolments for modern languages *doubled*.

Whatever you think about the merits of the EBacc — and it is patently a very blunt instrument — it is clear that computer science should be in it.   And now it is. In January 2013, year to the day after his first speech, he announced that "computer science will be added to the list of separate science options (so there are now four separate sciences instead of the traditional three) in the EBacc". The phrasing is significant: computer science is now officially joins physics, chemistry, and biology as the fourth science.

4.4. Training teachers

It is all very well to specify Computer Science as part of the curriculum, but quite another thing to find enough teachers able and willing to deliver it. This divides into two parts: attracting well-qualified graduates into the profession and training them to teach Computer Science; and offering support and training to the existing cohort of ICT teachers.

On the first of these issues, the Teaching Agency, a government agency responsible for pre-service teaching, started to put some wood behind the arrow of Michael Gove's sentiments in mid 2012, by announcing scholarships of £20,000 to support computer scientists who wanted to train to become teachers.   They also announced that in future all training course for ICT teachers, offered by some 60-odd institutions across the country, would only be funded if they included "Computer Science" in the course title. These announcements are significant for two reasons. First, they mention Computer Science by name, a substantial indication of its recognition within the educational establishment. Second, they put it in the same category as other high-status science subject like Physics (there is a similar scholarship scheme in place already for Physics). The third change is that the Teaching Agency published "Subject Knowledge Entry Criteria[20]" for Computer Science entrants to teaching, which has hitherto been unheard of in any other subject.

4.5. A Network of Excellence in Teaching Computer Science

It is not enough just to train new teachers. A huge issue is the training of existing ICT teachers, many of whom come from non-technical backgrounds. There are 3,500 secondary schools in Britain, and over 20,000 primary schools, so that means training a *lot* of existing teachers. Although these teachers recognise themselves to be under-qualified or out of practice, many of them are eager for change. They rightly feel exposed and vulnerable, as the proposed new curriculum makes demands that they will find it hard to meet --- but most are highly motivated, and more than willing to learn.

We began, in the Spring of 2012, by forming the Network of Excellence in Teaching Computer Science, whose primary mission is to support teachers in learning Computer Science themselves, and

[20] http://media.education.gov.uk/assets/files/pdf/s/subject%20knowledge%20requirements%20for%20entry%20into%20cs%20teacher%20training.pdf

to support them in teaching it in their classrooms. We wrote to every secondary school in the country to invite them to join; over 500 schools did so in the first six weeks.

But who will teach the teachers? We have two main routes. First, the Computer Science departments of our universities. For the last two decades university CS departments have been utterly disengaged from school ICT, because the subject was of no interest or relevance to them. Now there is a real prospect that CS will be taught to school children, every university academics suddenly have a real stake in what is happening at school. So our idea is simple: to invite university CS departments to offer courses in elementary CS to the school teachers in their area, not as a revenue-generating opportunity, but more or less pro-bono as a way to reinvigorate our discipline at school. Almost all the 110 CS departments in the UK have joined the Network, including heavyweights like Manchester, Cambridge, and Imperial. Several have already started putting on courses.

Second, we are identifying Master Teachers, school teachers who are already confident in teaching Computer Science, and offering them modest funding to put on training for other teachers in their area. This has proved to be a very quick win, with relevant, local, teacher-credible training available within weeks or months. We have started with 30 Master Teachers, and hope to have 600 in five years time. That is enough to make a real difference. The Network of Excellence is also encouraging some schools to become Lead Schools (over 100 have volunteered) with an institutional commitment to help other schools teach CS.

Realistically, universities lack both capacity and motivation to engage directly with 20,000 primary schools, so we envisage that training for primary school teachers will come mainly from their local secondary schools; the Digital Schoolhouse Project[21] is an excellent example of this kind of primary/secondary linkage in action.

A third huge, largely-untapped resource is the reservoir of expertise and goodwill that lies in our software developers and IT professionals. They are numerous, motivated, and keen to help; but they are busy, and have little idea of how to teach school children. After-school programming clubs are one productive way in, as Code Club[22] (itself led by two IT professionals) has shown, but we are also thinking about 1-1 mentoring partnerships and, more ambitiously, actual classroom teaching like the TEALS project[23] in the USA.

All of this is at a very early stage. Launching a new school subject from a near-zero base is no easy matter. Will the universities really be willing to offer professional development courses for free? Will teachers be willing to learn? Will head teachers be willing to give them time off to learn? In primary schools, there are few specialist ICT teachers; will primary schools really be able to teach Computer Science? There are no easy answers, and we do not underestimate the challenge. But there is such a sense of excitement and goodwill that it must be possible.

A particular challenge for CAS is to scale up from a bunch of volunteers running on adrenaline and enthusiasm, to an organisation capable of coordinating the training of the nation’s computer science school teachers, a challenge we are grappling with as we write.

4.6. Summary

These policy changes did not, of course, come out of the blue. CAS was formed in early 2008, and the four years after that involved many, many meetings. Education is dense with stakeholders: subject associations, head teachers associations, awarding bodies, professional societies,

[21] http://www.digitalschoolhouse.org.uk/
[22] http://www.codeclub.org.uk/
[23] http://tealsk12.org/

educational suppliers, think tanks, educational foundations and trusts, and so on. And there is just no substitute for meeting them face to face, especially for the first meeting to establish trust and common ground.

Perhaps surprisingly, we found surprisingly little disagreement in principle. The first author had the repeated experience of going to meetings prepared to assault the castle, only to discover that the door was ajar: "Oh yes, we agree with that." The castle has no defenders, just huge inertia.

Nevertheless, 2013 feels like a real inflection point. The air war is won: the DfE now understands and supports the importance of Computer Science, as a school subject that every child should learn at least at an elementary level, with opportunities for later specialisation.

But that was the easy part! Now the ground war begins: school by school, head teacher by head teacher, we must make the case, convey the vision, offer support and teaching materials, and train teachers. This process will take years, not months.

Meanwhile, CAS has moved from being a lonely voice at the bottom of a deep mine shaft, to a trusted group that the government consults for advice in this policy area. That in itself brings new challenges. When we began we could argue straightforwardly for a single issue: the importance of CS as a school subject. Now that we have become influential, there is a danger that our enthusiasm for CS will be mis-interpreted as a dismissal of all of ICT, and that the pendulum will swing too far. What we need is a proper balance of the discipline of computer science with the purposeful application of information technology to create digital artefacts, both programs, systems, and a range of media. None of us has a monopoly on truth, and reasonable people might differ on what exactly the "proper balance" is. But the world has changed: it is now accepted across the UK that CS can and should be taught at school, and the debate has now moved on to what, and how, and in what balance with other aspects of digital literacy.

5. Reflections

Although our journey is far from over, the landscape is quite different now from when we began. In this section we reflect on what we have learned.

- **One simple message**. CAS exists to do just one thing: to establish computer science as a proper school subject, from primary school onwards. We express this goal in many different ways, but it is still one goal. We are not about teamwork skills, though a computing education builds teamwork. We are not about using information technology to transform teaching and learning in other subject, important though that is. We are all for linkages between maths and computer science education, but we can only really explore that when we have nailed down what the computer science part might mean.

- **Explain what computer science is**. We need to find ways to explain what our discipline is, in ways that make sense to parents, civil servants, and politicians, not just to the technical community. Clearly distinguishing *disciplines* from *skills and technologies* is helpful. The term "computational thinking" resonates with many people, because it plainly something *people* do, not something *computers* do. (NB: CT needs to be articulated in some detail, lest the resonance be only superficial.)

- **Start with primary**. Although it is wildly ambitious to seek to establish computer science in school education starting even in primary school, it is a natural consequence of seeing computer science as an educationally-foundational discipline (like maths, or physics), rather than as a niche specialisation. Presenting the argument in this principled, educationally-founded way is more convincing than the more instrumental argument that computer scientists are crucial for our nation's economy, true though that might be. The two arguments, educational and instrumental, work powerfully together, and there is ample evidence that primary-aged children can understand and enjoy serious computer science.

- **Diversity: not just teachers**. Right from its foundation, CAS has been open to any individual who wants to fix the UK's education system. As well as school teachers, CAS members include university academics, school governors, parents, software developers and IT professionals, educational advisers, members of professional societies, and so on. This rich diversity of membership make it clear that CAS speaks for the discipline, rather than for one particular group of stakeholders (teachers, say, or employers).

- **Speak with one voice**. People in the educational world, present authors not excluded, have an inherent tendency to form a circular firing squad. Our children's education is important to us, and it is all too easy to focus on our own disagreements. To have an impact on the world we must instead focus on what unites us, and speak with one voice. So, complementing CAS's laser focus on computer science as a school subject, we have engaged with our colleagues in related areas — and indeed it turns out that the area is dense with stakeholders, many rich in wisdom and experience.

- **There is no "them"; there is only us**. CAS has virtually no funding. There is no smoothly-functioning headquarters with dozens of staff members working away. We do now have a couple of staff FTEs, but to a first approximation CAS is all periphery and no centre. This is curiously liberating. Everyone knows that there is no point in saying "CAS should do X or Y", because the rejoinder will surely be "Excellent, can you make that happen?". CAS functions only because its members are proactive: see an opportunity and get on with it.

  Of course, there is a real limit to what a loosely-coupled bunch of volunteers can do. At some point you need resources to pay staff, and we are reaching that point with the Network of Excellence. But still, a culture of empowerment is a good basis for a grass roots organisation.

- **Partnership not competition.** We have already mentioned the efflorescence of small organisations, in the UK and elsewhere, dedicated to inspiring young people with the joy and beauty of computer science. Again the danger is to see a finite sized cake of mind-share, with each organisation competing for a slice of that cake. Rather we should celebrate each other's achievements and talk each other up: the cake is ever growing and the more we work together, the bigger it grows.

- **There is no One True Way**. When CAS began, we were attracted by the idea of identifying a particular programming language, or sequence of languages, and developing a range of teaching materials based on these choices. For example, perhaps we begin with Kodu, move on to Scratch, and then to Greenfoot.

  We found that it was much better instead to focus advocacy around the principles of computer science, and encourage a rich diversity of approaches to teaching it. If a particular teacher knows Python, use that. If another teacher has the idea of teaching computing through HTML, CSS, and JavaScript, more power to her. It is the message that is important, not the medium.

6. Conclusion

The UK's education system in computing at school level is a work in progress. The easy part is done, but the ground campaign has only just begun. Nevertheless, the situation is radically different compared to even two years ago, and this may perhaps give encouragement and heart to similar campaigns in other countries, as well as one model for how change can come about.

Acknowledgements

We gratefully acknowledge the help of Neil Brown, Amanda Clare, Quintin Cutts, Peter Buneman and Sue Sentance in improving this paper.

References

Barr, V. & 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,* vol. 2, no. 1, pp. 48-54.

Bitto, D. & Mirolo, C. 2013, " Archaeology of Information in the Primary School" in *Informatics in Schools. Sustainable Informatics Education for Pupils of all Ages*, eds. I. Diethelm, R. Mittermeir & T., Springer Berlin Heidelberg, , pp. 115-126.

Bundy, A. (2007). Computational thinking is pervasive. Journal of Scientific and Practical Computing Vol. 1, No. 2 .

European Commission (2010). Evaluation of the implementation of the communication of the European Commission E-Skills for the 21st Century, Tobias Hüsing Werner B. Korte, European Commission Oct 2011

Lu, J.J. & Fletcher, G.H.L. 2009, "Thinking about computational thinking", *Proceedings of the 40th ACM technical symposium on Computer science education*ACM, New York, NY, USA, pp. 260.

Naughton, J. (2012). Why all our kids should be taught how to code. The Observer. Available at: http://www.guardian.co.uk/education/2012/mar/31/why-kids-should-be-taught-code

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc.

Perkovic L., Settle A., Hwang S. and Jones, J. (2010). A Framework for Computational Thinking across the Curriculum, Proceedings of the 2010 Conference on Innovation and Technology in Computer Science Education, 2010, 123-127.

Rushkoff, D. (2010). Program or be programmed: Ten commands for a digital age. Or Books.

Serafini, G. 2011, "Teaching programming at primary schools: visions, experiences, and long-term research prospects", *Proceedings of the 5th international conference on Informatics in Schools: situation, Evolution and Perspectives*Springer-Verlag, Berlin, Heidelberg, pp. 143.

Wing, J.M. (2006). “Computational Thinking,” Communications of the ACM, Vol. 49, No. 3, March 2006, pp. 33–35