Why all our kids should be taught how to code

There is a growing consensus that the way children in schools are being taught information technology is in need of a radical overhaul. Here John Naughton explains the problem and offers a manifesto for revolutionary action

John Naughton
The Observer, 31 March 2012

What's missing from teaching computing in schools is a big vision. Photograph: Alamy

A vigorous debate has begun – within government and elsewhere – about what should be done about information and communication technology (ICT) in the school curriculum. Various bodies – the Royal Society, the Association for Learning Technology, Computing at School (a grassroots organisation of concerned teachers) and the British Computer Society, to name just four – have published reports and discussion documents aimed at ministers and the Department for Education. Michael Gove, the education secretary, made an enigmatic speech at the recent BETT technology conference indicating that a rethink is under way in the bowels of Whitehall. Meanwhile, in another part of the forest, there are some astonishing developments happening – such as the fact that more than a million people have already placed orders for Raspberry Pi, the cheap, credit-card-sized computer developed by Cambridge geeks, which began shipping last week.

So something's happening: there's a sense of tectonic plates shifting. But as with most big policy debates, there's a lot of axe-grinding, lobbying and special pleading going on. Universities want to reverse the decline in applicants for computer science courses. Gaming companies want more programmers. The government wants more high-tech start-ups. Manufacturers want trainees who can design embedded systems. And head teachers want bigger budgets for even more computer labs. And so on.

What's missing from all this is a big vision. So here's my shot at one:

Starting in primary school, children from all backgrounds and every part of the UK should have the opportunity to: learn some of the key ideas of computer science; understand computational thinking; learn to program; and have the opportunity to progress to the next level of excellence in these activities.

We'll get to why this is important and necessary in a moment, but first we need to face up to a painful fact. It is that almost everything we have done over the last two decades in the area of ICT education in British schools has been misguided and largely futile. Instead of *educating* children about the most revolutionary technology of their young lifetimes, we have focused on *training* them to use obsolescent software products. And we did this because we fell into what the philosopher Gilbert Ryle would have called a "category mistake" – an error in which things of one kind are presented as if they belonged to another. We made the mistake of thinking that learning about computing is like learning to drive a car, and since a knowledge of internal combustion technology is not essential for becoming a proficient driver, it followed that an understanding of how computers work was not important for our children. The crowning apotheosis of this category mistake is a much-vaunted "qualification" called the European Computer Driving Licence.

What we forgot was that cars don't run the world, monitor our communications, power our mobile phones, manage our bank accounts, keep our diaries, mediate our social relationships, snoop on our social activities and even – in some countries – count our votes. But networked computers do all of these things, and a lot more besides.

So we need to admit that "ICT in schools" has become a toxic brand. We have to replace it with a subject that is relevant, intellectually sustaining and life-enhancing for students. For want of a better name, let us call it computer science. This is an umbrella term that covers two distinct areas. First a set of key concepts that are essential if schoolchildren are to understand the networked world in which they are growing up. And second, computer science involves a new way of thinking about problem-solving: it's called computational thinking, and it's about understanding the difference between human and artificial intelligence, as well as about thinking recursively, being alert to the need for prevention, detection and protection against risks, using abstraction and decomposition when tackling large tasks, and deploying heuristic reasoning, iteration and search to discover solutions to complex problems.

There will be lots of interesting discussions about the key concepts that students will need to understand, but here's one possible list for starters. Kids need to know about: algorithms (the mathematical recipes that make up programs); cryptography (how confidential information is protected on the net); machine intelligence (how services such as YouTube, NetFlix, Google and Amazon predict your preferences); computational biology (how the genetic code works); search (how we find needles in a billion haystacks); recursion (a method where the solution to a problem depends on solutions to smaller instances of the same problem); and heuristics (experience-based techniques for problem-solving, learning, and discovery).

If these concepts seem arcane to most readers, it's because we live in a culture that has systematically blindsided them to such ideas for generations. In that sense, CP Snow's "Two Cultures" are alive and well and living in the UK. And if you think they are too sophisticated to be taught to small children, then that's because you've never seen gifted and imaginative teachers go to work on them. In fact many UK readers in their 30s will have been exposed to recursion, for example, because once upon a time many UK schools taught Logo programming, enabling children to learn how a mechanised turtle could be instructed to carry out complex manoeuvres. But in the end most of those

schools gave up teaching Logo and moved backwards to training kids to use Microsoft Word.

Incidentally, the Logo story provides a good illustration of why teaching kids to write computer programs has to be an integral part of any new computer science curriculum. The reason is that there's no better way of helping someone to understand ideas such as recursion or algorithms than by getting them to write the code that will implement those concepts. That's why the fashionable mantra that emerged recently – that "code is the new Latin" – is so perniciously clueless. It implies that programming is an engaging but fundamentally useless and optional skill. Latin is an intriguing, but dead, language; computer code is the lingo of networked life – and also, it turns out, of genetic replication.

Another misconception that is currently rife in the debate about a new curriculum is that the primary rationale for it is economic: we need more kids to understand this stuff because our "creative" industries need an inflow of recruits who can write code, which in turn implies our universities need a constant inflow of kids who are turned on by computers. That's true, of course, but it's not the main reason why we need to make radical changes in our educational system.

The biggest justification for change is not economic but moral. It is that if we don't act now we will be short-changing our children. They live in a world that is shaped by physics, chemistry, biology and history, and so we – rightly – want them to understand these things. But their world will be also shaped and configured by networked computing and if they don't have a deeper understanding of this stuff then they will effectively be intellectually crippled. They will grow up as passive consumers of closed devices and services, leading lives that are increasingly circumscribed by technologies created by elites working for huge corporations such as Google, Facebook and the like. We will, in effect, be breeding generations of hamsters for the glittering wheels of cages built by Mark Zuckerberg and his kind.

Is that what we want? Of course not. So let's get on with it.