DOI:10.1145/2742693

Sally Fincher

Education

What Are We Doing When We Teach Computing in Schools?

Research on the cognitive, educational, and policy dimensions of teaching computing is critical to achieving "computer literacy."

LET ME TELL you a story.

When my son was in fifth grade, we went to a parent-teachers' evening. If you have ever been to one of these, you know the drill. You cross the school-gate line and trespass into classrooms that are normally forbidden territory. You gamely insert yourself into chairs designed for considerably smaller people. You walk around admiring nature tables, number lines, and colorful compilations of work accumulated over the school year. And as we wandered and admired we saw a colorful poster that said "SCIENCE: Gravity makes heavier things fall faster." This is what happened next:

"That's wrong."

"It's in the National Curriculum."

"I don't care: that's wrong."

"But we did the experiment!"

"It's wrong. You have to take it down. Now."

At which point, conflict-averse, I walked away.

Lorem ipsum dolor sit amet consect adipiscing nunc enim mauris sed massa

Every Teacher Must Teach

This is both a true story and a cautionary tale. It illustrates a considerable concern, which I believe we all should share, for the push to get computing taught in schools. Whenever subject matter is taught by non-specialists—and, of course, most school teachers are not computing specialists—disciplinary understanding will be compromised. School teachers use materials created by other people and do the best they can to incorporate them into their classes. My son's teacher did not deliberately set out to teach something wrong. She took materials available to her and tried to make them work. The U.K. National Curriculum said fifth-grade children should be taught about forces and motion. It said, "objects are pulled downwards because of the gravitational attraction between them and the Earth."[2] Perhaps in "the experiment" they sent two children to the top of a slide, one with a heavy cricket ball and one with a light supermarket plastic bag. On the ground there were children with stopwatches. Someone shouted "Go!" The children dropped their objects. Other

CREDIT TK

children pressed their stopwatches. The ball fell to the ground. Some children shouted "Stop!" The wind caught the plastic bag and blew it to the other side of the playing field. Other children shouted "Stop!" Maybe they did it again, with different children. And then they went indoors, charted their results, and wrote the poster. Perhaps the children enjoyed the experience. They might have gone home and said, "we did science today." The teacher had a good lesson. Except the poster remained on the wall, unchallenged, for the greater part of the school year and these 20 years later may still remain, unchallenged, in some minds: we learn wrong things just as hard as we learn correct things.

Every Child Must Learn

There is a current rhetoric that all children should be taught computer literacy. I think we would do well to pause and ask what it means to be "literate." In traditional terms, we agree it is important that everyone should be able to read and write—and that if they cannot, they are disempowered, unable to participate in society. It is not, of course, easy. Children do not learn at the same rate, or in the same way, but being able to read is so important a skill that everyone must have it—even those with no talent for it, or appetite to learn. So how do we teach *everyone* to read? Historically, we have tried several things.

Simplifying the Spelling: Initial Teaching Alphabet

The "initial teaching alphabet" (ITA) was developed by James Pitman in the 1960s and introduced into U.K. schools. At root, it recognized that English spelling is extremely irregular (consider: *though*, *rough*, *bough*, *through* and *thorough*) and this syntactical barrier made it difficult for children to learn to read. The ITA consisted of 44 "characters" that represented the sound of words, regularized spelling, and so simplified complexity: thus "Initial Teaching Alphabet" became inifhul teeching alfubet. Books and materials were produced to support this, so children could follow the adventures of corky's enjjn or share the problems of the pengwin hω cωdn't paddl

Did it work? In 1963–1964, a study compared 873 children who learned to read and write in the traditional way with 873 children who were taught using the ITA.[1] The good news was that children learned to read faster with ITA: the bad news was that children could not transfer that learning to traditional books. A teacher recalls "I was in my second year's teaching in Luton in 1968. ITA seemed to be a brilliant way of pushing the children on and they learned to read much earlier than usual. But—and the 'but' is enormous—some could not make the transition. I don't think they'll ever unlearn ITA spelling."[3]

Simplifying the Vocabulary: "Look and Say"

Another approach took the view that it was not sensible to change the texts that children were presented with. In "look and say" children were taught to memorize high-frequency words as whole shapes: it was thought that 30 repetitions of a word would be enough for a child to learn it. So, books were produced that introduced careful sequences of words with considerable repetition. These stories featured docile children who played all day and had a curiously leaden way of talking to each other:

John, see the aeroplanes. One, two, three aeroplanes. I can see three aeroplanes. John said 'See the aeroplane go up. See the aeroplane fly. The aeroplane can fly fast. Fly fast, big aeroplane'[4]

By the end of which the reader had seven exposures to *aeroplane*.

Did it work? Morag Stuart reports an experiment she conducted with colleagues in 2000 to see how easy it was for five-year-old beginning readers to store new, whole words in memory from repeated reading of whole word texts.[5] They worked with 16 new words.

Are we teaching an ITC—an *Initial Teaching Computing*—that, in avoiding "obvious" difficulties, leads to problems later?

"After the children had seen and read each word 36 times, no child was able to read all 16 of them, and the average number of words read correctly was five. We were quite shocked by this."

"When we tested children's ability to read words they'd experienced more than 20 times in their school reading, on average they could read only one word correctly."

Computing in Schools

So, those people who call for "computer literacy" must believe it is as important for our children to learn computation as it is for them to learn how to read. To me, that seems unlikely ever to be the case. But, if that's their goal, I know it cannot be achieved by enthusiastic effort (however expert) happy to engage mainly the "gifted and talented" in forums of their own choosing, in voluntary "after-school" clubs, or via subscription services like Bitsbox, which sends out monthly coding projects for children with the educationally naive pitch "Learning to code takes time and practice. Sometimes it's just plain hard. But that's okay—we know kids are willing to work at learning hard things as long as they don't get bored along the way."

Restricting the Syntax

In historical symmetry we can see a similar response to coding literacy as in traditional literacy. Block-based languages (like Blockly), microworlds (like Alice), and "Initial Teaching Environments" such as Scratch, reduce the syntactic complexity of coding to allow a more direct access to fluency. But they suffer the same problems as any "cut down" approach and the question of transition to "grown up" languages, remains as potent with them as it was for the ITA and reading. Are we teaching an ITC—an *Initial Teaching Computing*—that, in avoiding "obvious" difficulties, leads to problems later?

Restricting the API

Other approaches believe it is more appropriate to use real syntax, but constrain the environment to a particular (attractive) problem domain so learners become fluent in a constrained space. Event-driven environments (such as Greenfoot) or scaffolded systems (like Processing.js) aim for the learner to develop an accurate mental

CREDIT TK

model of what their code is doing, and ultimately transfer that to other environments. Although whether they actually do so remains unclear: we may be restricting things in the wrong way.

Still others hold that coding—howsoever approached—is insufficient for literacy and advocate a wider approach, taking in "computational thinking," for instance as embedded in the framework of the "CS Principles": Enduring Understandings, Learning Objectives, and Essential Knowledge.

What is resolutely held common with traditionally formulated literacy is that these approaches are unleashed on classrooms, often whole school districts, even into the curriculum of entire countries—with scant research or evaluation. And without carrying the teachers. If we are to teach computing in schools we should go properly equipped. Alongside the admirable energy being poured into creating curricular and associated classroom materials, we need an accompanying set of considered and detailed programs of research, to parallel those done for previous literacies.

In complement to efforts in mathematical and natural language education we need to undertake *cognitive research* to discover how children acquire computational concepts asking questions (for example) as to whether there is a "best order" for the presentation of concepts, or whether pedagogically focused "initial programming environments" are a more productive way to learn than "real language" teaching. And, if so, under what conditions? This is not virgin territory, but the majority of previous work has been on learning in cognitively mature undergraduates, and that is unlikely to transfer directly.

In parallel, we need a program of *educational research* to support teachers, to ensure ideas work in real classrooms and with real teachers—and so we do not repeat cycles of error. At the moment, teachers are faced with a plethora of plausible approaches and no way to choose between them but the conviction (and charisma) of their inventors. A recent *Computing at School* magazine (Autumn 2014) is not short of ideas: *A four step scaffolding exemplar using Scratch … A simple project utilizing the python turtle library … Functional programming: an example in VB.* Each of these is a response to the need for teachers to have something to teach, to be able to fill their lessons with engaging and useful material. But, at the same time, the evidence these are based on is solely "Do it like this! It works for me!"

Finally, we need *policy research* so we may effectively coordinate and disseminate practices at scale. It is not only individuals who can learn from research—districts, countries, and governments can, too.

Moral

It is tempting to think these are not true problems, that my opening story is an artifact of history, and that the attention of intelligent people, the support of professional organizations, and the commitment of money being applied to making our children computationally competent is sufficient for success. But I was recently talking with an industrialist who has a considerable commitment to outreach work. He was describing going into sessions for training school teachers (for children aged 5–11) and said "When I talk about computational thinking, they look horrified."

And what happens when companies stop donating their staff's volunteer effort? When the spotlight of governmental attention passes on? Without the scaffold of *evidence* we risk condemning both these teachers, their pupils, and our large-scale efforts (with Scratch or Alice) to failure (like ITA or look-and-say), or to be successful only in certain localities under certain conditions.

When you frame a subject as *literacy*, the educational problems that entails are very different to the problems of subject experts enthusing an engaged minority. We should learn from educational history, and—this time—do the research. C

References

1. Bell, M. The significance of the ITA experiment. *Journal of the Simplified Spelling Society*, 29 (2001).
2. Department for Education and Employment. *The National Curriculum: Handbook for Primary Teachers in England.* HMSO, 1999.
3. Lane, M. Educashunal lunacie or wizdom? 2001; http://news.bbc.co.uk/1/hi/uk/1523708.stm.
4. O'Donnell, M. In Munro, R., and Warwick, M., Eds. *Janet and John …* James Nisbet and Company, Welwyn, 1959.
5. Stuart, M. Learning to read the words on the page. In Lewis, M. and Ellis, S., Eds. *Phonics: Practice, Research and policy.* Paul Chapman, London, 2006.

Sally Fincher (S.A.Fincher@kent.ac.uk) is a professor of computing education and director of graduate studies at the University of Kent.