INSIGHT: Thinking Issues

Tony Clear

Everybody must cut code! Educational imperative, fad or fantasy?

Political momentum now seems to be growing towards ensuring the availability of computing education curricula in several countries, not only at the high school level but at even earlier stages of education. But what are the chances of these initiatives being successful? This column discusses recent developments in the context of earlier computing curricula implementations in a selected group of mostly western countries at the pre-university level. The challenges to be surmounted, and the seeming lack of concern for the potentially negative side effects of hasty curriculum implementations are presented, then complemented with a critical projection of the likely eventual outcomes.

In one example of such a development, recently the New Zealand Government became a signatory of the D5 Charter. Under the charter the D5 purports to represent "*a group of the most digitally advanced governments in the world. The D5 will provide a focused forum to share best practice, identify how to improve the Participants' digital services, collaborate on common projects and to support and champion our growing digital economies*"[6]. So immediately we see a set of utilitarian drivers underpinning the initiative, make the signatories richer and ensure that they come out as winners in a globally competitive digital race! Participating governments to date are Korea, UK, Estonia, New Zealand and Israel.

One article of the charter is the following commitment:

3.7. Teach children to code - commitment to offer children the opportunity to learn to code and build the next generation of skills

But as noted in [6], "What, if anything, the D5 may cost each member country in the future is open to question, as "*Participants will decide on a case by case basis how any joint initiatives will be funded and resourced*".

Programming early and programming for everyone has been a long argued perspective, from Seymour Papert in the 1980's with the Logo programming language [14], to more recent developments targeting younger and more diverse audiences in block based visual programming languages such as Alice and Scratch [10,11,15]. Complementing these movements as a counter to the narrow focus on 'programming' (or even worse 'coding'!) has been the push for more widespread adoption of the broader notion of "computational thinking" [16]. Educationally these developments fit within numerous curriculum initiatives country by country, with various cycles of nomenclature, IT fluency [3], digital literacy, computer literacy, digital technologies [1,7], coding skills [5,6], computational

thinking [16] etc. Individual country trajectories will have varied, but the progression of CS curricula at High School levels has mostly been mixed and confused, demonstrating inconsistency in differentiating between three different approaches: 1) teaching CS as a discipline in its own right; 2) disseminating computer literacy, and 3) engaging in the use of computers for teaching other subjects. In one country study reported in 2004 [9], the Finnish upper high school CS curriculum was shown to have moved significantly over a period from 1972 with 'Automatic Data Processing' (ADP) taught as a sub-field of mathematics, towards 'Information Technology' in 1985. Two courses were described: a) basic use of computers, b) programming). By 1994 CS studies had become subsumed under broad themes, and by 2003 in an increasingly devolved curriculum, the two themes covering CS were "technology and society" and communication and media competence", without specific reference to computers or CS. The authors of the study concluded "*enthusiastic teachers should be cherished as precious assets…[who] decide whether courses in CS are arranged or not*" and that "*the possibility to learn CS in Finnish high schools is changing to a game of chance*" [9]. This dismal picture would not be widely different from the experience of many other countries.

More recently the concerns voiced by key influencers in the UK technology community about the abysmal curriculum offerings their children had available to them in their schools, had a great political impact, as is evident in the quote below from the UK education minister Michael Gove:

> *"ICT used to focus purely on computer literacy – teaching pupils, over and over again, how to word-process, how to work a spreadsheet, how to use programs already creaking into obsolescence; about as much use as teaching children to send a telex or travel in a zeppelin.*
> *Our new curriculum teaches children computer science, information technology and digital literacy: teaching them how to code, and how to create their own programs; not just how to work a computer, but how a computer works and how to make it work for you." [5]*

I have sympathy for this concern myself seeing my granddaughter [an attendee of many CS Ed conferences and Scratch literate] being subjected to the same experience of end-user computing as ICT at high school.

Yet if we take on board the lessons learned from the New Zealand senior high school CS curriculum roll-out, a major move towards a more credible CS curriculum is challenging for systems and teachers. As noted in [1] a survey of teachers showed that the majority were aged 50 plus, had no experience in computer science as a subject and were not confident in teaching the new materials. To put this in perspective, would we seriously expect a computer scientist to teach senior high school Latin after a two day crash course and a pointer to a repository of someone else's hastily prepared curriculum materials? Yet these lessons reflect experiences with introducing a CS curriculum at high school level, not as now proposed in some countries [notably UK and Australia], from the elementary school and upwards [5,7]. As observed in [7] stakeholder consultation in Australia has "*identified significant concerns in relation to teacher development (particularly at F-7), appropriate pedagogy, and skills needed for integration of DT learning objectives with the teaching of other learning areas*". The authors further note

the limited research in the area to guide teachers in appropriate pedagogical strategies. The work by Ray Lister on neo-piagetian stages of cognitive development at tertiary level [12], should give some additional cause for concern about the typical student's ability to handle different forms of abstraction at the earlier schooling levels.

Even worse, with the way in which Anglo-phone governments (who mostly tend to be of a neo-liberal stripe) manage schools [and in some cases tertiary institutions] punitively through withdrawal of funding based on student pass rate thresholds [4,2], having demanding courses in which students fail is not a good thing. In [3] I have previously commented on the role of dumbed down end user focused ICT courses as an easy way of getting non-academic students through examination hurdles and making the school statistics look good. As also noted in the New Zealand context [1]

> *Two respondents mentioned that management and colleagues do not understand the new courses: "Educating other staff (still) that this is not a typing class," "Management, both Senior and Departmental do not understand the importance of the topic in terms of content and job opportunities. The digital technology [achievement standards] are viewed as being 'too hard' and there is a push to return to [unit standard] work where students gain credits for doing rather than thinking."*

Then further taking equity considerations into account, what is the likely impact of these developments in pushing CS further back in the curriculum. Will it merely reinforce the negative gender stereotypes about CS as a discipline earlier? If we acknowledge the findings of Jane Margolis in her dispiriting book about race, class, prejudice and quality of CS teaching in poorer and minority schools, then will this extension of the CS curriculum bear fruit in those schools, or be just another under resourced, poorly taught and marginalised subject?

Politically driven CS curriculum initiatives aimed at developing the competitiveness necessary for the much vaunted knowledge economy are probably merely a passing fad. Yet few could take issue with the need for a serious revamp of what passes for a CS/IT curriculum below the University level in most countries. Though when we see educators in the tougher schools struggling to teach basic numeracy and math at elementary and high school levels [13], the prospect of the universal success of a 'computational thinking' curriculum, which we could view as 'numeracy on steroids' seems rather far-fetched to me.

The utopian view will see the developing of programming skills from an early age as a part of a modern liberal education, in which students have the opportunity to express a new form of creativity, through that most plastic of all media - software! If we do see a set of curriculum initiatives that can succeed in re-sparking the passion, beauty, joy and awe: and making computing fun again [8], then that would be wonderful. I sincerely hope this will be the outcome, but with this ambitious broadening out of the CS curriculum and its attempted wholesale delivery across primary, secondary and tertiary sectors of education, I fear that things will persist. Much as they have done for the

last four decades of computing education in schools, I suspect we will continue to see isolated islands of excellence drifting in a sea of dross!

1. Bell, T., Andreae, P. and Robins, A. "A Case Study of the Introduction of Computer Science in NZ Schools." *Transactions in Computing Education*, 14 (2014): 1-31.

2. Clear, A. and Clear, T. "Introductory Programming and Educational Performance Indicators - a Mismatch". in *Proceedings of ITx New Zealand's Conference of IT*, Hamilton, New Zealand: CITRENZ, 2014: 123-128.

3. Clear, T. and Bidois, G. "Fluency in Information Technology – FITNZ: An ICT Curriculum Meta-Framework for New Zealand High Schools." *Bulletin of Applied Computing and IT,* 3 (2005): *http://www.citrenz.ac.nz/bacit/0303/2005Clear_FITNZ.htm*.

4. Dee, T.S. and Jacob, B. "The impact of no Child Left Behind on student achievement." *Journal of Policy Analysis and Management*, 30 (2011): 418-446.

5. Dredge, S. "Coding at school: a parent's guide to England's new computing curriculum." *theguardian*, September 4, 2014. http://www.theguardian.com/technology/2014/sep/04/coding-school-computing-children-programming. Accessed 2015 August 9.

6. Eskow, S. "NZ Govt commits to kids coding and open source." *IITP Techblog*, 13 July, 2015. http://www.techblog.nz/958-NZGovtcommitstokidscodingandopensource. Accessed 2015 August 9.

7. Falkner, K., Vivian, R. and Falkner, N. "The Australian Digital Technologies Curriculum: Challenge and Opportunity". *In Proceedings of the. Sixteenth Australasian Computing Education Conference (ACE2014)*, ACS, 2014, Auckland, New Zealand: 3-12.

8. Garcia, D.D., et al., "Rediscovering the passion, beauty, joy, and awe: making computing fun again, continued." *in Proceedings of the 40th ACM technical symposium on Computer science education*, ACM, 2009, Chattanooga, TN, USA: 65-66.

9. Kavander, T., and Salakoski, T. (2004). "Where have all the flowers gone? - Computer Science education in general upper secondary schools." *In Proceedings of the Fourth Finnish/Baltic Sea Conference on Computer Science Education*, Helsinki University, 2004, Koli, Finland: 112-115.

10. Kelleher, C. and Pausch, R. "Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers." *ACM Computing Surveys*, 37 (2005): 83-137. [doi>10.1145/1089733.1089734]

11. Kelleher, C. and Pausch, R. "Using storytelling to motivate programming." *Communications of the ACM*, 50 (2007): 58-64. [doi>10.1145/1272516.1272540]

12. Lister, R.," Concrete and Other Neo-Piagetian Forms of Reasoning in the Novice Programmer." *in Proceedings of the Thirteenth Australasian Computing Education Conference (ACE 2011)*, ACS, 2011, Perth, Australia: 9-18.

13. Margolis, J. *Stuck in the Shallow End: Education, Race, and Computing*. (Cambridge, Massachusetts: The MIT Press, 2008).

14. Papert, S. *Mindstorms: children, computers, and powerful ideas*. (New York: Basic Books, Inc., 1980)

15. Resnick, M., et al., "Scratch: programming for all." *Communications of the ACM*, *52* (2009): 60-67.

16. Wing, J.M. "Computational thinking", *Communications of the ACM*, 49 (2006): 33-35. [doi>10.1145/1118178.1118215]

D5 Charter

The D5 Charter

The D5 Charter
70x45mm (72 x 72 DPI)

Everybody must cut code!