Computer Science in New Zealand High Schools

Tim Bell
Department of Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
tim.bell@canterbury.ac.nz

Peter Andreae
School of Engineering and Computer Science
Victoria University of Wellington
New Zealand
Peter.Andreae@ecs.vuw.ac.nz

Lynn Lambert
Physics, Computer Science and Engineering Department
Christopher Newport University
Newport News, Virginia, USA
llambert@cnu.edu

Abstract

The New Zealand Ministry of Education has recently released a new "Digital Technologies" proposal for delivering computing topics in the final three years of High Schools. The proposal aims to address a number of issues by offering topics that will be academically challenging for students, and provide them with a broader view of the kinds of advanced topics they might study beyond High School. The proposed structure includes having Digital Technologies as a separate area in the technology curriculum, and includes a strand called "Computer Science and Programming" that has sufficient coverage to communicate to students what the subject area is really about.

This paper reviews the circumstances that led to this proposal, describes the international context (especially in the US) for High School computing curricula, and examines the published proposal in some detail. It also considers the issues that are likely to come up in the implementation of the proposal, and how they might be addressed.

Keywords: Computer Science curriculum.

1 Introduction

A secondary school Computer Science curriculum can have a significant influence on student career paths, both for laying the groundwork for further study, but more importantly, for exposing students to the topic. The latter is particularly valuable because the discipline of Computer Science is not well understood by High School students, who often make career choices based on an inaccurate perception and bad experiences unrelated to the topic itself (Margolis and Fisher 2002).

In recent times, New Zealand schools have rarely taught Computer Science – at best there have been courses on programming at some schools, but often computing education has been focused on general purpose applications and skills. Even worse, sometimes courses that teach "computing as a tool" have given students the impression that CS must be an extension of these topics. Of course, it is important for students to be able to use computers effectively, but often this has been a distraction from getting students involved in "computing as a discipline".

For a period (1974-1985) computing was taught as a discipline in NZ schools through "Applied Maths". However, there have been several changes since then, and recently assessment for computing courses that go beyond just using applications have typically been via "unit standards", which are pass/fail skills-based standards that are not attractive to top students who would like to get high grades to reflect their academic achievements.

In addition to these issues, NZ had adopted a technology curriculum that provided generic assessment tools for teaching areas ranging from food technology to digital technology, which meant that computing wasn't a subject area of its own, and the same kind of assessment criteria would be used for a large range of technologies (such as meal planning and software development). Because computing was combined with other technologies, it was less accessible as a discipline in its own right, and less attractive to students who were specifically interested in the discipline of computing.

Some progress had been made towards a broader and deeper approach to the curriculum, such as the "Fluency in IT" project (Clear and Bidois 2005), but mapping such proposals onto the new national technology curriculum was proving to be problematic.

In 2008, two reports were released that very clearly pointed out the weaknesses of the current offerings in schools, and called for action from the government agencies that set the standards used to determine the courses delivered by schools (Grimsey and Phillipps 2008; Carrell, Gough-Jones, and Fahy 2008).

This resulted in the Ministry of Education calling together a "Digital Technologies Experts Panel" (DTEP) representing industry, tertiary and High Schools, to develop a plan to address the issues raised in these two reports. The panel first met in November 2008, and by mid 2009 it had produced a body of knowledge and recommendations for a way forward.

The DTEP recommendations were released in May 2009, along with an agreement negotiated with the Ministry of Education for moving forward[1]. The agreement included:

- A specific area called "Digital Technologies" within the technology learning area.
- An aggressive timeline for implementation, with guidelines to be made available to schools in 2010, so that courses in this area could be run from 2011.
- A body of knowledge in five strands of Digital Technologies (see below).

The DTEP report also mentions the following, which seem likely to be achieved by the changes planned:

- Better alignment of school material to tertiary and industry expectations of areas such as Computer Science.
- Assessment standards that are academically challenging, and help students to meet entrance standards for tertiary study.
- Use of ICT-related terminology in schools that reflects the usage in industry and tertiary organisations.
- Urgent professional development for teachers.

Considerable work had already been done in previous years on "Digital Technology Guidelines" (DTG)[2], and the DTEP recommendations included adapting these guidelines to match the recommended body of knowledge, rather than starting from scratch. This meant that the new material could be delivered taking advantage of the existing momentum achieved by the DTG.

The new proposal (called "Technological Context Knowledge and Skills") was posted publicly for comment in August 2009[3]. At the time of writing, the plan is being updated to take account of feedback, and the version reported on here is the August 2009 version. The public discussions of the proposal indicate that it is being well received by industry, tertiary institutions, and schools, which bodes well for its implementation.

If the new proposal is successful, it should inspire and prepare students to contribute to the growth of New Zealand's economy through innovative work based on good foundations in subject knowledge, and give students an understanding of the different areas in ICT so that students can make sensible choices. It should also address the issue of computing not being regarded as having a high academic status

[1] The full version is available from http://www.techlink.org.nz/curriculum-support/tks/

[2] http://dtg.tki.org.nz/

[3] http://www.techlink.org.nz/curriculum-support/tks/

in schools, and so should attract more high academic achievers into the field.

Section 2 reviews some of the international work on the development of CS curricula that constitutes a context for the new NZ guidelines; section 3 describes the proposed NZ learning area for Digital Technologies in general, and section 4 focuses on the proposed Computer Science topics within Digital Technologies. The issues surrounding implementation of the new guidelines are discussed in section 5, and we draw conclusions in section 6.

2 CS in school curricula

Computing in school curricula is often diluted because it has to cover three quite different directions: (1) using computers as a tool for teaching (e.g. e-learning), (2) using computers as a tool for general purpose applications (sometimes called ICT), and (3) computing as a discipline in its own right (including programming and CS). Sometimes administrators and leaders confuse these roles, and this can make it difficult for Computer Science to be visible as a discipline in its own right.

Although computing as a tool is commonly taught around the world, relatively few countries have a significant CS curriculum, and even fewer make such a curriculum mandatory for schools (Ragonis 2007).

In the United States, there is no federal organization that guides the curriculum of Computer Science. The main federal law regarding education in the United States is the “No Child Left Behind” act. This law states that all teachers must be highly qualified except teachers in non-core areas (Wilson and Harsha 2009). The non-core areas are: Physical Education, Computer Science, and vocational education. There are several implications of this: first, Computer Science is not considered a core area, so schools tend to emphasize it less than core areas (the states receive funding based on how well they are doing in core areas). Second, it is up to the 50 states to implement Computer Science programmes. Thus, there is no single technology or Computer Science program in the United States, but many.

Although there are no federal guidelines and differing state requirements, several different organizations have developed technology or computer guidelines for the United States. The Computer Science Teachers Association (CSTA) standards differentiate technology and using computers in the support of education in general from the field of Computer Science in its model K-12 Computer Science curriculum (Tucker *et al.* 2006). The International Society for Technology in Education (ISTE) has developed technology standards called NETS, National Education Technology Standards[4]. The National Association for Educational Progress (NAEP) is developing technology literacy guidelines[5] that will become part of the nation’s report card in 2012, although these are not specifically computing technology. With no federally enforced standard, the *de facto* standard for High School curricula are Advanced Placement exams – of 23,000 high schools nationally, 17,000 offer some AP courses. AP Computer Science, which has the same curriculum and test across the country, is almost exclusively programming (currently in Java). A group of prominent Computer Science educators is attempting to redesign the AP curriculum to create a course that is less centred on programming (Cuny 2009).

Many of the difficulties implementing effective computing curricula are common to a number of countries, and the NZ experience has reflected the experience of others, including the rapid decline in tertiary CS enrolments after the year 2000 (Vegso 2008). The recent developments in NZ appear to have addressed many of the issues raised, and no doubt there will be lessons to learn and new material developed as the new proposal is implemented in schools.

3 New Digital Technology Guidelines

The proposed guidelines for Digital Technology in NZ schools are given in a “Technological Context Knowledge and Skills” (TCKS) document released by the NZ Ministry of

[4]http://www.iste.org/AM/Template.cfm?Section=NETS

[5] A draft is available at http://www.edweek.org/media/nagb_assessment_devel_comm_aug_7-09.pdf

Education (Dinning 2009). The guidelines have five "contexts" for Digital Technologies:

- Digital Information (digital tools and systems for managing information),
- Digital Infrastructure (hardware and networks, including installing software),
- Digital media (video, audio, layout/design, web, graphics, animation, games, web),
- Electronics (electronic and embedded systems), and
- Programming and Computer Science (concepts from CS and Software Engineering, designing and implementing programs).

These five areas address a range of interests and career paths that students might take, and are aimed at the final three years of high school.

Each area contains a set of objectives specifying the different aspects of knowledge and skill in the area. Each objective is broken down into three levels (6, 7 and 8) of the NZ curriculum, which would correspond to the last three years of High School for most students (years 11 to 13) with a list of "indicators" that give more details of the objective for each level.

In addition to these subject-specific guidelines there are supplementary "generic" achievement standards on "Technological practice", "Nature of technology" and "Technological knowledge", which would be available to assess topics such as the history of computing, the effect of digital technology on society, or project management.

If the proposed guidelines are accepted, then the ministry will develop a set of achievement standards addressing the knowledge and skills described by the indicators in the document, and guidelines for teachers. The way the curriculum works in NZ, schools will then be free to make up courses that are built around their own selection of achievement standards, based on local interests and strengths. A likely outcome in many schools is that a year 11 student may study an introduction to several of the five areas in a single course, whereas at year 13 schools might offer a whole course based on just one or two of them.

Some of the "contexts" (especially Digital Information and Digital Media) will be more concerned with the computer as a tool, but the others are more concerned with the computer itself. In the following section we will look in detail at the "Programming and Computer Science" context.

4 CS in the new guidelines

What had previously been just "programming" in the existing Digital Technology Guidelines now appears as "Programming and Computer Science", reflecting a concern for giving students a broader basis for making an informed decision about possible paths in the tertiary sector. The first of the three objectives in the published proposal addresses this broadening:

> "Demonstrate an understanding of concepts across Computer Science and Software Engineering."

The indicators for the above objective introduce the fundamental concepts of algorithms and programming languages at level 6. At level 7, these are expanded to include four further important concepts:

- The ideas of complexity/tractability/ computability – that some problems are inherently difficult or impossible to solve on a computer.
- Coding (*e.g.*, compression, error correction, encryption), and how it has enabled new technologies.
- That programming languages are specified precisely.
- The need for Software Engineering methodologies, and an appreciation of the steps in the Software Development Life cycle.

The indicator at Level 8 is broader, and allows a selection of topics from across Computer Science and Software Engineering.

The wording of (and the examples in) the indicators make it clear that the goal is an *appreciation* of what Computer Science and Software Engineering are about, and not an in-depth understanding of the content of the topics. For example, the concepts of complexity/tractability/computability at level 7 are intended only to have the students understand that some problems are very difficult to solve, no matter how clever the algorithm, and that some problems are impossible. It would not be appropriate at this level for the students to have to determine the complexity class of a problem or construct a proof of intractability, but they might appreciate that binary search is significantly better than linear search, even if only for searching a telephone book. Students who gain an appreciation of these concepts will know that Computer Science is more than just programming and will be much better placed to start a tertiary qualification in Computer Science – or to decide that Computer Science is not what they want to study!

The other two objectives for the Programming and CS context address two complementary aspects of programming. One addresses the *design* of programs:

> " Be able to understand, select and design data types, data structures, algorithms, and program structures for a program to meet specified requirements, and evaluate user interfaces."

The other addresses the processes for actually *constructing* programs:

> " Be able to read, understand, write, and debug software programs using an appropriate programming language, tools, and software development process."

Although these two aspects would almost certainly be intertwined in teaching practice, distinguishing designing from constructing emphasises that programming involves understanding and design at a more abstract level, as well as knowing the technical details of a programming language and being able to read and write programs in that language. There is also a practical advantage in distinguishing them, at least for assessment, in that weaker students who cannot cope with the design aspects may still be able to pass achievement standards addressing the practical skills of reading, understanding, writing and debugging programs if they are given sufficient guidance and support on the design aspect.

The design objective also includes a component on evaluating user interfaces. Although actually *designing* a good user interface is a more advanced topic than is appropriate at school level, it is quite feasible for students to analyse existing interfaces from a design perspective. Because most students will already have used a wide variety of interactive programs in a range of contexts, including multiple programs for the same task (*e.g.,* mobile phones, browsers, mail clients, picture viewers, DVD players), there will be plenty of options for getting students to compare, evaluate, and suggest improvements to existing user interfaces. At level 6, such evaluation would be strictly informal; at level 7, it would be based on lists of useability heuristics, and level 8 would use a wider range of Human-Computer Interaction (HCI) principles. Having students look at this aspect of computing means that they can broaden their view to appreciate the broader skills and knowledge (such as psychology for HCI or linear algebra for graphics) that are valuable for constructing successful digital systems.

Both programming objectives have a sequence of indicators at the three levels that build up from having students develop very simple programs. At level 6, the indicators require only programs that use variables, expressions, selection, and loops, and the primitive data types available in the chosen language. This is sufficiently simple that popular introductory languages (such as Scratch or Alice) could be used to assess them if delivered appropriately. Level 7 extends the indicators to include methods (or procedures/ functions/subroutines) and compound data structures (*e.g.,* arrays, or lists); level 8 adds the use of data from files and procedures with parameters and return values. The higher levels are likely to require a conventional general purpose language to cover the concepts listed (e.g. Java, Python, or Visual Basic).

Level 8 also adds a greater understanding of data types, with an appreciation of the properties and limitations of different data types. This might include an understanding of different ways of representing numeric data (binary, hexadecimal, fixed point, floating point, *etc.*) but it could equally be addressed in the context of representing textual data or image data. For many students, understanding different ways of representing pixel colour values for images may provide better motivation than traditional scientific calculation.

The proposal makes no mention of topics such as classes, packages, inheritance, or exceptions, and does not require programs to have graphical user interfaces. Of course, with some languages or program development tools, students may be exposed to programs using such concepts, but they are not required.

No particular programming language is specified, and teachers could use any appropriate language they wish as long as it has sufficient constructs to cover the structures required. At level 6, a very wide range of languages would be possible, including domain or application specific languages, as long as they supported programming with variables, expressions, selection, and loops. At level 7, the selected language would need to support procedural abstraction. At level 8, a general purpose programming language is required, along with an appropriate software development environment.

The indicators also require some documentation in the students' programs, but this is at the level of choosing good names, suitable layout, and using some appropriate comments. For the small programs that the students would be able to construct, elaborate documentation is not only unnecessary, but is generally unmotivating. Level 7 also introduces testing.

At Level 8, the indicator for the program construction objective requires some level of discipline in the programming process, with some problem analysis for simple requirements, and the use of a simple software development process. In general, this would be a simplified version of an agile process, emphasising repeated cycles with increasing requirements and careful testing against the requirements at each stage. The goal is not to gain a mastery of software engineering practice, but to become aware of the need for discipline in the programming process, and to appreciate the role of aspects such as requirement specification, testing, and debugging, in addition to the actual writing of program code.

5 Implementation

The proposed changes have a number of implementation challenges and implications for teacher training and student career paths.

Because the proposal introduces some topics not previously taught in schools, most teachers will require professional development to be able to teach them. For many, this may mean learning to program, and even for some of those who are comfortable teaching programming, it will require gaining some familiarity with the overview of Computer Science topics. Fortunately the new standards will be introduced over a period of four years, which gives teachers some time to get up to speed. During this time, there will need to be extensive communication with teachers so that they can keep in touch with developments, be aware of strong support for the transition, and feel engaged in the process.

New material will be needed for teaching topics that haven't existed before in schools. The Computer Science academic community in NZ is offering extensive support to develop material and help with professional development. This material will be published online, so it can be used by the international community.

If the new guidelines bring about the hoped-for growth in the discipline, more teachers will need to be found to deliver the resulting classes. This in turn will require more training opportunities, particularly in the Colleges of Education where teachers receive their key qualifications. For new teachers, College of Education courses will need to expand to cover the new topics proposed. Some teachers who are already in service, but are not currently involved in computing, might elect to retrain in these areas, in which case distance-learning or flexible courses may be more accessible. Finally, students and graduates with a background in Computer Science might be recruited to become teachers and bring their expertise to the classroom, given appropriate training to qualify them for the role.

There will be implications for tertiary institutions because the students leaving school may now have more advanced knowledge and experience in the discipline. While some schools may not offer the full range of standards proposed, others could potentially have students graduating who are competent programmers in a language such as Java or Python, and who have a reasonable understanding of algorithm analysis, including simple searching and sorting algorithms. For some tertiary institutions, this may represent a similar standard to their first-year Computer Science courses, and they may need to consider alternative paths for such students. In either case, students who have done well in these areas can be identified and potentially offered scholarships or more challenging courses.

In addition, students will need guidance on what other subjects to take at school. For example, Computer Science departments would typically be looking for students with strong mathematics and communication skills, and structures may need to be put in place to ensure that students are aware of the importance of these complementary subjects for success in their chosen career path. This will include communicating the new pathways to career counsellors and advisors.

6 Conclusions

New Zealand is on the verge of delivering an exciting programme for *computing as a discipline* in High Schools. The current design reflects enough technical material that students can get some insight into the career paths available to them, and also provide academic challenges to make the courses attractive to top students.

Implementing such changes requires a lot of careful design and testing, particularly for topics that have never been taught before. Teachers will need considerable help to become comfortable with new topics, and given the fast time scale, support from the tertiary community will be essential.

It is important that the change is not seen as a bigger list of things to learn in less time. The new topics are generally at the level of exposing students to them so they have an *appreciation* of their significance. More important than the details of what is taught is that students know what career paths are available, and are able to get a sufficiently accurate taste of the discipline to find out if it suits them or not, rather than making up their mind based on incorrect information and mislabelled topics.

Once the context and skills outlined above have been finalised, they will guide the creation of guidelines for teachers, and standards for assessment. This process will need to be complete enough by early 2010 that schools can plan and publish their Year 11 (level 6) programmes for students making decisions about their 2011 courses. Despite this rapid pace, the pipeline is three years long, and the first students to leave school having completed the new programmes will not be entering the tertiary institutions (or the workforce) until 2014, so it will be some time before the effects of the changes are fully realised.

7 Acknowledgement

We are grateful to an anonymous referee for detailed feedback and suggestions.

8 References

Carrell, T., Gough-Jones, V. and Fahy, K. (2008): *The future of Computer Science and Digital Technologies in New Zealand secondary schools: Issues of 21st teaching and learning, senior courses and suitable assessments.* http://dtg.tki.org.nz/content/download/670/3222/file/Digital%20Technologies%20discussion%20paper.pdf. Accessed 16 Sep 2009.

Clear, T. and Bidois, G. (2005): Fluency in Information Technology – FITNZ: An ICT Curriculum Meta-Framework for New Zealand High Schools. *Bulletin of Applied Computing and Information Technology* Vol. 3, Issue 3. ISSN 1176-4120. http://www.naccq.ac.nz/bacit/0303/2005Clear_FITNZ.htm. Accessed 16 Sep 2009.

Cuny, J. (2009): *A clean-slate approach to High School CS.* http://www.cra.org/Activities/summit/Cuny_A_Clean_Slate_Approach_to_High_School_CS.pdf. Accessed 16 Sep 2009.

Dinning, N. (2009): *Technological Context Knowledge and skills: Exploring specific knowledge and skills to support programmes in technology*. Materials for consultation to support Ministry decision making. http://www.techlink.org.nz/curriculum-support/tks/resources/Technological-Context-Knowledge-and-Skills-07-2009.pdf. Accessed 15 Sep 2009.

Grimsey, G., and Phillipps, M. (2008): *Evaluation of Technology Achievement Standards for use in New Zealand Secondary School Computing Education: A critical report*. NZ Computer Society. Available from http://www.nzcs.org.nz/news/uploads/PDFs/200805NCEAReport.pdf. Accessed 16 Sep 2009.

Margolis, J. and Fisher, A. (2002): *Unlocking the clubhouse: Women in computing* , The MIT Press, Boston, MA.

Ragonis, N. (2007): Computing Pre-University: Secondary Level Computing Curricula. In *Encyclopedia of Computer*

Science, 4th Edition, Ralston, A., Reilly E. D., and Hemmendinger, D. (Eds.)

Tucker, A. (editor), Deek, F., Jones, J., McCowan, D., Stephenson, C., and Verno, A. (2006): *A Model Curriculum for K-12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee.* Association for Computing Machinery (ACM), New York, New York, (Second Ed.)

Vegso, J. (2008): Enrollments and Degree Production at US CS Departments Drop Further in 2006/2007, CRA bulletin, 1 March 2008. http://www.cra.org/wp/index.php?p=139. Accessed 16 Sep 2009.

Wilson, C. and Harsha, P. (2009): IT policy The long road to Computer Science education reform. *Commun. ACM* 52, 9 (Sep. 2009), 33-35.