A case study of the Introduction of Computer Science in NZ schools[1]

TIM BELL, University of Canterbury
PETER ANDREAE, Victoria University of Wellington
ANTHONY ROBINS, University of Otago

For many years computing in New Zealand schools was focused on teaching students how to *use* computers, and there was little opportunity for students to learn about programming and computer science as formal subjects. In this paper we review a series of initiatives that occurred from 2007 to 2009 that led to programming and computer science being made available formally as part of the National Certificate in Educational Achievement (NCEA), the main school-leaving assessment, in 2011. The changes were phased in from 2011 to 2013, and we review this process using the Darmstadt model, including describing the context of the school system, the socio-cultural factors in play before, during and after the changes, the nature of the new standards, the reactions and roles of the various stakeholders, and the teaching materials and methods that developed. The changes occurred very quickly, and we discuss the advantages and disadvantages of having such a rapid process. In all these changes, teachers have emerged as having a central role, as they have been key in instigating and implementing change.

Categories and Subject Descriptors: K.3.2 [**Computer and Information Science Education**]: Computer Science education

General Terms: Design

Additional Key Words and Phrases: High school, Computer science education, Darmstadt model

ACM Reference Format:
Tim Bell, Peter Andreae, and Anthony Robins, 2014. A case study of the Introduction of Computer Science in NZ schools. *ACM Trans. Comput. Educ.* 0, 0, Article 0 (2013), 30 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

In 2011 New Zealand introduced Computer Science as a mainstream subject available to students in their final three years of high school [Bell et al. 2010; Bell et al. 2012]. Like several other countries, prior to this introduction the focus had been on teaching students to *use* computers rather than to be developers; at best, some schools taught a little programming.

The new Computer Science content that was introduced not only covers a much improved programming focus but also gives students the chance to explore a range of computer science topics beyond programming, including algorithms and complexity, human-computer interaction, encryption, artificial intelligence, formal languages,

[1]This case study is partly based on prior studies: [Bell et al. 2010; Bell et al. 2012; Thompson et al. 2013; Thompson and Bell 2013]

Authors' addresses: Tim Bell, Department of Computer Science and Software Engineering, University of Canterbury; Peter Andreae, School of Engineering and Computer Science, Victoria University of Wellington; Anthony Robins, Computer Science Department, The University of Otago.

DOI: http://dx.doi.org/10.1145/0000000.0000000

computer graphics and more. These topics are not intended to be covered in a great deal of detail; the emphasis is on breadth rather than depth, with the main goal being to enable students to understand what computer science is about, and thus to help them make informed career choices with an understanding of computer science that is beyond just programming [Denning and McGettrick 2005]. The topics that appear in the standards are loosely based on the ACM/IEEE computer science curriculum [Cassel et al. 2008; Sahami et al. 2013], and provide students with an overview of the kinds of topics that computer scientists have expertise in.

In addition to making the subject visible in schools, the changes were delivered in a way that provided assessment that could count towards academically-oriented qualifications (e.g. entrance to university), as previously the qualifications available were not appropriate for the more academic students, and thus such students were often advised to avoid computing at school, which in turn provided them with the wrong impression of the nature of the discipline of computer science as a non-academic subject.

The new content was phased in very quickly, and a significant challenge with this wide range of topics and highly compressed timescale has been to prepare teachers to deliver a range of material that few of them have encountered before. Teaching programming (which few teachers were experienced with) presents a unique set of challenges well known in the Computer Science Education literature, and many of the topics outside of programming lacked a body of coherent resources for teaching.

In this paper we will look at how these challenges were addressed and identify lessons learned from the changes, we will report on the current state of uptake of the new standards and the reactions of the various stakeholders, and we will make recommendations for others going through similar transitions. Some of the statistics reported here are from a survey of teachers at the start of 2012 [Thompson et al. 2013], which was repeated in May 2013 [Thompson and Bell 2013], asking the same questions to see how teachers' views had changed, and to collect data on the new standards that had been made available between the two surveys. We refer to them below as the 2012 and 2013 surveys respectively; the 2012 survey reflects the situation after one year of teaching the standards, and the 2013 survey was late enough in the year to pick up information on the full three years over which the new content was released. The surveys were announced through a national digital technology teachers' mailing list, and comparative statistics were made by taking only submissions from teachers in schools that made submissions to both surveys. More details about the methodology and initial results are available in the earlier report [Thompson et al. 2013].

This paper addresses the main issues using headings from the "Darmstadt model" [Hubwieser et al. 2011], with specific subsections for each of the issues in the model. Table I lists the sections where each is discussed.

Section 2 introduces the context of computing in New Zealand high schools, and the process by which computer science was introduced. Section 3 gives details about the new content, and Section 4 discusses the process by which it was implemented. Section 5 reviews issues surrounding assessing the subject in the New Zealand context, and Section 6 reviews how various stakeholders responded to the new standards. We conclude with a summary of key issues and lessons learned from the process of the change in Section 7.

2. COMPUTING IN NZ SCHOOLS

After some initial use of computer programming as part of the maths curriculum from 1974–1985, the curriculum became dominated by teaching students how to *use* computers. Added to this, the confusion of computing education with *computer aided education* (CAE), which has evolved into "e-learning," meant that computers in schools were heavily in demand, but were not associated with teaching computing as a disci-

Table I. Index of sections relating to the Darmstadt model

Category	Section
Educational or school system	2.1
Specific socio-cultural related factors (history of ICT and CS in school, age, gender, social and Immigration background of students, family socialization, public opinion, techno-economic development)	2.2
Research, funding, education policies, quality management	2.3
Intended learning objectives, competencies or standards	3.1
Intended knowledge about computer science	3.2
Curriculum issues	3.3
Education, qualification and professional experience of teachers	4.1
Motivation of students and teachers	4.2
Applied, proposed or developed media (textbooks, soft- or hardware tools, visualization or other didactical software, unplugged media)	4.3
Applied or proposed teaching methods	4.4
Technical infrastructure	4.5
Examination and certification of students	5.1
Related extracurricular activities	6.1
Results, outcomes or consequences	6.2

pline. This heavy use of computers for purposes other than teaching computer science gave decision makers false confidence that computing education was being addressed in the education system, and made it difficult for those who were concerned about teaching computer science to make a case for change. Combined with the view held among some computer scientists that teaching programming badly might do more harm than good, it was not unusual for academics to advise that computing not be taught at all [Koblitz 1996]. The vocationally-oriented courses that transpired also led to a view among academically oriented students that they should avoid computing at school, as the qualifications available were not suitable for entrance to university or getting scholarships, and by the time they were leaving school they would be doing well in other subjects and be more naturally inclined to carry on in those subjects.

In this section we first review the context of education in New Zealand, and then look at the process that led to computer science being introduced as an academically rigorous subject in schools.

2.1. Educational or school system

The population of New Zealand is around 4.4 million people, with 342 secondary schools and 155 "composite" schools (schools that are a mixture of primary and secondary classes, typically small rural schools that might not offer all year levels). Because New Zealand is relatively small, is independent, and has a unified education system under the control of the Ministry of Education, the country is able to introduce innovations in education relatively quickly. In the case of computer science, the time from when a panel was convened to look at concerns about the lack of computer science in schools, to the time when computer science started being taught was 2 years and 2 months. The fast time frame was partly due to the opportunity to build on a framework that had already been developed, but also was driven by a sense that the window of opportunity needed to be seized while there were government officials and contractors who understood the issues and had permission to act, teachers who were willing to help make changes, and support for change from tertiary education institutions and industry. All of this occurred in an environment where the various stakeholders were willing to work in partnership with each other.

Schooling in New Zealand is generally divided into primary ("Year 1" to "Year 8"), and secondary ("Year 9" to "Year 13," often referred to as high school). Students start school on their 5th birthday, so the typical age group of a Year can be crudely estimated by adding four to the year number (e.g. Year 9 students are typically around 13 years

old). The main school leaving qualification for students is the "National Certificate of Educational Achievement" (NCEA), which is the main focus in Years 11 to 13 of high school for most students. The NCEA qualification is flexible, and can be endorsed in various ways depending on combinations of small modules (referred to as "standards") that students pass. The standards that are used for assessing NCEA are typically equivalent to a few weeks of work in a course, with students usually taking about 5 courses at a time. The standards are usually taken in years 11 to 13 of a student's schooling, which are categorised as levels 1 to 3 for assessment, and confusingly, the curriculum on which teaching is based is referred to as levels 6 to 8.

In New Zealand two separate agencies manage high school assessment and teaching respectively. The standards for assessment are managed by the New Zealand Qualifications Authority (NZQA), which is a government agency that is responsible for quality assurance of assessment for secondary and non-university tertiary qualifications. It works closely with the Ministry of Education, which is responsible for shaping and resourcing the education system, and in this context provides resources and support for teaching the subjects.

The NCEA qualification allows schools to put together courses to suit local interests and the needs of students, based on a selection of the relatively small assessment standards (note that the standards only specify the knowledge or skills that are to be assessed; this obviously influences the content to be taught, but there is a separation in principle). Students gain credits by achieving the standards, which are generally worth 3 to 6 credits each. One credit corresponds to about 10 hours total of teaching, homework and assessment. Several standards are grouped together for a course to make up about 18 to 24 credits; 80 credits must be passed each year to be awarded NCEA (amongst other requirements), with students typically gaining about 100 credits; 120 credits corresponds to a full workload in a year. The combinations of standards are flexible, so (for example), a computing course might combine a programming standard with a web page standard to investigate web programming, but the same programming standard might be combined with an electronics standard in a different course to explore robotics. Full details of the requirements for NCEA are available from the NZQA website.[2] There are two types of standards: unit standards, and achievement standards. *Unit* standards are competency based, having a simple pass/fail outcome, and are generally associated with a skill that a student can either demonstrate or not. *Achievement* standards can be passed at three levels: Achieved (A), Merit (M), and Excellence (E). *Achievement* standards tend to be favoured by more academically-oriented students, and are important for gaining entrance to university and applying for scholarships. *Unit* standards are primarily associated with Polytechnics and Private Training Establishments, but high schools have been able to offer many of them.

The New Zealand school year starts in January/February (the end of summer), and student results are released in January of the following year, so the academic year corresponds to the calendar year. The changes discussed here were introduced in the 2011 to 2013 school years, so at the time of writing (mid 2013) all of the changes have been introduced, but student results are available for only the first two years. The two teacher surveys that we have used reflect the situation at the end of the first year of changes (start of 2012), and part way through the third year (May 2013), by which time all the standards were available.

2.2. Specific socio-cultural related factors

In this section we explore the origins of the new computer science standards, and how public perception lagged behind the changes. Socio-cultural factors are affected by mis-

[2]www.nzqa.govt.nz/qualifications-standards/qualifications/ncea/

understandings about what computer science is (both in the school system and with the general public), nervousness among teachers at having to teach new topics that are outside of their experience and have not been taught before, and the range of confusing terminology surrounding the discipline — even the word "technology" is understood in some contexts to mean only digital devices, whereas the New Zealand curriculum adopts a broad definition of "intervention by design to expand human possibilities." Combined with the strong views that many people hold about education, and the importance of getting curriculum design right, these factors inevitably create tensions, and a key element of the successful deployment of a new curriculum in New Zealand is that the many parties involved have taken the time to listen to each other and find commonalities that have made progress possible.

2.2.1. History of ICT and CS in schools. From 1974 to 1985, programming was able to be taught in NZ high schools through the maths curriculum. However, it did not emerge as a discipline connected to computer science, and despite concerns being raised [Sallis et al. 1990; Brown 1998; Savidan 2003], the subject in schools became dominated by using computers as a tool. One local approach to foster interest in schools, started in 1995, was the University of Waikato scholarship for final year school students, who could take written and practical exams to win a scholarship that included access to advanced courses at the university. This has continued to the present, but is mainly of interest to students in the Waikato region.

In 1995 a new school curriculum was introduced that included "technology" as a learning area, and ICT was defined within that area with a broad and literal view of it being about "information" and "communication," but there were concerns that schools were not engaging with the topic [Savidan 2003]. Some progress had been made towards a broader and deeper approach to computing in the curriculum, such as the "Fluency in IT" (FITNZ) project [Clear and Bidois 2005], but mapping such proposals onto a new national technology curriculum that was being released in 2007 was proving to be problematic.

Around this time the only national assessment available in computing was through *unit* standards, and these tended to be focused on applied computing rather than the discipline of computing and developing new systems — the titles of unit standards include "Produce simple desktop published documents using templates," "Produce a spreadsheet from instructions using supplied data" and "Find information using the Internet." Some programming unit standards were available e.g. "Create and use simple command sequences in a computer language" (a level 1 standard), and "Create a computer program to provide a solution" (level 3). However, these standards did not expose students to the richness of the discipline of computer science, and had a fairly dated view of what a program might be.

Another problem with the *unit* standards was that they were a simple pass/fail assessment and are generally associated with skills-based courses. For academically oriented students this was unattractive because it did not afford the opportunity to build up a portfolio of grades that could be used to distinguish their achievements when applying for university and/or scholarships. Academically inclined students favoured *achievement* standards, available for subjects such as maths and physics, where students could get a grade of Achieved, Merit or Excellence, and could demonstrate their ability as a high achiever.

Also at this time, industry and universities were developing an acute awareness that the number of students studying computer science had plummeted since the year 2000, as it had in other western countries, and that a dire shortage of skilled graduates was looming.

Two major changes were initiated in 2007. The first was a new curriculum for New Zealand schools[3], which provided general guidelines in eight learning areas, one of which was technology. Technology had been introduced to the curriculum in 1995, but had significant revision for the new curriculum released in 2007. The learning area of technology provided generic assessment tools for teaching subjects ranging from food technology to digital technology, which meant that computing was not a subject area of its own, and the same kind of assessment criteria had to be used for a large range of technologies (such as meal planning and software development).

The second change in 2007 was that the FITNZ project [Clear and Bidois 2005] project led to a new "Digital Technologies Guidelines" project[4] which piloted new material for teaching computing-related topics in schools. The DTG project proposed seven "strands" for computing in schools: Electronics & controls, Programming & software, Digital information, Digital media, Digital infrastructure, Digital society, and Digital concepts & tools. From 2007 to 2010 the DTG guidelines were piloted in three phases, with 14, then 60 and finally 120 schools.

However, in 2008 after the first phase of the DTG, and as the generic technology curriculum was being considered, concerns were raised by teachers, industry and tertiary institutions.

Industry and tertiary concerns were raised through a report released in April 2008 from the New Zealand Computer Society [Grimsey and Phillipps 2008] (the NZCS is the main national organisation for computing professionals; it has since changed its name to the Institute of IT Professionals NZ, or IITP). The report addressed two main questions: how appropriate the assessments in the technology curriculum were for preparation for a degree in computer science, and how appropriate they were for students who were going to be computer users. The main concerns in the report were around the use of generic "technology" standards that were designed to cover a range of technologies, with broad titles such as "Develop a technological solution to address a given brief" and "Present an outcome developed through technological practice," and the use of language that refers to physical materials that does not obviously relate to computing, such as "batch production" of the final product. There was also concern about the cognitive level required by the standards, and the assumption that "the food technologist, the wood worker, computer scientist and IT technician use the same cognitive skills in each of their disciplines." The computing requirements could be interpreted as a major software engineering project, which is quite a different enterprise to, say, designing and building a wind turbine for charging batteries on a boat. The report concluded that computer science should not be taught as a part of the technology curriculum, and pressed for computer science to have its own specific standards that relate to topics beyond programming, giving many examples from international sources (such as the CSTA and ACM curricula) including algorithms, AI, the limits of computation, and user interface design. It also recommended that the subject be made available in a way that would "appeal to brighter students."

Concerns were also raised through a national teachers' association; in a newsletter on 15 May 2008 the Post-Primary Teachers' Association (PPTA) announced in a bulletin "members on the group have become concerned about the workload expectations around the DTG, the tight and unsympathetic timelines and the lack of relevance of the resources they are being required to develop," and advised members to not participate "pending assurances about the development of a better process." However, by the end of June it was reported that "teachers have now withdrawn their protest after the ministry agreed to tackle some of their concerns."

[3]http://nzcurriculum.tki.org.nz/Curriculum-documents/The-New-Zealand-Curriculum
[4]http://dtg.tki.org.nz/

In August 2008 a second report appeared that had been commissioned by the Ministry of Education and written by three teachers [Carrell et al. 2008]. It considered the state of computing education, particularly in the context of the recent changes and the "disparity between graduate numbers from tertiary education and the employment needs of industry," a phenomenon that was also a concern in the US and UK with plummeting enrolments in computer science at a time when demand for graduates was growing. The report observed that "there is a lack of teacher confidence because the subject Computing is perceived to be second rate, is uncoordinated, under resourced, unsupported, lacking in a professional body, and in dire straits." It recommended that a solution could be built on the existing work done on the DTG, and in the curriculum computer science should have its own standards, either in a separate learning area, or as a separate set within the technology curriculum. The issue of teacher professional development was also raised: "investment in teachers in our subject area is long overdue," and there was a call for a national subject association to support teachers.

These concerns 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[5]. The panel's report included a joint announcement between the panel and Ministry of Education, indicating that the recommendations were broadly accepted and would be actioned urgently (to be used in schools in 2011).

The DTEP report distinguished between "ICT as a discipline" and "ICT as a tool," noting that the discipline was needed for the country to be an innovator. It recommended that a new area called "Digital Technologies" should be developed, which ideally would be a learning area of its own, but because of the time this would take, should initially sit within the Technology curriculum, but with its own achievement standards.

The general structure of the DTG areas was kept, but the Programming & software area was renamed "Programming and Computer Science" to make computer science visible as a discipline, and to indicate that it is more than programming. Retaining the term "computer science" in the title required some vigilance, as decision makers editing the standards might not see any difference between "programming" and "computer science," and thus remove the latter in the interest of conciseness; or educators involved in the process might be comfortable with programming but concerned about introducing unknown topics that seem overly academic, and prefer to make it less visible. For similar reasons, when a four-letter abbreviation was required a case was made to call it "PRCS" rather than the initial "PROG" that was chosen. It was helpful to have simple examples on hand to demonstrate what is meant by computer science, and industry advocates who had achieved well because of their computer science background were also helpful.

Building on the DTG, rather than starting from scratch, meant that the new material could be delivered taking advantage of the existing momentum achieved by the DTG. The DTEP report also called for assessment standards that were academically challenging and would help students to meet entrance standards for tertiary study, and urgent professional development for teachers.

At the same time, the Ministry of Education was in the middle of an extensive review of all curriculum areas which involved revising or rewriting all the Achievement Standards used in secondary schools, with significant changes in the Technology area. Most importantly, they were introducing a range of new knowledge and skills

[5]The body of knowledge, DTEP report and press release are available from http://technology.tki.org.nz/Curriculum-support/Knowledge-and-Skills-documents

standards to complement the generic standards. In response to the recommendations from the DTEP and other reports, it introduced a new subject of Digital Technologies into the Technology curriculum area and formed groups to write new resources and new Achievement Standards for Digital technologies, alongside the revision of existing standards in other parts of technology.

A key result of all this work was a matrix of assessment standards[6] that includes a set of generic technology standards, but has specific standards in four areas of technology: Construction and Mechanical Technologies (focuses on making and knowing how to make products and devices), Design and Visual Communication (focuses on where visual literacy and creative thinking is developed, using visual communication techniques), Digital Technologies (focuses on applying and knowing about computer science, electronic and digital applications), and Processing Technologies (focuses on formulating and knowing how to formulate processed products).

The Digital Technologies standards allow schools to offer a range of different computing related courses, but most importantly allow academically strong courses that address programming and computer science in a way that will lead into academic study in computer science and software engineering at the tertiary level. Furthermore, some of the generic technology standards can be used within a digital technologies context and be combined with the Digital Technologies standards to put together a course that teaches subject knowledge, and then applies or reflects on it in a practical environment. An example of a generic standard that could be used this way is "Demonstrate understanding of the ways a technological outcome, people, and social and physical environments interact," in which students might look at issues such as privacy and equity in a digital environment.

2.2.2. Public opinion. A significant challenge for schools has been to shake off the perception of computing, arising from the previous curriculum, as a non-academic subject that is not appropriate for students heading for tertiary qualifications, and containing little that most students do not know already. Some of the new courses are now too challenging for those who would have been encouraged to take computing previously, and it has been important to keep options in place for those who do need to learn how to *use* computers rather than develop new systems. At the same time, it has been hard to change the perception of the courses fast enough to attract a new group of students, including those who are heading for university qualifications. For example, for schools that were reported in both the 2012 and 2013 surveys, 45% reported that a significant number of students took the course who lacked the academic ability to do it well in the first survey, and in 2013 the figure was 46% for the same schools (although we note that the number who felt they attracted a significant number of highly capable students rose from 41% to 50%.) It is a challenge to communicate the changes clearly to students and their parents because they are getting information from a wide variety of sources, such as career counsellors, deans, principals, parents and the press, and not all of those sources will necessarily understand the changes.

A strong media campaign is required to address this; in New Zealand there were a number of press releases and public articles announcing the changes and explaining their value to students, industry and the country, but there were still two parallel public perceptions. For example, in March 2012, over a year after the new standards had been deployed, a teacher was quoted in the media saying "the current ministry-provided ICT curriculum for senior students (Years 11, 12, and 13) focuses on outdated and 'static' IT skills, instead of general 'IT thinking' like programming theory, strategy, and logic; which would prepare students for higher education in ICT and careers in

[6]Available from http://ncea.tki.org.nz/Resources-for-aligned-standards/Technology

the sector," [7] and in April 2013 a student wrote an opinion piece that appeared in a major newspaper saying that "Information and communications technology (ICT) has been trampled by English, maths and the sciences as a subject that yields no value in progressing to the next stage of education." [8]

Having champions in the media (such as high-achieving industry people) is a valuable way to address misunderstandings about the courses, but another issue is that their terminology may not match the courses that are being offered. The key phrase that was needed to promote the changes in schools was "Computer Science and Programming," since that is the name of the standards that were put in place, but often media articles would mention terms like ICT, Information Science, computing, coding and software engineering, all of which are legitimate terms, but do not link directly to the offerings in schools. One initiative to address this was a brochure and web site made available to schools to help them promote the connection between the new courses and the employment opportunities for students.[9] Also, the Ministry of Education is launching a "Vocational Pathways" project to help students become more aware of the options available to them.[10]

2.3. Research, funding, education policies, quality management

From the early 1990s the CS Unplugged project (at the University of Canterbury in Christchurch, University of Waikato in Hamilton and Victoria University in Wellington) had developed experience in communicating computer science without using computers, and in environments where little time was available to learn programming [Bell et al. 1995; Bell et al. 2012b]. While this did not drive the changes, it had provided a base of expertise working with schools that provided confidence that advanced topics from computer science could be communicated to school students without first requiring them to have significant programming skills, and was influential in making it possible to have computer science as a separate standard that was not dependent on programming standards. Also, because it had been used with primary school children who were able to work with concepts from computer science, it gave some confidence that high school students would be able to do so as well.

Research on teaching introductory programming had been carried out at the university of Otago since the early 2000s, which provided considerable background to inform the design of school programming standards [Robins et al. 2003; Rountree et al. 2004; Garner et al. 2005; Robins 2010].

Similarly from the mid 2000s various initiatives to promote robotics in schools had been independently established, the main one being RoboCup Junior[11], which runs annual regional and national competitions for schools involving hundreds of pupils. Robotics promotes programming and a range of other engineering and technology skills, and this practical experience with schools (at both primary and secondary levels) informed the design of the new programming standards.

Research underpinning the changes in the technology learning area of the school curriculum, focusing on the broad area of technology more than computing in particular, includes a review of teachers adapting to the 1995 introduction of technology in the curriculum [Compton and Jones 1998] and reviews and case studies and the development of a framework for technology assessment [Compton and Harwood 2003; Compton and France 2007]. The research previously mentioned that focuses more on

[7] http://computerworld.co.nz/news.nsf/news/govt-failing-ict-teachers-says-tech-educator
[8] http://www.stuff.co.nz/technology/digital-living/8577142/Education-not-in-sync-with-IT-goal
[9] Published on http://csanz.ac.nz
[10] http://youthguarantee.net.nz/vocational-pathways/
[11] http://www.robocupjunior.org.nz/

computing and computer science in the curriculum [Clear and Bidois 2005; Grimsey and Phillipps 2008; Carrell et al. 2008] provided the data to drive political change.

The computer science standards have been tracked through two major research projects: a detailed analysis of student work [Bell et al. 2012a] and a survey of teachers to identify good practices and issues during the transition [Thompson et al. 2013]. Outcomes from this research are discussed below, including results from an updated survey of the teachers.

Student work is assessed by the NZQA government agency, which tracks student performance and makes general statistics publicly available online.[12] Some of these statistics are reported later in this document.

Funding, of course, bears on the deployment of the new standards in several ways. The development of the new standards required a budgetary commitment from the Ministry of Education, which included costs for bringing together experts to develop the standards, exemplars and guides. There has been little central government funding for schools for professional development (PD) of teachers specifically relating to the transition. All schools are expected to set aside some of their budget for PD, and it is up to each individual school how to deploy it, with predictably variable results, particularly given that administrators might not understand how much help teachers might need to teach the new standards. (There is also a limited contestable pool for whole school programmes, but this cannot be applied to PD within a specific subject area.) There has, however, been indirect support from the Ministry of Education in various ways as described in Section 4.1. Some sponsorship for PD has been forthcoming from industry, particularly from Google Inc. for funding the CS4HS PD workshops, also described in Section 4.1, and development of the "CS Field Guide" (Section 4.3).

3. DETAILS OF THE NEW STANDARDS

In this section we describe the eight new achievement standards that comprise the "Programming and computer science" strand of Digital Technologies, and discuss some of the issues that arose around details of the standards.

3.1. Intended learning objectives, competencies or standards

The official source for the new standards is the New Zealand Qualifications Authority (NZQA), and all Digital Technologies standards can be accessed via http://www.nzqa.govt.nz/ncea/assessment/search.do?query=Digital+Technologies.

Table II summarises the "Programming and computer science" standards. There is a stream of the standards that focus on programming (1.45, 1.46, 2.45, 2.46 and 3.46), and another stream that are focused on topics from computer science (1.44, 2.44 and 3.44). In principle the computer science standards can be taught without the programming standards, but most classes combine them, or teach only programming. For example, in the 2013 survey, only 17.7% of teachers reported teaching the 3.44 standard without the 3.46 standard, and at level 1 the corresponding proportion was only 9.1%. In contrast, 80.0% of the teachers using 3.46 (programming) are also using 3.44.

The *programming* standards are all "internal," which means that they are assessed by the teacher, with external auditing (moderation) to ensure consistency. The *computer science* standards are all "external," which means that the they are assessed by an anonymous appointed marker (or group of markers), organised by NZQA. There are requirements for a minimum number of external credits on a student's record to meet various criteria and endorsements; it is therefore important that the computer science standards are external, since this encourages teachers to include them in their courses

[12]http://www.nzqa.govt.nz/studying-in-new-zealand/secondary-school-and-ncea/secondary-school-statistics/

Table II. Achievement standards in the "Programming and computer science" strand of Digital Technologies

DT number	NZQA Standard number	Title	Credits	Assessment
Level 1				
1.44	AS91074	Demonstrate understanding of basic concepts from computer science	3	External
1.45	AS91075	Construct a plan for a basic computer program for a specified task	3	Internal
1.46	AS91076	Construct a basic computer program for a specified task	3	Internal
Level 2				
2.44	AS91371	Demonstrate understanding of advanced concepts from computer science	4	External
2.45	AS91372	Construct a plan for an advanced computer program for a specified task	3	Internal
2.46	AS91373	Construct an advanced computer program for a specified task	3	Internal
Level 3				
3.44	AS91636	Demonstrate understanding of areas of computer science	4	External
3.46	AS91637	Develop a complex computer program for a specified task	6	Internal

(although the external credits might be obtained in standards other than these ones, since there are few rules about how standards can be combined to create a course). The external assessments are done by students submitting reports that should reflect their experience with the topic, rather than an exam. This is discussed further in Section 5.1.

Each credit represents a nominal 10 hours of student work (both in and out of class time), each standard is usually 3 or 4 credits, and about 18 to 24 credits are required to make up a course, so even if a class does all the programming and computer science standards for a level it will typically only be about half a course. The difference can be made up with either generic technology standards (looking at broader issues in computing, or technology in general), or with other digital technology standards, such as electronics, media (including web design) and information. The small number of credits means that it is a lot easier to slip a computer science standard into a course even if they are not the main focus (for example, a course on web design could use them to create a coherent course that includes programming and interface evaluation), and this has the benefit of exposing more students to the topic, even if they would not have chosen it themselves (although we note that in the surveys many teachers commented that the number of credits is too low for the amount of work required).

Each of the computer science standards covers topics in computer science, generally corresponding to areas of the ACM/IEEE curriculum [Sahami et al. 2013]. Table III identifies the main topics for each standard. For 1.44 and 2.44, all topics must be addressed by a student to achieve the standard. For 3.44, students need only look at two of the topics in detail; since the goal of the standards is to give students an understanding of what computer science is, the process of choosing the two topics will usually be an opportunity to review the breadth of the field, without creating too much "busy" work for students to look at each of the topics in detail.

The goal of the computer science topics is that students should understand the main issues, even if they do not have the skill to work with software that uses the concepts. For example, with the 1.44 algorithms topic students should find out that the difference between algorithms such as quicksort and selection sort, or binary search and sequential search, is not just a linear factor, and can have a huge impact on perfor-

Table III. Topics in the computer science standards

DT number	Topic	Level of detail
1.44	Algorithms	Understanding the difference between a program and an algorithm, and evaluating two different algorithms for the same problem, for example, by timing selection sort and quicksort for various values of n
1.44	Programming languages	Understanding the role of compilers, interpreters, and high/low level languages
1.44	User interfaces	Evaluating usability based on activities such as observing a user completing a task using a think-aloud approach
2.44	Data representation	Binary, representation of numbers, text, images, and sound, and especially the implications of the choice of the number of bits (e.g. 8/16 bit characters, 16/24 bit colour)
2.44	Coding	Covers encryption, error detection/correction, and compression, at the level of evaluating software and systems that use these techniques on practical systems (for example, measuring the amount of compression for various kinds of images and compression methods)
2.44	Usability heuristics	Evaluating an interface in the light of usablity heuristics such as those given by Nielson (useit.com)
3.44	Formal languages	Familiarity with the notation for simple languages such as regular expressions and their corresponding Finite State Machines (using provided software to convert between the two)
3.44	Network communication protocols	Understanding the issues being addressed to ensure reliable communication of data between two parts of a network in the face of different kinds of threats and failures
3.44	Complexity and tractability	Understanding that there are problems that cannot feasibly be solved using computers, and the exponential costs that can arise from brute force exhaustive searches
3.44	Intelligent systems	Understanding the role of various concepts in AI, such as the Turing test, and experimenting with implemented systems, such as chatbots and machine learning
3.44	Software engineering	Understanding the challenge of "programming in the large," and the kind of processes (e.g. agile) used to create software
3.44	Graphics and visual computing	Understanding the kinds of techniques used to create and understand images, such as transforms, simple rendering techniques, and image processing algorithms.

mance and scalability. However, they are not required to implement the algorithms, although more capable students could be encouraged to do so. It would be reasonable for students to understand how selection sort works through animations or physical demonstrations, and they might even be able to understand the principle of quicksort, but the key concept being taught is that the wrong algorithm can be significantly slower than a carefully chosen one. Likewise, for usability the main goal is for students to become sensitive to how even the most polished interface can have usability issues, and that this is an important area to master when designing interfaces, but they are not required to design and implement an interface.

The list of topics at level 3 can look daunting to teachers without a background in computer science, but there are ways to approach each topic without requiring a strong background, and yet still understanding the key issues surrounding the topic.

The programming standards represent a progression from introductory work at level 1 (often taught using Scratch), through to the equivalent of an introductory university course at level 3. Level 1 addresses tasks that involve input and output, and can be expressed as a single procedure (or method/function) program using sequence, selection and iteration, but only requires simple data (no arrays, lists, or structures). Level 2 addresses tasks that involve multiple procedures and also use an indexed data structure. Level 3 requires the use of basic object-oriented programming concepts (classes

and objects with encapsulation, but not inheritance) and a simple GUI implementation with event handling.

For level 1 and 2, the tasks are split into two parts: designing a program for a task, and implementing a design as a program. The "design" standards (1.45 and 2.45) involve producing an "algorithmic structure" which can be written in informal English, pseudocode, or a graphical format. The "implementation" standards involve constructing a working program in a programming language and testing and debugging it. The two would normally be taught together, but having separate standards makes it possible for a student show their relative strengths in the two areas (for example, a student might not be able to come up with a good design, but could show that they can implement a program given a design, and thus get credit for the 1.46 standard only even though they attempted both). This split is somewhat controversial, and the reasons for it and the problems with it will be discussed in a later section.

At levels 1 and 2, drag-and-drop languages are allowed (Scratch is popular at level 1, with many schools moving to text-based languages at level 2). At level 3, students must use a text-based programming language (Python is the most popular language at levels 2 and 3).

3.2. Intended knowledge about computer science

Part of the purpose of the new topics being introduced to schools is to provide students with some grounding in CS concepts, but the primary goal is giving them exposure to the topic, providing the opportunity for them to find out if it is something they might be passionate about. Significant factors in the declining interest in CS amongst school students in western countries include the widespread misunderstanding of the subject and the career [Woratschek and Lenox 2009; Tucker 2010], and the confusion of the discipline of computing with learning how to use the computer as a tool. Providing a lightweight overview of the topic is intended to counter this by giving students the opportunity to explore a broad range of areas of computer science without taking the risk of committing too much time to engage with the subject. The relatively small number of credits also helps because the material can be integrated in existing courses rather than having to be a course in itself.

Another important reason to keep the amount of content fairly light is that universities currently accept students with little or no computing background, so if some students who happened to do CS at high school had done up to three substantial years on the topic, this could create a diversity of incoming students that would be difficult to cater for. If the first year university course starts with introductory programming then the specialist students would be bored, and if it assumes three years of CS background, it would be beyond what less-prepared students could cope with. The net effect could be to significantly reduce the number of students taking CS at university level!

Importantly, starting small enabled the topics to be deployed quickly, reducing the time needed for teacher training, developing supporting resources, and administrating the new programme. In the long-term when CS has become well established at high school this may become less of an issue, and computer science in schools could grow more into a more substantial programme, to be comparable with subjects like physics, where a substantial high school background is usually assumed for students who wish to study it at university.

The process of designing the standards was informed by similar initiatives overseas, and considerable attention was paid to discussions that led to curricula such as the US CS principles project [Astrachan et al. 2012], the Exploring Computer Science course [Goode and Margolis 2011], and the UK curriculum changes [Furber 2012], as well as programmes in Israel [Gal-Ezer et al. 2009] and Korea [Yoo et al. 2006], where computer science was already well established in schools.

3.3. Curriculum issues

Here we explore three curriculum issues that come up as a result of the changes.

3.3.1. Workload and preparation. The number of credits that each standard is worth has been the topic of considerable debate. NZQA has guidelines that restrict the number of credits, and the benefit of a lower number of credits is that the standard can be more easily combined with others to expose more students to the topic. However, teachers have repeatedly reported that the amount of work done by students to meet the new standards is out of proportion with other learning areas. This may be partly due to the subject being new, and the effort required by all involved in this unfamiliar area is higher, and it will also be affected by the fact that students are meeting the topic for the first time quite late in their schooling (whereas other subjects such as maths will have been gradually scaffolded from very early on).

Some have taken an optimistic view, that the subject will be seen as challenging but worthwhile; in the 2013 survey comments one teacher observed: "... most (by far) students do cope, and even if they really want [Merit or Excellence] but don't actually think at that level (like I said, it is hard), they accept that just having the Standard means an awful lot."

The focus of the changes has been on NCEA, as this is where standards are set. However, since this only touches the last three years of school, the next challenge will be to introduce some of the concepts earlier, which can significantly reduce the pressure on students once they reach NCEA, and also provides more awareness of the subject earlier on. In the 2013 survey one teacher commented: "This year my year 9 students did the the same HTML learning as my year 11 students using the CodeAvengers website and they produced equal results in less time." Younger students can be more open to learning the new ideas, partly because of their developmental stage, and partly because they are not under pressure to learn for assessment.

Teachers are permitted to teach this material earlier, and it can fit into existing broad curricula guidelines, but in the absence of formal guidance it is up to each individual to work out how to achieve it. Once the advanced curriculum has settled it is likely that more work will be done to explore more structured approaches to introduce programming and computer science earlier in the curriculum.

3.3.2. Choice of programming language. The choice of programming language has been left to teachers, and this has been useful since they can choose languages that will work in their context. This choice of languages was reported in both the 2012 and 2013 teacher surveys. For level 1 programming (Year 11) the most popular language has been Scratch, and this increased from being used by 47% of teachers in 2012 to 62.4% in 2013. Python was the second most popular language, increasing from 4.1% in 2012 to 25% in 2013; this is most likely due to it being popular beyond level 1, and only few teachers choosing to use it at the introductory level rather than starting with Scratch and then switching. JavaScript was not used by any respondents in 2012, but 16.3% were using it in 2013, almost certainly due the recent availability of the online "CodeAvengers" lessons that were written specifically for the standards (see Section 4.3).

Beyond level 1, the 2013 preferences for languages were Python (53.8%), JavaScript (13.8%) Visual Basic (10.0%), Java (8.8%), C# (4.8%) and PHP (2.5%). JavaScript was the one language for which interest changed noticeably between the two years, and again this could be put down to the availability of a ready-packaged resource for teaching it. The choice of languages reflects the availability of resources (particularly the Python and Java textbooks and the CodeAvengers system, both described in Section 4.3).

There are some questions around the transition from a drag-and-drop language (such as Scratch or AppInventor) to a text-based language, as this is not a trivial step. A text based language is not required until level 3, but if a drag-and-drop language is used at level 2 then students have a lot of catching up to do in their final year. In particular, AppInventor may be more motivating for level 2 students, but if they intend to advance to level 3, they would be advised to learn a text-based language instead. This reflects concern elsewhere that languages like Scratch may get students into bad habits [Meerbaum-Salant et al. 2011], although if a drag-and-drop language was not permitted for level 1 then the uptake would likely have been significantly less.

3.3.3. Split programming standards. The separation of the programming standards into two parts (design and implementation) has caused much discussion. The design component (e.g. 1.45) involves specifying variables and their types, the program structure, and how the program should be tested. The implementation component (e.g. 1.46) involves implementing the program in a suitable language, documenting the program (which might be comments in the code), and debugging it.

The two are clearly interdependent, and the intention is that students would do both together, and ideally intertwined (i.e. design and implement more than one program). The benefit of the separation is that a student can get credit for one even if they do not achieve sufficiently well with the other, but the level 3 standard combines both aspects, since by that level one would expect that students could do both tasks. At level 1, weaker students in the class might focus only on the implementation, and be provided with substantial design guidance.

Another reason for a split is purely political: as a general policy, the ministry officials' strong guidance was that each standard should be 3 or 4 credits, corresponding to 30 or 40 hours of learning and assessment time. Given that a first course in programming is typically several times larger than this, to get a reasonable number of hours on the topic it would need to be spread across more than one standard.

The split also makes it easier for teachers who are teaching a broad course across several strands of digital technologies to include at least some component of programming and computer science. For example, it is more likely that a small programming standard could be included in an embedded electronics course or a digital media course. For such teachers, especially those without a background in programming or computer science, a single large standard might have been perceived as too much of a commitment.

A pedagogical reason for the particular split into "design" and "implementation" was to emphasise the importance of design by having half the credits associated with the design process. This was intended to counter the tendency of new programmers to jump into coding without thinking carefully, and to spend all their time attempting to modify their program until it works. A related pedagogical argument for this split is that many students find the design aspect particularly difficult when they start programming and need large amounts of help. Since students either get all the credits for a standard or none, splitting the standards in this way means that a student who failed the algorithm design component could be provided with a design outline which would still allow them to attempt the implementation standard to get part of the credits, and gain some appreciation for what programming is about.

Teaching the implementation standard in isolation appears to have happened, with 12% of teachers in the 2013 survey reporting that they are teaching 1.46 but not 1.45. This is reflected in the student results from 2012 (details appear in Table IV in Section 4.2.1), with 2,531 students attempting the 1.45 standard, but 3,865 (53% more) attempting 1.46. We note that this does not mean that students were not taught both areas, but may have been advised to only take the assessment on the implementation.

At level 2 the difference is not so great, with only 6% of teachers offering only 2.46 without 2.45, and 1,171 students attempting 2.45 and 1,482 (27% more) attempting 2.46. At level 3 students are required to take both combined.

A counterargument to the split approach is that at this level students should be able to design and implement simple programs, so that making it possible to do just the implementation standard will lead to students who have experienced a very limited and inadequate version of "programming." It is intended that the material in the matching standards be covered together, but as long as it is split into separate standards this can never be guaranteed. A problem in practice is that, as noted above, standards can be passed at different levels. The requirements specified for both plan (e.g. 1.45) and program (e.g. 1.46) are different at each level, and reasonably specific. Hence there is a problem with those students who are attempting the implementation standard (e.g. 1.46) in isolation (or after having failed the planning standard). Although the official advice is that students should be given Excellence level plans, there is some confusion about this, since, for example, a student who is combining the two standards and working at a Merit level would have a worse plan (their own Merit level plan) than one who had failed the 1.45 standard, or else would have to abandon their own plan and use a given one. Clearly the quality of the plan supplied will significantly influence the quality of the program produced, and thus contribute to determining student achievement. In their current form the standards do not address this difficult question, and as a result there is uncertainty in schools about how to deal with this, which can lead to inconsistent grading between schools. From a pedagogical perspective this seems quite unsatisfactory.

4. IMPLEMENTATION

In July 2009 the DTEP panel recommended an urgent deployment of new achievement standards, and by January 2011 students were attending classes that used the new standards. This rapid deployment left very little time for teachers to prepare, and although ideally a designated programme for new teachers would have been put in place [Ragonis et al. 2010], there was neither the time nor resources to recruit new teachers, so the pressure was on for existing teachers to upskill. To add to the challenge, during that period the city of Christchurch experienced a series of major earthquakes that closed schools and universities for several weeks, and brought about major disruption that continues to the present (it has been claimed to be the fourth largest insurance event in world history, and the most damaging earthquake was on 22 Feb 2011, the month that schools had just started classes using the new standards).

In this section we review issues relating to the implementation.

4.1. Education, qualification and professional experience of teachers

The 2012 survey [Thompson et al. 2013] found that 60% of the digital technology teachers were aged 50 or older. This means that it is an experienced group (68% had 10 years or more classroom experience), which they were able to use to make decisions about deploying the new standards. However, few had significant computing qualifications, with 47% reporting only rudimentary programming skills and 12% unable to program, and only 56% reported having any specific qualification in the computing field. 22% had some sort of computing degree, which provided a core of teachers who could assist with peer support.

The survey found that teachers generally had a low level of self-confidence, with only 64% feeling any confidence about teaching the programming standards, and 44% feeling any confidence about teaching computer science.

Considerable effort was put into helping teachers develop confidence and competence with the new standards. Below is a list of the main efforts, in decreasing order of use by teachers as reported in the 2013 survey [Thompson and Bell 2013]:

Teachers' association. A national teachers association, the "New Zealand Association for Computing, Digital and Information Technology Teachers" (NZACDITT), was formed in March 2009, and from the start played a key role in teacher preparation and advocacy. Teachers were able to share ideas, resources and concerns through a discussion group and a website, as well as local and national meetings.
National CS4HS (computer science for high school) workshops. The CS4HS events are annual two- and three-day workshops sponsored by Google Inc., held at a university and run by university staff and experienced teachers. These have been run internationally to engage teachers in computer science and computational thinking in general [Blum and Cortina 2007; Bort and Brylow 2013], but the NZ events started in December 2011, and were very specifically focused on the new standards.
Local teacher organisation meetings. Clusters of teachers (sometimes already started as part of the DTG, or even existing prior to that) organised regular meetings to share ideas.
Peer support. Many teachers valued individual peer support from a colleague (possibly in the same school or nearby).
Formal course. Some teachers had undertaken personal study in a formal course. A long term course is particularly valuable for learning programming, as this is not easily learned in a short block course (in fact, Peter Norvig has argued that it takes some years[13]).
Official events. Several events were organised by the Ministry of Education, such as professional development meetings.
University contacts. A network of university contacts was set up, where one academic at each of the country's eight universities was nominated as a local contact who could provide advice to local teachers, including finding senior students who could speak to classes.
Personal study. Some teachers undertook personal study using informal resources, such as online training.
Industry visits to schools. Industry programs were set up for school visits, where schools can arrange for professionals to talk to a class. Two programmes are available in New Zealand: FutureInTech[14] initiated by IPENZ (Institution of Professional Engineers New Zealand), and ICT-Connect[15], run by the IITP (formerly called NZCS, the organisation that sponsored the report that called for curriculum changes [Grimsey and Phillipps 2008]).
Online resource guide. A guide was developed early in the transition process, funded jointly by the Ministry of Education and all computer science departments in New Zealand universities [Murugesh et al. 2010]. The guide was published on the NZACDITT website.[16]
Online "CS Field guide". An online site that is intended to have a similar role to a student textbook was developed, called the "Computer Science Field Guide". It is an open-source interactive site that is being developed to provide information at the level required for the new computer science standards, including notes for teachers. Initially just over a third of the new topics were supported by the site, with more

[13] http://norvig.com/21-days.html
[14] http://www.futureintech.org.nz/
[15] http://www.ictconnect.org.nz/
[16] The material can be found at http://nzacditt.org.nz/resources by selecting the "Programming and CS" search option.

being added so that eventually it will cover all the material in the computer science standards.[17]

Helpline. A confidential "helpline" was piloted through the universities, where teachers could ask questions that they might not be comfortable asking in public because of their lack of confidence. It has had a little use, but most teachers seem to be comfortable with asking questions in the NZACDITT forum (which has fostered a supportive environment), or by approaching experts directly (such as the university contacts).

University study. A post-graduate education course that is available for teachers to learn by distance, and obtain a formal qualification in teaching computer science.

University students in class. Some teachers have obtained professional development funding from their school which they have used to hire a senior computer science student to teach beside them in the class. The teacher provides the classroom management skills, and they learn from the student in an authentic context.

From the 2013 survey it was clear that resourcing for professional development (PD) is a major issue, with only 42% reported receiving "good" financial support to undertake PD, 50% reported receiving "partial" support, and 7% were expected to fund their own PD. In the free-form comments, one of the most often mentioned issues was a lack of time or opportunities for professional development, with some teachers feeling overwhelmed or exhausted by the changes. Others have reported enjoying the new opportunities and stimulation of more interesting topics to teach.

Free-form comments in the survey indicated a high value was placed on ready-to-use content. Although many disparate resources have been identified, teachers benefit greatly from material designed specifically for the local standards. Some local resources for this were mentioned favourably, including the CodeAvengers online programming lessons[18], Python and Java workbooks written specifically for the new standards[19], and the online "Computer science field guide"[20].

4.2. Motivation of students and teachers

4.2.1. Student motivation. The main evidence of student interest is the number who have participated in the assessments. Table IV shows the number of students who attempted each of the new standards in 2012 (the level 3 standards were not available until 2013), with the largest uptake being level 1 programming, where 3,865 students attempted the 1.46 standard. There were about 61,000 students in New Zealand in year 11 in 2012, which means about 6% of those students have attempted a standard in programming. Given that there are 497 schools in New Zealand that might have year 11 students (in 2012), and that the membership of NZACDITT is about 216 people (in July 2013), and that the membership includes teachers who are not teaching the new standards, many students will not have had the opportunity to attempt the standard at their school, and so this number is as much an indication of availability as student interest.

The pass rates for the new standard (around 70%) are typical for NCEA, although we note that the number of Excellence grades is higher for the programming standards than for computer science. This may be partly due to the nature of the topic (programming courses often have bimodal outcomes [Robins 2010]. Note also that at each level more students attempt the implementation standards (1.46 and 2.46) than planning

[17]Student access is currently through http://csfieldguide.org.nz/.
[18]http://www.codeavengers.com/
[19]http://www.cs.otago.ac.nz/schools/
[20]http://csfieldguide.org.nz/

Table IV. Students sitting and grades achieved for programming and computer science standards in 2012

Standard	Number of results	Not Achieved #	Not Achieved %	Achieved #	Achieved %	Merit #	Merit %	Excellence #	Excellence %
1.44 Basic concepts	953	307	32.2	365	38.3	166	17.4	115	12.1
1.45 Plan program	2,531	738	29.2	850	33.6	447	17.7	496	19.6
1.46 Implement program	3,865	1,070	27.7	1,314	34.0	792	20.5	689	17.8
2.44 Advanced concepts	693	208	30.0	300	43.3	136	19.6	49	7.1
2.45 Plan program	1,171	377	32.2	377	32.2	191	16.3	226	19.3
2.46 Implement program	1,482	474	32.0	399	26.9	268	18.1	341	23.0

(1.45 and 2.45), a difference of 1,334 at level 1 and 311 at level 2. These students may have been given a plan to implement, but we do not know what level of plan they are being given, so this is an unknown influence on outcomes. A further factor may be the method of assessment, where programming is assessed internally during the year, whereas computer science is a report handed in at the end of the year. The analysis of student work for 1.44 revealed that some had clearly run out of time or made a rushed attempt, presumably by leaving the work too late [Bell et al. 2012a].

Participation increased between 2011 and 2012; in 2011 there were 2,213 students who passed the 1.46 programming standard, and in 2012 there were 2,795, a 26% increase. Computer science increased by a larger percentage, from 440 to 646 (47%), probably reflecting a more cautious uptake with the unknown subject in the first year. Also, the number achieving level 2 programming in 2012 (1,008) was only 46% of those achieving level 1 the previous year (2,213), indicating a large number who have just dabbled in the subject, while the number of students achieving computer science increased 10% from 440 achieving 1.44 in 2011 to 485 achieving 2.44 in 2012 (level 1 is not a strict pre-requisite for level 2), which shows a substantial increase in interest in that area, probably helped by the better provision of resources by the second year. While the numbers seem small, this would represent a significant proportion of computer science enrolments in New Zealand universities, and bodes well for increased numbers if the students continue in the subject.

4.2.2. Teacher motivation. Adoption of the new standards was not compulsory, but many teachers took on the new standards; the 2012 survey [Thompson et al. 2013] showed that the main motivation of the early adopters was to provide better opportunities for students (90%), although most also indicated that they were personally interested in the subjects (62%), that they thought it was good for the country (45%) or simply the right thing to do (47%). Only 8% were motivated by school management's requirements, reflecting the grass-roots nature of the changes.

4.3. Applied, proposed or developed media

For level 1 programming (1.45 and 1.46) the Scratch language (which was the most popular) already has a lot of teaching resources available, and free online resources are appearing for other languages. Two other resources were developed to support the new programming standards. One is a set of four programming textbooks[21], one for each of level 2 and 3, each in Python and Java (more than 100 schools in New Zealand signed up to receive them by July 2013). The other is the "CodeAvengers" interactive website[22], which offers lessons in JavaScript that cover the programming standards for level 1, 2 and 3 (about 65 schools have signed up for the CodeAvengers JavaScript courses by July 2013).

[21] http://www.cs.otago.ac.nz/schools/
[22] http://www.codeavengers.com

Both of these resources were initially offered at no charge while they were being developed, with a gradual release of sections as they became available. They are now available at a nominal fee (except CodeAvengers level 1 is free).

For the computer science topics, the task of developing resources to support teachers has largely fallen on the universities, and the computer science departments in the main universities in New Zealand contributed to developing support material [Murugesh et al. 2010], initially producing a list of thousands of resources that could be used to teach these topics at high school level, plus guidance for teachers. The resource guide included resources and teaching guides from the CSTA repository (http://csta.villanova.edu), CITIDEL (http://www.citidel.org), CS4FN (http://cs4fn.org), CS Unplugged (http://csunplugged.org) and CS Inside (http://csi.dcs.gla.ac.uk/). Other sources included suggestions gleaned from teacher mailing lists and guides for teaching computer science in other countries. Some of these resources were originally intended for outreach, and would need to be adapted for teaching a course that is being assessed [Bell et al. 2012a].

This collection became overwhelming, and spurred the development of an online "textbook" (funded by Google Inc.), which was released as a beta version in January 2013. It is an on-line open-source interactive resource that covers the topics at an appropriate level for school students and provides extensive guidance for teachers[23]. The "Field Guide" currently has just enough material to cover topics that teachers need the most help with, but ultimately it will be broadened to cover all of the computer science topics for NZ schools, and beyond, as resourcing permits.

Another source of resources (the best of which are being collected into the "Field guide") was a course on CS education being run at Canterbury university for fourth year students [Bell and Lambert 2011]. The purpose of the course was to give the students a general background in communicating their discipline, but for their major project students were encouraged to develop a resource to meet a pressing need for delivering the new curriculum, and this resulted in teaching plans for level 1 and 2 computer science that have been widely used in New Zealand, as well as resources to help deliver specific topics. Students in the course were motivated by their own experience in high school to improve the situation in schools, and found it rewarding when their assigned work ended up having a positive effect for hundreds of students.

4.4. Applied or proposed teaching methods

While teachers are free to use whatever teaching methods are appropriate for their environment, most will appreciate strong guidance and ready-to-use resources, particularly given the large amount of new material that they need to learn themselves, and then prepare to teach. This was particularly important because of the short lead-in time to prepare for teaching the new material, and the lack of prior training teachers had in the new topics. The more pre-packaged teaching material was, the easier it was for teachers to adopt it, and so there was a strong motivation to deliver such material to teachers to support the changes. The result of this was the books and on-line resources described in Section 4.3; these resources gradually appeared and were refined during the time that the standards were phased in.

Teaching methods for the computer science topics were heavily driven by the nature of the assessment, which was required to be a personalised report of 14 pages maximum (see Section 5.1 for more details). Because the writing of the report isn't done under exam conditions, students need to show authenticity by personalising the experience (an early examiner's report says that they need to hear the "student voice" in the document). To teach this effectively, the teacher needs to guide the student on choos-

[23] http://csfieldguide.org.nz/

ing a personal example such as using their own alarm clock to evaluate for usability heuristics, or testing the speed of algorithms using their own data sets. These personalised examples give the student an authentic experience where they are working with a real digital system (not necessarily one that they have programmed themselves), but are looking at the computer science behind it, and using it as an example to explain concepts to the examiner (such reporting on a friend struggling to set the time on an alarm clock, and articulating the usability issues). As long as the teacher has sensitised the student to the issues that need to be investigated, and makes sure they have chosen a sufficiently rich example, the learning will follow from the student's experience. An analysis of student projects for the 1.44 standard found that the teacher's guidance had an important influence on the student's opportunity to understand and articulate the computer science concepts [Bell et al. 2012a].

4.5. Technical infrastructure

The new standards were designed in a way that they could utilise existing infrastructure without significant costs for new software, and often more importantly, without a lot of time required to deploy new systems. The widespread use of restrictive security in school systems means that getting new software installed can be a major undertaking, and some teachers have reported that technical staff in their schools are reluctant to have students compiling their own programs and running them within the school network because of security risks. Workarounds include running the operating system off a USB stick so that the teacher has full control over the programs available.

Much of the software that supports the standards can be run in a browser, and this is a useful trend as it makes it a lot easier for teachers and students to deploy the software. For example, an in-browser version of Scratch has recently been released in beta, and teaching resources such as "Interactive Python"[24] run the programming language locally using JavaScript in the browser. The CodeAvengers site developed for the new standards uses JavaScript, which again is already available on standard computer setups. The "CS Field guide" was written using only HTML5 and JavaScript, which is compatible with most recent web browsers.

Easily deployed software (particularly if it is free, operating system independent and trivial to install) also makes it more likely that students can transfer their work back and forth from home.

When recommending resources, software that could run on multiple operating systems was favoured, and this is common with the languages that are popular in schools, although, for example, Visual Studio is not simple to get running on Linux or MacOS. The operating systems used in schools are decided locally, and not necessarily by those teaching programming and computer science. In the survey of teachers 78% reported using Windows exclusively, 8% used MacOS exclusively, 8% used a mixture of Linux and Windows, and 5% used a mixture of MacOS and Windows.

Some bandwidth may be needed for resources such as the "CS Field guide," which contains a number of videos and detailed graphics images, although it is being developed in a way that it could be delivered offline so that schools with limited bandwidth (e.g. rural schools) could download it once or obtain it on physical media to share locally.

[24] http://interactivepython.org

5. ASSESSMENT

5.1. Examination and certification of students

The achievement standards are presented as small collection of criteria that are in three groups: those required for Achieved, Merit and Excellence respectively. Typically there are two to four criteria (or "bullet points") for each of these grades. To be given an Excellence grade, a student must meet all the criteria for the two lower grades as well, although with the external computer science assessment a "holistic" approach is taken, so if, for example, a student meets all criteria for all grades except one for "achieved," they might be given a "merit" grade based on their overall performance.

Exemplars of work are provided by NZQA as guidance to teachers on the kind of standard that is expected, although this is a double-edged sword as it can make teachers and students feel constrained to follow the exemplar closely, and at an extreme they may feel pressure use the exemplar as a template in which to substitute their own numbers or observations. This kind of work is not acceptable, but because the exemplars are publicly available on the NZQA website, markers need to be aware that students inevitably have access to them.

The internal assessment of programming means that the teacher can observe the process that a student uses to program, including how independently a student is working once they have learned to program. In fact, independence is required in the 2.45 and 2.46 standards in order for a student to be given a merit grade; if they require too much help from the teacher they may only get an achieved grade at best.

Due to various resource constraints and timetabling issues, assessment for the external computer science standards could not be done by tests or exams, and so they are assessed by submitting written reports or projects [Bell et al. 2012a]. For many topics, this is pedagogically desirable, and allows students to engage in more interesting and engaging learning tasks. It also has the useful side effect of encouraging students to develop their written communication skills. However, the report based assessment has placed much higher demands on teachers and those who were writing resources, since it requires a lot more guidance to help students produce work that genuinely reflects an understanding of computer science, and especially on personalising a report so that it is clear that the student has done independent work rather than just copied or paraphrased standard information. Furthermore, for some topics (particularly programming languages in the 1.44 standard), a test or exam would have been an easy means of assessment, so the need for a report has required a lot of creativity to develop appropriate tasks.

The issue of choosing suitable tasks for the external standards is examined in depth in a project that analysed the projects submitted by 151 students for the 1.44 standard in 2011 [Bell et al. 2012a]. The analysis of the student work found that teacher professional development and support was essential so that teachers could provide students with good projects. For example, for the algorithms topic in the 1.44 standard, students would find quite dramatic differences if they compare two contrasting algorithms such as binary search and linear search for large inputs, but 10% of the students had compared two $O(n^2)$ sorting algorithms (such as bubblesort and insertion sort), which have very similar performance. Some mistook the "cost" of an algorithm as being its number of lines of code (which was meaningless to plot for different values of n), and others took the broad interpretation that any program is an algorithm, and evaluated their own programs (such as a bouncing ball), which again were unlikely to have interesting asymptotic performance.

Personalisation of their work was another issue; running one's own experiment is not hard, and makes for an authentic report, but some simply chose to paraphrase authoritative sources, sometimes in a way that demonstrated that they did not under-

stand what they were writing. A related issue is that if the prompts from the teacher sound like an exam question, they are likely to encourage a terse exam-like response.

Generally students did not do so well when they were trying to do the evaluations on their own software. The computer science standards do not require that students write the software, and it is quite legitimate to evaluate existing systems. In fact, some students who implemented their own algorithms introduced bugs that prevented them from seeing the true behaviour (even binary search is notoriously hard to get right [Pattis 1988]), and students who evaluated interfaces that they had developed understood the interface too well to invoke confusing behaviour in it.

The length of the reports is limited to 14 pages; some teachers had complained that this is too long, although it is a maximum, and in the analysis of student work one student obtained an Excellence grade with under 7 pages of writing; the average Excellence report had about 10 pages of writing (not including padding such as a table of contents). Most students did not seem to be unduly constrained by the limit, and 91% of the submissions used 11 pages or fewer.

6. RESPONSE TO THE NEW STANDARDS

The new standards have increased awareness of computer science in New Zealand, and many students are now engaged with the subject who previously at best would have had only limited exposure to programming.

6.1. Related extracurricular activities

The creation of programming and computer science standards for schools has been accompanied by extracurricular activities, some of which are connected to or have become connected to the changes in schools.

One extra-curricular activity that existed before the standards came about is the "Programming Challenge for Girls"[25] (pc4g), an annual one-day programming challenge using the Alice language, for year 10 girls. This continues to be popular, and is now seen as an advertisement for classes teaching the new programming and computer science standards, since it is the year before students take NCEA. The event includes professional development time for teachers while the girls complete the challenge, and then a talk to the girls about computer science while their work is being marked. Finding time for professional development is difficult for teachers, and this model gives them some "free time" while their students are being occupied, which has been particularly valuable with the rapid introduction of the new standards.

The CodeAvengers project has resulted in "Code Camp" for students that is based around the resource, and this has provided another opportunity for students to engage with programming in schools.

A large New Zealand software company started a competition called "Codeworx"[26] based around the Raspberry Pi, to help enthuse students about the new opportunities. Information is provided on how projects could be used to achieve the new standards, so that students can compete for a prize and complete achievement standards at the same time.

At Canterbury University a "Computer Science Club"[27] started in 2012 for younger students who would like to learn the material taught in the standards, but were too young to do so. The club grew rapidly with very little advertising, and developed a program loosely based around the topics in the new standards, using a badge system

[25] http://www.pc4g.org.nz/
[26] http://codeworx.co.nz/
[27] http://computerscienceclub.org/

in a project led by Dr. Peter Denning and Susan Higgins.[28] The club is now being considered for use in some schools, including as a recruitment tool for engaging with students from "feeder" schools.

At the University of Otago a popular robotics club for school students has started in 2012, with an emphasis on programming for robotics. Robotics and programming projects have also been offered over many years in University outreach programs such as Hands-on Science, and the Otago University Advanced School Sciences Academy.

As well as the RoboCup Junior New Zealand competition noted above, various other programming competitions and related activities also occurred or are starting. These had been running prior to the changes, and some teachers are now beginning to use robotics with the new standards. As well as programming, they can cover the new electronics standards, as well as some of the generic technology standards.

6.2. Results, outcomes or consequences

Here we review the effect of the changes from the point of view of the various stakeholders in the changes: teachers, students, universities and industry. Overall there has been enthusiastic support and participation from champions in each of these groups, but the transition is a challenging time and has required considerable personal commitment from some. A particularly valuable outcome is that the different stakeholders have been engaged in a much higher level of communication with each other and their relationships have been strengthened: teachers have spent considerable time working with universities getting professional development and advice; universities have been more engaged with the details of what is being taught in the classroom; industry have sent staff to speak in schools to support teachers; university students have mentored both school students and teachers; universities have collaborated with industry to urgently address the need for resources to deliver the new standards; and education officials have engaged with all these groups to get advice. Many direct relationships have been formed between all these groups, which result in students getting more holistic guidance and a better sense of the opportunities available because their teachers are better connected.

6.2.1. Teachers. Teachers have emerged as the lynchpin in the success of the changes: they have considerable freedom to choose whether or not to offer a particular standard, they have been active in lobbying school management to introduce the new content, they are the ones who have to deliver the material at the "chalkface," much of the professional development has been delivered by teachers since they understand the context best, and the national association of teachers (NZACDITT) has had a crucial role, for which the work has necessarily fallen on the teachers themselves.

Understandably it has been a challenging and emotional time for many teachers: while many see the changes as an opportunity to improve their status, many have little or no training in the new topics, and this creates a lot of uncertainty and even a potential threat to them continuing in their job, as well as making them feel that what they have been teaching in the past has not been valued. Valuing teachers, most of whom have decades of teaching experience and yet can feel like outsiders, is an important component of the success of the changes [Ni and Guzdial 2012]. With the rapid introduction of the changes, support material for teachers was delivered late (particularly in the first year — for example, exemplars of student work were not available until halfway through 2011), and this meant teachers were taking on a major risk by adopting the new standards.

[28]http://cyberadventurers.org/

Professional development opportunities make a big difference to teachers' views of the subject. In the surveys, a teacher who had not attended any PD event commented "A pity that all that money was spent on a piecemeal range of standards that are far too academic for learners," whereas another remarked on how the PD had changed their view of the difficulty of the material: "Teaching Computer Science that appeared difficult on paper but was interesting and NOW I've had some PD (CS4HS) 2012 should be delivered better." Providing good professional development for teachers is strongly in the interests of both universities and industry, and partnership arrangements with the Ministry of Education have arisen from this motivation to address the urgent needs.

The transitional period is demanding on teachers, and for those who engaged with the new standards, considerable personal time and effort was required. In the 2013 survey eight of the comments noted that the transitional period was difficult ("Hopefully, the work load will ease as courses settle in"), but showed hope for the long term ("I am now into my third year with the [Year 11] students and feel really confident delivering the standards").

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."

Educating school management is an ongoing challenge, and is an important element for the success of computer science in schools.

6.2.2. Students. We have already observed that thousands of students have achieved the new standards (Table IV), and that numbers are increasing, particularly in the computer science standards (Section 4.2), and that these numbers are significant compared with the number of students who currently study computer science at university, although the first students from these courses will not arrive at university until 2014, and it may take a couple of years until the full impact can be measured.

Because courses adopting the new standards will be more academically oriented compared with the previous skills focussed courses, a major challenge for schools is to attract the right kinds of students to the new courses, particularly those heading for university qualifications, while still catering for weaker students who are after some basic computing skills. In larger schools this can be achieved by having a choice of courses, but in small schools one teacher may end up having to work with a wide variety of students in the same class. This was the topic that was mentioned the most in the comments in the 2013 survey. Of the 16 comments on this, 5 were positive ("For the first time in many years [I] have a Year 13 programming class with very able students"), but another 6 had attracted students who expected the older style of course about using computers ("not what the students expected, they are still thinking the old . . . courses that their brothers took"). The remainder had mixed experiences, and/or had to deal with classes that had a wide range of abilities.

The risk is that if the new nature of the course is not advertised properly, it may attract weak students who fail, class sizes then drop, and there is a risk of losing the teaching position, so the net effect is a decrease in computing being taught in that school. In the 2013 survey 53% of the teachers reported that a significant number of students who took the course lacked the academic ability to do it well, and 25% of teachers reported that the changes have led to significantly fewer students taking digital technologies. 40% reported that their course is attracting an increasing number of capable students, so while the changes appear to be successful in many schools, there are also schools where the message does not seem to be getting through to students.

6.2.3. Universities. An immediate result of the changes is that universities have become considerably more engaged with schools, both helping teachers with professional development, advising education officials on the new standards, and preparing and helping to deliver new materials in the classroom. As well as professional development events, university students have become involved in developing resources for schools and providing help for teachers [Bell and Lambert 2011], and universities are beginning to put formal teacher training courses into place to help them teach the new standards. One model that holds a lot of promise is to have carefully selected university students work in the classroom with teachers; this provides professional development for the teacher, experience for the university student, and a link to the university for the school students. Such arrangements could be made through university "work integrated learning" courses, so the university student also gets credit for the work.

In 2014 universities will need to adapt their entry level courses for the new students who will be coming through. Because the changes essentially bring students to the level of a first course in computer science, the focus will be on transitions through the first year of university, with some students being offered direct access to advanced classes, and the possibility of scholarships being based on the new achievement standards. One model that appears to be popular with students is for a university to offer a parallel first-year course that is worth the same credit as the traditional introductory course, but gives better prepared students more challenging projects to work on. Such students enjoy the opportunity for a challenge, they get to work with other advanced students, and the class is a good entry to have on their CV.

It is important to communicate to students that the new standards are valued by universities, particularly because in the past students have often been advised to take subjects like maths as preparation, and not computing classes. All eight New Zealand universities have combined to produce a jointly endorsed information brochure for students about the new standards and the transition to university[29]. Since the new standards are not available in all schools, universities will need to continue to provide an entry path for students who have not taken the subject in secondary school.

6.2.4. Industry. Industry support has been essential for helping to drive the change, and the process has provided increased communication between industry, schools and universities.

Fortunately the industry input has not focussed on wanting particular languages or skills taught, but primarily on growing the numbers of students interested in computer science and related subjects. The contribution from industry (and industry organisations) has included lobbying for the changes, contributing to decisions about the changes, funding much of the professional development and preparation of teaching resources (although run independently by universities without requirements to promote the use of particular software or products), and sponsoring competitions and events to motivate students to study computing.

Direct contact between teachers and industry has helped teachers realise the importance of their role and develop confidence for adopting the new standards. Industry organisations have also provided speakers and mentors to engage directly with students and encourage them to study computer science.

Industry (and industry organisations) have an important role to play communicating the importance of the new content to school management, students, and the public (parents and other influencers), since authentic voices offering career pathways are a key driver in having the changes adopted.

[29]Available from csanz.ac.nz

7. CONCLUSIONS

The introduction of computer science to New Zealand schools has occurred over a very short period and has been very demanding on those involved, but has brought about rapid deployment of much needed change.

Key observations from the process are:

Initiating change. The changes were brought about by reports from industry and teachers who understood the needs and were sensitive to the school environment [Grimsey and Phillipps 2008; Carrell et al. 2008]. The reports provided a thorough explanation of the current problems and suggested a constructive way forward; a similar report brought about change in the UK a few years later [Furber 2012]. The adoption process was essentially a grassroots movement, driven by teachers who had caught the vision and were empowered by the official changes to standards and support from industry and universities, but were primarily motivated because it would be better for their students, because of their personal interest in the topic, and because of the value to the country.

Window of opportunity. The political window of opportunity was seized, and those involved worked as quickly as possible to develop proposals, create resources, and support teachers. While this did mean that resources were not always available when needed, and that the content may not have been as polished as it should have been, it avoided losing the momentum for change.

Not too much content. The programming and computer science content represents only about half a course for each of the final three years of high school, and keeping the amount lower has been important to assist with widespread adoption, as it reduces teacher professional development needed, and makes it easier to include it as part of a course in related topics. Despite the huge pressures on resource developers and teachers alike caused by the rapid introduction, it still seems well worthwhile to have taken advantage of the short window of opportunity that occurred, even if it will take some time to iron out the problems and spread the content through a wider number of schools. This prevented the proposed changes getting bogged down in the development process, which could have ultimately resulted in no change. Early adopters were effectively pilot programmes, and are now in a position to share with colleagues what worked — and did not work — for them.

Publicity and media. Successful adoption means that students must appreciate that the new standards are available, that they are academically challenging, and that they are valued by universities and industry. A teacher may have caught the vision, but because students have a lot of flexibility in their course choices, teachers will need help educating those who have influence over student decisions about the nature and purpose of the changes, including school management, counsellors and parents. For this reason the media has an important role to play, and press releases from education officials, industry and universities can help to keep the changes visible for the public. Misinformation can persist for years, and so frequent accurate releases of information are important!

Ready-to-use resources for teachers. Teachers who are new to computer science have a lot to assimilate, and the more they are provided with resources that are ready to use and are pedagogically sound, the more likely they are to be able to deliver the new courses well, which is needed for a positive cycle of adoption. This means adapting resources intended for other contexts, and developing new resources, that match the requirements exactly. It is healthy that a small selection of resources has appeared so that there is some choice for teachers, but before they were available teachers indicated that they were overwhelmed by the variety of choices available, and having to construct a course from multiple disparate resources.

Industry support. Support from industry has been essential for resourcing teacher development, and it is impressive that most of the support has not been tied to particular products, but simply to increase the overall skill base in New Zealand. Having support from industry is also motivating for teachers, as it gives a clear message that their work is valued by those who would like to employ their students some years in the future.

Communication between stakeholders. The rapid changes have been made a lot easier because the various key stakeholders (industry, the secondary education system and universities) have opened good lines of communication, and been prepared to find compromises so that something could be deployed for the benefit of students, rather than having it mired in discussions that could prevent change from ever happening.

Curriculum level. Having programming and computer science start in the third to last year of high school is useful, but the pressure on students and teachers is likely to reduce as the new material filters down to lower levels, where it is not formally assessed, but provides a chance for students to become familiar with the language and culture of the subject.

In the process of introducing this new topic to New Zealand schools, much has been learned, the attitudes of many teachers and officials have changed, and universities and industry have become much more engaged with schools. Given the fast timeline, the lack of resourcing, and the impact of the Canterbury earthquakes, the uptake of the new standards has been remarkably high, and schools are now entering a period of consolidation where best practices are being shared and resources are being polished in the light of the experience during the transitional period.

APPENDIX

ELECTRONIC APPENDIX

The electronic appendix for this article can be accessed in the ACM Digital Library.

ACKNOWLEDGMENTS

The authors acknowledge the many New Zealand teachers who had the vision and courage to embrace the changes and make them work for their schools, and the leadership of the NZACDITT teachers' organisation. We particularly acknowledge the input we have received through discussions with the following teachers and officials: Alison MacDonald, Diana Beeby, Elizabeth Douglas, Gil Hunter, Jenny Baker, John Creighton, Julie McMahon, Karen Fahy, Kim Stephen, Linda McCavana, Margot Phillipps, Maurice Alford, Max Ross, Melinda Stevenson, Neil Leslie, Patrick Baker, Ron Elder, Tim Carrell, Vanja Venrooy, Vilna Gough-Jones, Cheryl Pym, Geoff Keith, Geoff Gibbs, Niall Dinning, Scott Telfer, Tony Turnock, and Vicki Compton.

There has also been a number of university staff, students and industry representatives who have caught the vision and played a key role, and we particularly acknowledge Ann McGrath, Caitlin Duncan, David Thompson, Gordon Grimsey, Heidi Newton, Ian McCrae, Ian Witten, Jack Morgan, Janina Voigt, John Hine, Linda Pettigrew, Lynn Lambert, Michael Trengrove, Michael Walmsley, Paul Matthews, Rhem Munroe, Robert Sheehan, Ross Peterson, Ross Young, Sally-Ann Williams, Samuel Williams, Sandy Garner, Stephen Corich, Stephanie Borgman, and Sumant Murugesh. We are grateful to David Thompson for assistance with the teacher surveys reported here.

REFERENCES

Owen Astrachan, Ralph Morelli, Dwight Barnette, and Jeff Gray. 2012. CS principles: piloting a national course. *Proceedings of the 43rd ACM technical symposium on Computer Science Education, Raleigh, NC, USA* (2012), 319–320.

Tim Bell, Peter Andreae, and Lynn Lambert. 2010. Computer Science in New Zealand High Schools. In *ACE '10: Proceedings of the 12th conference on Australasian Computing Education (Australian Computer*

Science Communications), Tony Clear and John Hamer (Eds.), Vol. 32. Australian Computer Society, Inc., Brisbane, Australia, 15–22.

Tim Bell, Peter Andreae, and Anthony Robins. 2012. Computer Science in NZ High Schools: The First Year of the New Standards. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education, Raleigh, NC, USA*, Laurie A. Smith King, David R. Musicant, Tracy Camp, and Paul Tymann (Eds.). ACM, New York, 343–348.

Tim Bell, Gwenda Bensemann, and Ian H Witten. 1995. Computer Science Unplugged: capturing the interest of the uninterested. In *Proceedings of the 14th NZ Computer Conference*. Wellington, New Zealand.

Tim Bell and Lynn Lambert. 2011. Teaching Computer Science Majors About Teaching Computer Science. In *Proceedings of the 42nd ACM technical symposium on Computer Science Education, Dallas, TX, USA (SIGCSE '11)*. ACM, New York, NY, USA, 541–546.

Tim Bell, Heidi Newton, Peter Andreae, and Anthony Robins. 2012a. The introduction of computer science to NZ high schools: an analysis of student work. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education (WiPSCE '12)*. ACM, New York, NY, USA, 5–15. DOI:http://dx.doi.org/10.1145/2481449.2481454

Tim Bell, Frances Rosamond, and Nancy Casey. 2012b. Computer Science Unplugged and related projects in math and computer science popularization. In *The Multivariate Algorithmic Revolution and Beyond: Essays Dedicated to Michael R. Fellows on the Occasion of His 60th Birthday*, Hans L Bodlaender, Rod Downey, Fedor V Fomin, and Daniel Marx (Eds.), Vol. LNCS 7370. Springer-Verlag, Berlin, Heidelberg, Heidelberg, 398–456. DOI:http://dx.doi.org/citation.cfm?id=2344236

Lenore Blum and Thomas J Cortina. 2007. CS4HS: an outreach program for high school CS teachers.. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education, SIGCSE 2007, Covington, Kentucky, USA*, Ingrid Russell, Susan M Haller, J D Dougherty, and Susan H Rodger (Eds.). ACM, 19–23. http://dblp.uni-trier.de/db/conf/sigcse/sigcse2007.html\#BlumC07

Heather Bort and Dennis Brylow. 2013. CS4Impact: measuring computational thinking concepts present in CS4HS participant lesson plans. In *Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13)*. ACM, New York, NY, USA, 427–432. DOI:http://dx.doi.org/10.1145/2445196.2445323

M E Brown. 1998. The Use of Computers in New Zealand schools: A critical review. *Computers in New Zealand Schools* 10, 3 (1998), 3–9.

Tim Carrell, Vilna Gough-Jones, and Karen Fahy. 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*. Technical Report. http://dtg.tki.org.nz/content/download/670/3222/file/DigitalTechnologiesdiscussionpaper.pdf

Lillian Cassel, Alan Clements, Gordon Davies, Mark Guzdial, Renée McCauley, Andrew McGettrick, Bob Sloan, Larry Snyder, Paul Tymann, and B Weide. 2008. Computer science curriculum 2008: An interim revision of CS 2001. *Report from the interim review task force* (2008).

Tony Clear and Graham Bidois. 2005. Technology Technology FITNZ : An ICT Curriculum Meta-Framework for New Zealand High Schools. *Bulletin of Applied Computing and Information Technology* 3, 3 (2005). http://www.naccq.ac.nz/bacit/0303/2005Clear_FITNZ.htm

Vicki Compton and Bev France. 2007. Towards a new technological literacy: Curriculum development with a difference. *Curriculum Matters* 3, 1 (2007).

Vicki Compton and Cliff Harwood. 2003. Enhancing technological practice: An assessment framework for technology education in New Zealand. *International Journal of Technology and Design Education* 13, 1 (2003), 1–26.

Vicki Compton and Alister Jones. 1998. Reflecting on teacher development in technology education: Implications for future programmes. *International Journal of Technology and Design Education* 8, 2 (1998), 151–166.

Peter J Denning and Andrew McGettrick. 2005. Recentering computer science. *Commun. ACM* 48, 11 (Nov. 2005), 15–19. DOI:http://dx.doi.org/10.1145/1096000.1096018

Steve Furber (Ed.). 2012. *Shut down or restart? The way forward for computing in UK schools*. The Royal Society, London. http://royalsociety.org/education/policy/computing-in-schools/report/

Judith Gal-Ezer, Orit Hazzan, and Noa Ragonis. 2009. Preparation of high school computer science teachers. *ACM SIGCSE Bulletin* 41, 1 (March 2009), 269. DOI:http://dx.doi.org/10.1145/1539024.1508965

Sandy Garner, Patricia Haden, and Anthony Robins. 2005. My program is correct but it doesn't run: a preliminary investigation of novice programmers' problems. In *Proceedings of the 7th Australasian conference on Computing education-Volume 42*. Australian Computer Society, Inc., 173–180.

Joanna Goode and Jane Margolis. 2011. Exploring Computer Science. *ACM Transactions on Computing Education* 11, 2 (July 2011), 1–16. DOI:http://dx.doi.org/10.1145/1993069.1993076

Gordon Grimsey and Margot Phillipps. 2008. *Evaluation of technology achievement standards for use in New Zealand secondary school computing education*. Technical Report. New Zealand Computer Society (NZCS), Wellington. http://www.nzcs.org.nz/news/uploads/PDFs/200805NCEAReport.pdf

Peter Hubwieser, Michal Armoni, Torsten Brinda, Valentina Dagiene, Ira Diethelm, Michail N Giannakos, Maria Knobelsdorf, Johannes Magenheim, Roland Mittermeir, and Sigrid Schubert. 2011. Computer science/informatics in secondary education. In *Proceedings of the 16th annual conference reports on Innovation and technology in computer science education-working group reports*. ACM, 19–38.

Neal Koblitz. 1996. The case against computers in K-13 math education (Kindergarten through calculus). *The Mathematical Intelligencer* 18, 1 (1996), 9–16. DOI:http://dx.doi.org/10.1007/BF03024811

Orni Meerbaum-Salant, Michal Armoni, and Mordechai Ben-Ari. 2011. Habits of programming in scratch. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education (ITiCSE '11)*. ACM, New York, NY, USA, 168–172. DOI:http://dx.doi.org/10.1145/1999747.1999796

Sumant Murugesh, Tim Bell, and Ann McGrath. 2010. A Review of Computer Science Resources to Support NCEA. In *First annual conference of Computing and Information Technology Research and Education NZ (CITRENZ2010)*, Samuel Mann and Michael Veerhaart (Eds.). 173–181.

Lijun Ni and Mark Guzdial. 2012. Who AM I? Understanding High School Computer Science Teachers' Professional Identity. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education, Raleigh, NC, USA*. 499–504.

Richard E. Pattis. 1988. Textbook errors in binary searching. *ACM SIGCSE Bulletin* 20, 1 (Feb. 1988), 190–194. DOI:http://dx.doi.org/10.1145/52965.53012

Noa Ragonis, Orit Hazzan, and Judith Gal-Ezer. 2010. A survey of computer science teacher preparation programs in Israel tells us: computer science deserves a designated high school teacher preparation!. In *Proceedings of the 41st ACM technical symposium on Computer science education (SIGCSE '10)*. ACM, New York, NY, USA, 401–405. DOI:http://dx.doi.org/10.1145/1734263.1734402

Anthony Robins. 2010. Learning edge momentum: A new account of outcomes. *Computer Science Education* 20 (2010), 37 – 71. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.167.651

Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: A review and discussion. *Computer Science Education* 13, 2 (2003), 137–172.

Nathan Rountree, Janet Rountree, Anthony Robins, and Robert Hannah. 2004. Interacting factors that predict success and failure in a CS1 course. In *ACM SIGCSE Bulletin*, Vol. 36. ACM, 101–104.

Mehran Sahami, Steve Roach, Ernesto Cuadros-Vargas, and Richard LeBlanc. 2013. {ACM/IEEE-CS} computer science curriculum 2013: reviewing the {I}ronman report. In *Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13)*. ACM, New York, NY, USA, 13–14. DOI:http://dx.doi.org/10.1145/2445196.2445206

Philip Sallis, D Ferguson, A Frampton, V Ham, A Milne, T McMahon, L Parker, N Parker, and V Ramsay. 1990. Report of the Consultative Committee on Information Technology in the school curriculum. (1990).

Val Savidan. 2003. ICT and the New Zealand secondary school curriculum. *ACE Papers* 12 (2003), 123–144.

David Thompson and Tim Bell. 2013. Adoption of new Computer Science high school standards by New Zealand teachers. In *The 8th Workshop in Primary and Secondary Computing Education (WiPSCE 2013)*, Maria Knobelsdorf, Ralf Romeike, and Michael E. Caspersen (Eds.). ACM, Aarhus, Denmark. http://www.iitp.org.nz/files/wipsce-teachers-2013.pdf

David Thompson, Tim Bell, Peter Andreae, and Anthony Robins. 2013. The role of teachers in implementing curriculum changes. In *Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13)*. ACM, Denver, CO, 245–250.

Allen B Tucker. 2010. K-12 Computer science: Aspirations, realities, and challenges. In *Lecture Notes in Computer Science*, Juraj Hromkovic, Richard Kralovic, and Jan Vahrenhold (Eds.). Springer, Chapter Teaching F, 22–34.

Charles R Woratschek and Terri L Lenox. 2009. Student Attitudes and Perceptions Regarding Computing and Its Related Disciplines. *Information Systems Education Journal* 7, 58 (2009), 3–22.

SeungWook Yoo, YongChul Yeum, Yong Kim, SeungEun Cha, JongHye Kim, HyeSun Jang, SookKyong Choi, HwanCheol Lee, DaiYoung Kwon, HeeSeop Han, EunMi Shin, JaeShin Song, JongEun Park, and WonGyu Lee. 2006. Development of an Integrated Informatics Curriculum for K-12 in Korea. In *Informatics Education — The Bridge between Using and Understanding Computers*, Roland Mittermeir (Ed.). Lecture Notes in Computer Science, Vol. 4226. Springer Berlin / Heidelberg, 199–208. DOI:http://dx.doi.org/10.1007/11915355-19

Received April 2013; revised July 2013; accepted November 2013

Online Appendix to: A case study of the Introduction of Computer Science in NZ schools[30]

TIM BELL, University of Canterbury
PETER ANDREAE, Victoria University of Wellington
ANTHONY ROBINS, University of Otago

A. GLOSSARY AND WEB LINKS

This online appendix provides a glossary and web links to some of the key terminology specific to CS education in New Zealand.

Achievement standard. An assessment standard that is given a grade of Not Achieved, Achieved, Merit, or Excellence (compared with a *unit* standard which is a pass/fail assessment). Several achievement standards are combined to assess a course. See http://www.nzqa.govt.nz/qualifications-standards/standards/

AS91074. (Also known as 1.44) Level 1 Computer Science achievement standard: Demonstrate understanding of basic concepts from computer science. http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2011/as91074.pdf

AS91075. (Also known as 1.45) Level 1 Program design achievement standard: Construct a plan for a basic computer program for a specified task. http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2012/as91075.pdf

AS91076. (Also known as 1.46) Level 1 Program implementation achievement standard: Construct a plan for a basic computer program for a specified task. http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2012/as91076.pdf

AS91371. (Also known as 2.44) Level 2 Computer Science achievement standard: Demonstrate understanding of advanced concepts from computer science. http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2012/as91371.pdf

AS91372. (Also known as 2.45) Level 2 Program design achievement standard: Construct a plan for an advanced computer program for a specified task. http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2012/as91372.pdf

AS91373. (Also known as 2.46) Level 2 Program implementation achievement standard: Construct an advanced computer program for a specified task. http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2012/as91373.pdf

AS91636. (Also known as 3.44) Level 3 Computer Science achievement standard: Demonstrate understanding of areas of computer science. http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2013/as91636.pdf

AS91637. (Also known as 3.46) Level 2 Programming achievement standard: Develop a complex computer program for a specified task. http://www.nzqa.govt.nz/nqfdocs/ncea-resource/achievements/2013/as91637.pdf

CodeAvengers. Interactive online courses that teach the basics of web development and computer programming (HTML5, CSS3, JavaScript) http://www.codeavengers.com/

CS4HS. Computer Science for High School. A Google sponsored event for high school teachers, run in many countries, but used in New Zealand to provide urgently needed professional development for teachers. The event has been run annually starting in December 2011.

DOI: http://dx.doi.org/10.1145/0000000.0000000

CS Field Guide. An on-line, open-source, free resource developed initially to meet the urgent needs of teaching the new New Zealand achievement standards in computer science. The student version is available at http://csfieldguide.org.nz/, and a teacher version is available on request.
Digital Technologies standards. The term given to the grouping of new standards for teaching computing in NZ schools; there are five general areas: Computer science and programming, Digital information, Digital media, Digital infrastructure and Electronics. http://dtg.tki.org.nz/
DTEP. Digital Technology Experts Panel. A committee representing stakeholders formed in November 2008 to advise on Digital Technologies in schools, including suggesting how learning could be structured and to make recommendations on assessment standards. The report from the panel is available at http://technology.tki.org.nz/Curriculum-support/Knowledge-and-Skills-documents
DTG. Digital Technologies Guidelines, a NZ Ministry of Education funded project run from 2007 to 2010 to revise the computing curriculum in schools, http://dtg.tki.org.nz/
FITNZ project. Fluency in IT project, a 2006 review of computing in NZ education, aligning it with the needs of industry and tertiaries [Clear and Bidois 2005]. This lead to the DTG project.
IITP. (Institute of IT Professionals, previously known as the New Zealand Computer Society, NZCS. A national organisation that "works with government, the education sector and academia, the industry, the IT community and the public at large to increase the education, professionalism and expertise of those working in the ICT sector and advance education across the board in the interests of New Zealand." http://www.iitp.org.nz/
IPENZ. Institution of Professional Engineers New Zealand. "The professional body which represents professional engineers from all disciplines in New Zealand." www.ipenz.org.nz
Ministry of Education. "The Government's lead advisor on the New Zealand education system, shaping direction for sector agencies and providers." http://www.minedu.govt.nz/
New Zealand Curriculum. Official documents that "set the direction for student learning and provide guidance for schools as they design and review their curriculum." http://nzcurriculum.tki.org.nz/Curriculum-documents
NCEA. National Certificate in Educational Achievement, the major school-leaving qualification in NZ usually completed in the last three years of school. http://www.nzqa.govt.nz/qualifications-standards/qualifications/ncea/
NZACDITT. New Zealand Association for Computing, Digital and Information Technology Teachers. "An association with the goal of advocating for our subjects. The aim of the association is to create a community of teachers where we can share resources, communicate and speak with one voice to get our subject area recognised and supported." The NZACDITT maintains a forum for members, and a web site that includes resource guides and shared resources. http://nzacditt.org.nz
NZCS. The name of the IITP before July 2012.
NZQA. The New Zealand Qualifications Authority, a government agency that manages school and other non-university qualifications in NZ. http://www.nzqa.govt.nz/
Otago programming books. A series of four books, one for the level 2 and 3 programming achievement standards, in Java and Python. http://www.cs.otago.ac.nz/schools/
Technology. In the NZ curriculum this is defined as "intervention by design to expand human possibilities" http://seniorsecondary.tki.org.nz/Technology

TKI. Te Kete Ipurangi ("the online knowledge basket"), the main online portal to teaching information and resources from the Ministry of Education. http://www.tki.org.nz/

Unit standard. A pass/fail assessment standard, usually developed by an industry training organisation. See http://www.nzqa.govt.nz/qualifications-standards/standards/

Year. This is the term equivalent to "grades" in the US; students start in Year 1 when they turn 5 years old, and the final year of school is Year 13 (when students are typically around 17 or 18 years old).