Teaching Computer Science Majors About Teaching Computer Science

Tim Bell
Department of Computer Science and Software Engineering
University of Canterbury
New Zealand
+64 3 363-2987
tim.bell@canterbury.ac.nz

Lynn Lambert
Physics, Computer Science and Engineering
Christopher Newport University
Newport News, VA
+1 757 594-7826
llambert@cnu.edu

ABSTRACT
This paper describes the design, implementation, and evaluation of a course teaching Computer Science majors about teaching Computer Science. The course was designed to address the need for teachers and resources to support rapid changes in topics being taught in high schools. It also helped prepare students for research in Computer Science Education, and for careers involving computing and education. The course is described in detail, and is evaluated based on student feedback and the outcomes from the course.

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

General Terms
Design, Economics.

Keywords
Computer Science Education, K-12.

1. INTRODUCTION
A number of countries currently have initiatives to improve K-12 Computer Science education. For such initiatives to be successful, there needs to be a good supply of people able to teach Computer Science in K-12 schools. At the same time, we need to develop resources suitable for use in the K-12 environment, since the new curricula being developed don't have a canon of textbooks and teaching resources that more traditional subjects have to build on.

SIGCSE'11, March 9–12, 2011, Dallas, Texas, USA.

This paper reports on a course for CS major students that addresses both the issue of preparing teachers and the challenge of developing new resources. The course covers CS education at all levels, but focused particularly on high school as this is the area with the greatest change in New Zealand at present.

Table 1 shows four sources of new K-12 CS teachers based on their existing background. Each of the groups have relative advantages in terms of availability and preparedness, and all four groups should be considered for growing the teacher population. Teachers in New Zealand come through all four of these paths, although relatively few of the current computing teachers have a formal CS background. This paper discusses the design, implementation, and evaluation of a course that focuses on a group of students in the top right quadrant: existing senior CS students.

Table 1: Sources of potential K-12 Computer Science teachers

	Qualified to teach	**Not qualified to teach**
Has CS background	Ready to teach CS courses	Need to learn about teaching
No CS background	Need professional development in CS	Need both teaching and CS

We developed and taught a course on Computer Science education for honors students in their fourth year of a Computer Science major. The class does not qualify them to teach, but it gives them a taste of teaching and enables them to explore issues in Computer Science education, including the very education they are receiving at the time!

There are several reasons for a student to take such a class, even if they are not preparing to be a teacher:

- it helps them to understand their own education by understanding the pedagogical theory and CS-related issues that their own professors must deal with,
- they are able to create educational resources that can be used by qualified teachers,

- during a career in industry they may be involved in outreach and mentoring to schools or provide policy input to educational organizations, and it provides a good base for this,
- it may be a stepping stone for some to become teachers or professors, and
- it provides a foundation if they are interested in becoming CS Education researchers.

The course outline explained the course to students as follows:

"Computer Science Education has received a lot of attention recently, as governments seek to boost the competitiveness of their country in the face of falling rolls in universities. A lot of innovation and research over the last few years has changed the kind of resources that are available, and there are a rich range of materials available (e.g. alice.org, cs4fn.org, csta.acm.org). There has also been a lot of debate about related issues such as which programming languages are best for beginners, and how best to get more women involved in CS.

In New Zealand, the computing topics taught in schools have been revised in 2009, with the major changes appearing in NCEA [the national qualification taken by most students in their final three years of high school] from 2011. This has raised many new research questions about which topics can and should be taught at high school, and the best way to deliver them.

In this course we will look at the many issues surrounding CS education. There will be a major project where students will develop new educational materials and/or evaluate them. As much as possible, the project will be done in conjunction with students from a computing class at the College of Education, who will be able to provide a stronger educational background to complement the CS background of our own students.

The course is aimed at people who may be interested in being CS educators (teachers or lecturers), as well as participating in outreach programs (such as industry-sponsored events and school visits), and those who may be involved in educational research and policy. It will assume that you have a broad 3-year background in Computer Science, but no previous background in education is required."

A course focused on CS education, rather than education in general, is able to tackle the many topics specific to CS education, such as the surges and declines in student interest, gender imbalance, and the choice of a first programming language. We found that there was no shortage of topics to include in the course!

The remainder of this paper gives more detail about the goals of the course (section 2), its structure and assessment (section 3), and a required student project to develop educational resources (section 4). We evaluate the course in section 5, and draw some conclusions in section 6.

2. GOALS OF THE COURSE

The stated goal of the course was to produce students who are well-informed about the wide-ranging issues surrounding Computer Science education, are able to provide balanced and sound advice to existing computing teachers who have a weak background in Computer Science, and are able to develop and use effective resources themselves.

Because a key motivation for this course was to address the issues raised by new topics being introduced into senior high school computing classes in New Zealand [3], the course had a focus on these issues and students were encouraged to develop resources and engage with teachers from this sector. The new topics for high schools, to be introduced in NZ in 2011, contain an optional component called "Programming and Computer Science" that gives senior students an introduction to programming as well as an overview of the breadth of topics in Computer Science. The goal of these topics is to spark interest in the field in the face of low student enrolments in Computer Science. For new courses like this to be successful, NZ teachers will need more resources and professional development (PD) in Computer Science to deliver the new material.

The United States has similar concerns with its high school curriculum. Many would like to modify how Computer Science is taught in high schools, to increase interest in Computer Science, to improve diversity, and to give a more accurate picture of Computer Science. There has been recent focus on the Advanced Placement Computer Science course. The AP CS exam, one measure of high school Computer Science enrolment, has seen a 2% decline in the number of students taking the test from 2000 to 2010, while the number of total number of students taking AP exams has risen by 61% [2]. CRA enrolment tracking in colleges and universities [4] reflects similar trends. An important component of addressing the high school pipeline is to change how colleges train Computer Science teachers, and to improve the resources that Computer Science teachers have. Some are attempting to create a new AP course more focused on Computer Science concepts and less on programming. One goal of this new course would be to increase participation in Computer Science, especially among women. In order to do that, the US will need to recruit more teachers [6]. Computing Education for Computer Science majors would help provide both new teachers and resources for these new courses.

There have been some successes revising high school curricula and preparing teachers to teach Computer Science in the high schools [7, 8]. These have made a point of providing teachers with (a) good professional development and (b) resources for them to use. Teaching a course about teaching Computer Science to CS majors is a less direct approach, but is more easily implemented by CS departments. It may produce a few teachers who are well-prepared to teach Computer Science and may create some new Computer Science education researchers, but importantly, it will also contribute to educational resources, will help students understand their own education, and will provide education-aware professionals who can interact with students and teachers effectively. This means that those who end up working in commercial companies or some other career outside of K-12 education are better prepared to engage in school outreach and support, ultimately improving the relationship between industry and schools.

3. COURSE STRUCTURE AND ASSESSMENT

The course was run over one semester (15 weeks including a three-week break from lectures) in the Computer Science and Software Engineering department at the University of Canterbury. The topics covered were based around the following formal outline, although we also explored topics of interest to students as

discussions arose in class.

- Curricula (school and university, New Zealand and international, CSTA/ACM guidelines, ISTE NETS, AP exams)
- Pedagogy (learning models e.g. student centered, instructional design, high-order thinking, Bloom's taxonomy, learning styles; socio-cultural theory e.g. Vygotsky, constructivist approach, Piaget; teaching and learning tools e.g. Learning Management Systems, distance learning, e-learning)
- K-12 Computer Science education and outreach (curricula, programming languages, robotics, kinesthetics, camps, clubs and events)
- Choosing a first programming language (issues such as OO-first, Initial Learning Environments)
- Programming environments for children/learners (Scratch, Alice, Greenfoot)
- Visualization and algorithm animation
- Gender, diversity, disability, and equity issues
- Educational games for CS
- Robotics for K-12

The course was taught by two instructors and several guest lecturers. Three of the guests were from our College of Education, but we also had one school teacher and two CS Education research students speak to the class. Student assessment consisted of a final exam (worth 40% of the final grade) that covered the content of these lectures, and a substantial project (60%), to create a resource for high school Computer Science teachers.

The course enrolled 10 Computer Science majors who had essentially no background studying education, but who were strong in Computer Science (they already had a 3-year Bachelor's degree, and this was a postgraduate class). This was a typical class size at this level. Some of the students had a passion for education and improving the state of K-12 education in NZ, motivated by the lack of CS courses available when they went through high school — generally high school courses in NZ have been focused on user skills, and students in our class were enthusiastic about the discipline of computing, not simply using computers as a tool. Although such students have been part of an educational system for many years, this does not mean that they understand pedagogy, and depending on exactly where they were taught, their model for pedagogy might be based on weak examples that they instinctively emulate, so there was also some "un-learning" to be done.

The class met for two hours of lectures each week, for about 10 weeks of the semester. A lot of the course revolved around the major project, which is described in the next section. Early in the course we had a small online quiz reviewing the lecture topic each week, and students volunteered to write questions. Writing questions for such a quiz is a great exercise for students, and their peers could give them feedback on the quality of the quiz questions. However, their motivation to develop quizzes petered out, apparently as the project became more important later in the course.

Some of the lecture topics were easy for the students to pick up (e.g. introductory programming languages), while other topics were unfamiliar. Few of the students were aware of the various curricula despite their own courses being set in the ACM/IEEE curriculum frameworks. The material on pedagogy was also largely unfamiliar, and ideas like higher-order thinking and constructivist teaching helped them to understand some of the techniques used in their own education. Where possible, examples were given to relate these ideas to Computer Science (for example, examples of exercises from various levels of Bloom's taxonomy).

4. STUDENT PROJECTS

A central part of the course was that students were required to work on a major project to develop an educational resource. This had the dual benefit of having them apply the pedagogical ideas they had learnt, and also generated new resources for CS education. We encouraged students to focus on high school level material, since there was an immediate need for new resources in this area. The "resources" could be anything that could be deployed by an educator, for example:

- lesson plans (including guidance for teachers and sample assessment)
- programming exercises that use CS concepts (such as Scratch, Alice and Greenfoot activities that involve more than animation and games)
- an interactive website, game or visualization
- a set of classroom posters
- creative work such as videos or cartoons that convey CS concepts
- kinesthetic activities or role plays that teach concepts
- material for teaching teachers
- multimedia presentations on topics using systems such as Flash or PowerPoint, with detailed notes that teachers can use in class for introducing topics

The students were given specific examples of a range of existing quality resources to illustrate the variety of approaches that could be taken.

The requirements for the project were reasonably flexible, although each student was expected to have the following structure in their report:

- A choice of a topic from CS that they would create the resource for;
- An evaluation of existing approaches (if any) for teaching that topic (and their strengths and weaknesses for high school teaching);
- An explanation of the target audience and the design choices for the resource (in terms of pedagogical theory), including choices that were considered and rejected;
- An evaluation of the resource – they needed to provide evidence that it works, or at least give ideas on how to improve it based on experience with it. The resource could be evaluated by trying it on students (and since it probably contains feedback/assessment material, that could be used to evaluate what they learned.) The resource could also be evaluated by showing it to teachers or other experts, and asking them questions about how they could use it. On-line resources could have automated feedback, such as comments on YouTube, the number of hits, time spent on each page, or tracking the parts of the resource that people used.

We suggested a list of topics from the new NZ school curriculum for which there was a lack of resources and so would particularly benefit from resources being created, and about a half of the projects produced material that was related to these ideas. The

other half were more focused on a student's interest or perception of what they would have liked more help with when they studied the topic.

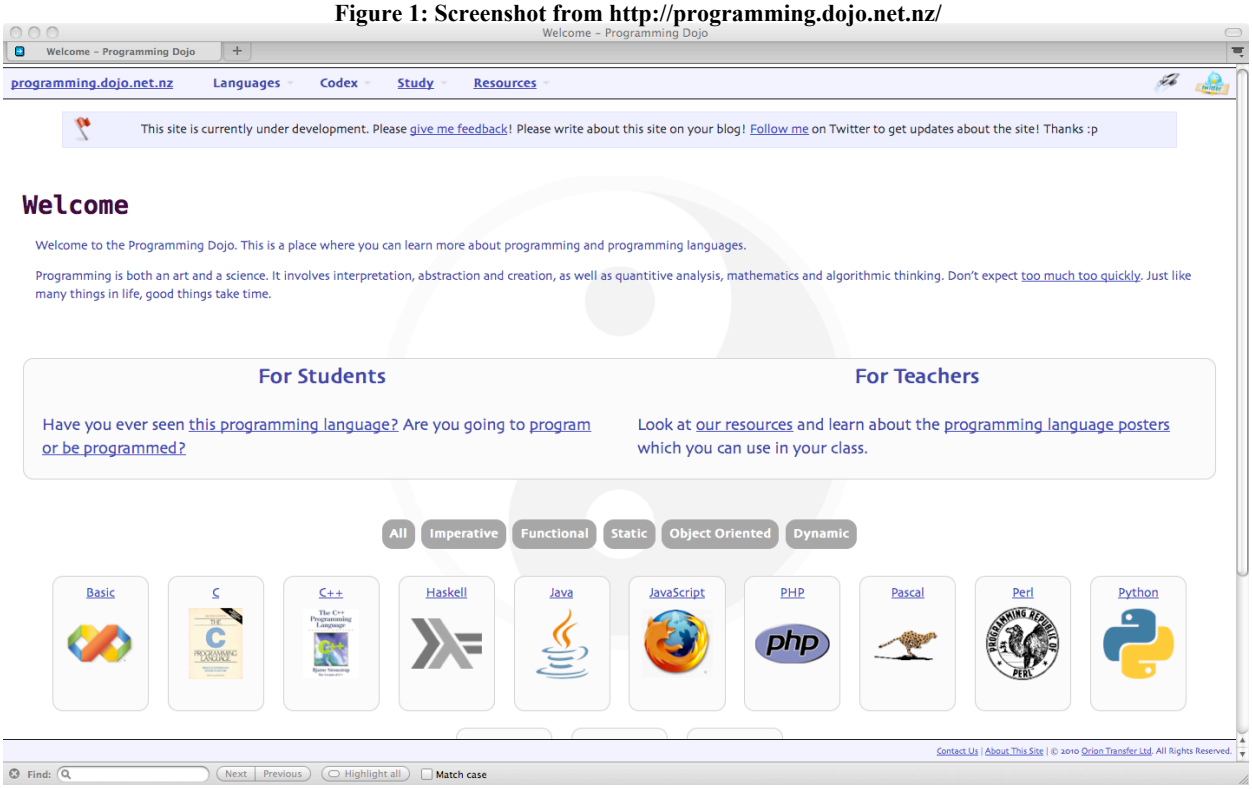
Students were encouraged to look through CS education research papers from organizations such as SIGCSE and ITiCSE for relevant background material, and as models of what CS Education researchers do. They were expected to look for existing material that taught their topic from places like text books, on-line CS activity collections, videos (e.g. on YouTube), online games, Wikipedia entries, online tutorials, and papers and ideas from SIGCSE and ITiCSE. They had to critically review these existing resources for their topic and explain why their one was likely to be an improvement.

The students regularly shared their projects with each other through small class presentations and discussions, and posting material online. This enabled them to exercise the critical view of educational material that we wanted them to develop, and help them get constructive feedback for their projects from their peers and lecturers. Towards the end of the course we arranged a meeting at a local school where students could present their projects to teachers, who provided the feedback that was required for their projects. Four of the 10 students took this opportunity; others received feedback from online communities, or friends who were teachers.

Here is a sample of the projects that the students chose:

- "Programming Dojo" (http://programming.dojo.net.nz/). This web site, created by one of the students (and later blogged by CSTA [5]), contains an extensive overview of programming languages from an educational point of view (some of the home page is shown in Figure 1; not visible in the screenshot are links for Ruby, Scheme and Smalltalk). The site has taken on a life of its own beyond the course; for example C# and Scratch are being added due to interest from teachers. The site also reviews the strengths of different languages for teaching programming by answering 16 questions about each one, such as "Is there a compiler/interpreter available at no cost?", "Are people in education already familiar with the language and environment?", and "Are there good learning resources available for use in education?" It also contains downloadable posters showing the same problem being programmed in many different languages. This website has received considerable attention from teachers. For the evaluation the student posted the site to various programming language forums, and received a very strong response as people defended the appropriateness of their preferred language for teaching programming, and provided answers for the questions used to evaluate the languages! The site itself had over 6000 hits in the first two weeks, and the posters were downloaded hundreds of times. Needless to say, this was considered a very successful project, and exceeded expectations.
- The Traveling Salesman problem with Google maps. This on-line system was aimed at illustrating tractability, which is a proposed topic for Year 12 students (second to last year of high school in NZ). It had information about the TSP and its associated computational issues, and then showed various TSP problems being solved on a page where students could interact with a Google map; the impossibility of trying all possible routes for large numbers of cities becomes very obvious as students have to wait for exponentially increasing amounts of time to get a result. The resource included notes for teachers.
- Posters promoting Computer Science. This student researched and designed four posters. Three of them featured Turing, Hopper and Dijkstra respectively, and showed a mixture of both humorous and thought-provoking quotes from those people, as well as giving a short but intriguing biography. The fourth poster illustrates how ray tracing works.

Figure 1: Screenshot from http://programming.dojo.net.nz/



- Formal languages. This project was essentially an outline of a lesson plan that enabled teachers who know little about regular expressions or grammars to teach some simple concepts based on games and puzzles. It drew on several existing published resources, and added some new material. The plan was aimed at addressing a new assessment standard in the New Zealand system where students were required to appreciate the nature of formal languages.
- "Java arrays for kids". This resource was aimed at younger students (despite the title, the resource is reasonably language-independent). It provides some off-line kinesthetic activity ideas based on placing items in small plastic containers, and an on-line game that culminated in trying to sort an "array" of numbers where the only operation possible is to move numbers between two chosen locations in the array.

5. RESULTS

The students were surveyed at the end of the course. Exact statistics for the responses cannot be published to protect the privacy of the students, but we note the following trends in the responses.

Students felt strongly that they had a broad view of Computer Science education around the world after taking the course, and that it helped them think about Computer Science in new ways. They were also positive about having a better understanding of Computer Science after taking this course. Participation in the course made some stronger advocates for Computer Science, especially how it is taught in secondary schools. Some became more interested in teaching high school and/or university Computer Science after taking the course.

Positive comments included "The paper is great. It is creative and useful." Negative comments included "too much stuff to read," "would have been great to work with education students" (the planned collaboration with an education class fell through due to unavailability of the education students, although two education students ended up becoming involved after our course had completed), and several comments about wanting more time and guidance on their projects.

Some of the difficulties the students encountered were a result of the inevitable differences in culture between Education and Computer Science. The Education lecturers did provide a lot of reading material which CS majors aren't so used to dealing with. This could be improved by focusing the material on directly relevant material, as the key material covered only made up a small part of the reading that was handed out.

In general, the two primary lecturers were satisfied with the course, the projects, and the students. The project was loosely structured, which allowed creativity, but also made it difficult for students to narrow their focus, and it required them to shift gear from the large programming projects typically required in senior courses, to complete a working project that relied on external input in only one semester.

The instructors felt that there were several factors that influenced the success of the students. Students from other cultures who were used to different pedagogical styles, and those whose own education included a great deal of rote learning and "sage on the stage" style teaching had a difficult time developing interactive projects that would require higher-order thinking rather than rote memorization.

The multiple guest lecturers from around the university and local schools meant that students heard from experts in multiple areas, including education. That would not have been possible if taught solely by Computer Science professors. It did mean that there was less control in terms of content, delivery and interest to the students, and some of the reading on educational theory was a different style than the students were used to. Having guest professors had the benefit of providing another point of dialog between educationalists and Computer Scientists, and we reciprocated with our College of Education by giving guest lectures in some education classes. Such interdisciplinary relationships can take a lot of effort to develop, and having a common interest in this class gave a concrete goal to work on together.

One of the challenges that came up was that students with web-based projects needed to place it in a publicly visible on-line location so that they could get feedback on it. It was difficult to find a host site that would be suitable if the material became very successful, but not too embarrassing if the material wasn't so good (and since it was initially posted for early evaluation, it needed to be clear that it wasn't a university-endorsed final product). Some of the resources needed PHP and related backend support for their web pages, so it wasn't just a matter of finding somewhere to put static pages. In the end, a variety of sites were used depending on the type of resource and intended audience, but this had the unfortunately consequence that most of the projects are no longer easily accessed online.

The requirement for our students to evaluate their resources was valuable, but for them to get direct feedback from school students would require ethics approval, which typically takes several weeks to obtain. Because the ethics application requires the "experiment" to be fully designed, this made it effectively impossible for students to undertake such evaluations in the timeframe of the course, and thus the feedback had to be based on the opinion of professionals (teachers) and people making public posts on-line. A longer timeframe for the assessment would be required if school students were to be used in the evaluation.

6. CONCLUSION

Teaching a course on CS education to advanced CS students has a number of benefits as well as challenges. The benefits included having students who could work with depth and breadth in CS topics, and produce new resources that can be used for younger students. The students were more informed about pedagogy, and especially issues surrounding CS education.

The main difficulties revolved around students being used to more technical courses, and also not having a strong background in education — possibly not even having had a good experience in their own education.

Next time we would try to find a way to link this group directly with a class of education students, and ideally run joint projects to combine their respective CS and education expertise.

A course teaching Computer Science students about teaching may not develop a large number of Computer Science teachers, but it will enrich the students' undergraduate experience, provide resources to existing teachers, and enlighten soon-to-be professionals about computing education. It also encourages engagement between Computer Science and Education departments, providing a base for other initiatives.

7. REFERENCES

[1] Computing Curricula 2001 Computer Science, Association for Computing Machinery, http://www.acm.org/education/education/education/curric_vols/cc2001.pdf. Accessed September 8, 2010.

[2] College Board, " AP Examination Volume Changes (2000-2010)", February, 2010. Accessed at http://professionals.collegeboard.com/profdownload/Exam-Volume-Change-2010.pdf , December 4, 2010.

[3] Tim Bell, Peter Andreae and Lynn Lambert. "Computer Science in New Zealand High Schools," Proceedings of ACE-2010 (Australasian Computing Education Conference).

[4] Peter Harsha, "Computer Science Majors Increase at Most Significant Rate Since Dot Com Boom", Computer Research Policy Blog, March 16, 2009, http://www.cra.org/govaffairs/blog/2009/03/computer-science-majors-increase-at-most-significant-rate-since-dot-com-boom. Accessed September 1, 2010.

[5] "Programming Dojo", by Samuel Williams, May 22, 2010http://blog.acm.org/archives/csta/2010/05/programming_doj.html. Accessed September 1, 2010.

[6] Jan Cuny & Chris Stephenson. "What's Going On With AP CS: The CS 10,000 Teachers Project", at The Computer Science and Information Technology Symposium, http://www.csta.acm.org/ProfessionalDevelopment/sub/CSIT10Presentations/Stephenson_Cuny.pdf, July 13, 2010. Accessed September 6, 2010.

[7] "A Model for Improving Secondary CS Education," *Proceedings of SIGCSE-2005*, St. Louis, pages 332-336.

[8] *Stuck in the Shallow End,* MIT Press, April 30, 2010.