An Evolving Educational Advisory Role for Computer Science

Andrew J. McAllister, Genevieve Audet-Perron, Amanda Gilks, Debbie McAnany, and Rick Wightman
Faculty of Computer Science, University of New Brunswick
P.O. Box 4400, Fredericton, New Brunswick, Canada, E3B 5A3

Abstract – *Recent movements like Code.org seek to have every school-aged child taught the rudiments of computer programming. The goals of such initiatives are to recognize Information Technology (IT) skills as a foundational science in the new digital age, and to remove barriers for entry of young people into IT professions. While these goals are important, such change brings with it new needs for collaboration between post-secondary Computer Science educators and other stakeholders, which include teachers, government departments, post-secondary departments of Education, and industry associations. This paper provides lessons learned from experience with a variety of such interactions by members of the University of New Brunswick's Faculty of Computer Science. Immediate goals for continuing and enhancing these interactions are described.*

Keywords: Computer Science Education, Recruitment, Coding, Misperceptions of Computer Science

1. Introduction

Computer Science (CS) has historically been taught most often by post-secondary institutions. Now Computer Science courses are being introduced more and more as electives at the high school level (for example [1, 7, 10, 12]) and even in earlier school grades (for example [2, 6, 8]). We are also starting to see efforts to promote coverage of fundamental CS topics such as coding for all students in schools [11, 13]. This latter push is being popularized by movements like Code.org in the United States [5] and Code Kids in Canada [4]. The rationale for universal exposure to CS at a young age includes the following:

(a) Virtually everyone in our society is experienced as a consumer of information technology, but only a select few understand what is involved in creating that technology;
(b) Computational thinking is starting to be seen as a foundational competency in today's digital age. During their career, many professionals in a wide variety of disciplines will bump into the need to modify a spreadsheet, understand a macro, or tweak a website. Widespread exposure to CS basics will dramatically lessen the sometimes painful learning curve involved with such tasks, and will tend to improve the quality of the results; and
(c) Over the last decade or so common misperceptions of CS have resulted in lower enrollments in CS degree programs [3, 14]. Widespread exposure to CS concepts and skills is seen as one way to combat this trend.

Rolling out CS education in school systems is not, however, without its challenges. School teachers are taught what they need to know to teach traditional school curriculum topics, but relatively few teachers have a background in CS. Information technology (IT) professionals are currently blessed with such promising and well-paid career opportunities that teaching positions are often seen as less attractive. Then there is the question of what should be taught. Should the focus be on skills such as coding? Or are there other topics that would also provide significant benefits?

As school systems in numerous jurisdictions worldwide wrestle with these questions, we see a new and growing role for post-secondary CS educators as advisors to this process. This paper describes our preliminary experience with this role at the University of New Brunswick (UNB), and provides predictions for how this role is likely to expand in nature and scope in the near future. Our goal is to help other educators become more effective in this advisory role.

2. Experience with Students

The recruiting and transfer advising teams within UNB's Faculty of Computer Science bring a wealth of insight into the questions posed above based on years of experience dealing with students faced with the opportunity to choose whether or not to enter a Computer Science, Information Systems, or Software Engineering degree program. Our recruiters:

- Meet face-to-face with over 5,000 students in approximately 75 high schools each year, covering schools in the Canadian provinces of New Brunswick, Prince Edward Island, and Nova Scotia, as well as the U.S. State of Maine;
- Interact with middle school students at dozens of yearly math and science fairs; and

- Conduct "CS Unplugged" sessions with dozens of local elementary school classes each year. These sessions involve no technology, instead showing young children how they can use algorithms such as sorting values or keeping information safe, using hands-on physical resources like tool boxes with locks and our "super fun sorting mat."

All of these interactions involve both (a) an educational component where we strive to dispel common myths and explain the true nature of Computer Science to the students, as well as (b) considerable discussion where we hear about the students' perceptions, concerns, fears, goals, and preferences. This experience has led us to a number of insights, and has given us the opportunity to develop, refine, and gauge the effectiveness of intervention strategies for dealing with various challenges.

Insight #1: Fear is a significant factor when students choose what to study after high school.

One of the most frequent comments we hear from high school and potential transfer students is, "I heard CS is too hard." This even occurs when the student in question is already in a technology-oriented university program of study such as Engineering and is considering transferring to CS. Friends will advise against the switch, saying they are likely to fail out of the much tougher program.

Other fears also come into play, such as:

- "I don't want people to see me as geeky."
- "I don't want to be in classes filled with geeky people; I want to spend my time at university with people like me."
- "I'm a people person. I don't want to spend my entire career sitting behind a computer and never dealing with people."
- "I have no idea what Computer Science involves (except that I would be studying computers) and that uncertainty scares me."
- "I've never taken any CS courses, but I'm afraid I won't like all the strictly technical subject matter covered in them."
- "I've heard there are no jobs in CS anymore; I'm afraid I'll get to the end of my program and not be able to find a job."

A considerable percentage of high schools have very little idea what they would like to do for their careers, so discussions about their options often focus on what they believe they *don't* want to do. It doesn't take much fear and uncertainty to rule out IT from the conversation.

Insight #2: Familiarity with a subject area lessens students' fears of the unknown.

Middle and high school students typically gain repeated exposure to subjects such as math, English, sciences, history, and so on. As a result they tend to develop confidence that they can handle the work involved with learning in these areas, because they have been there and done that. For the large number of graduating high school students who have no particular career path in mind, this confidence often seems to tip them in direction of programs such as Bachelor of Science or Bachelor of Arts. The student may not have much of an idea of where such a degree will lead them, but we frequently hear that at least they are not afraid their studies will be "too hard."

One of the benefits of providing CS instruction as part of core school curriculums is that CS would become one of those familiar and therefore somewhat less scary subject areas. We believe, however, that CS courses in schools can and do sometimes fail to alleviate other fears. For example, a course that focuses solely on coding can actually reinforce fears that a CS degree and resultant career would be all about sitting behind a computer and dealing strictly with technology.

Insight #3: Understanding the true nature of CS and IT career paths seems more important than competency in CS subject matter when selecting a degree program.

We almost never hear a high school student say they lack the confidence to enroll in a CS degree because they feel they lack sufficient skills in coding or algorithm design in order to successfully begin the program. Given that CS courses are not required in high school (and are not required for applying to CS degree programs), most students seem to realize this lack of knowledge is not a barrier to entry. What we do hear is, "I like using computers," or "I like playing computer games so I thought CS might be something I like."

When asked what they think CS is about, however, the answer is almost always, "I'm not sure," "I don't really have any idea," or "The study of computers, I guess." Our experiences in this regard are consistent with the findings of more formalized studies such as [9].

Young people learn what doctors, police officers, lawyers, and a wide variety of other professionals do through a number of life experiences, including books, television, movies, and real-life interactions with these professions, for example by having an annual check-up with a family physician. These experiences place these professions in the consciousness of young people and increase the chances they will consider them as possible career choices. The IT profession, however, is somewhat of an invisible trade by

comparison. Most young people are unlikely to wander into an IT professional's workplace. Scrum sessions and JAD workshops don't seem to be high on the list for inclusion in Hollywood films. Given this absence of information, the general public ends up having very little idea what IT is all about.

We have had good luck getting young people to understand the problem-solving nature of CS by using anecdotal stories, which we relay verbally to individuals or groups as the situation dictates. One favorite is about a computer scientist who visited a neonatal intensive care unit at a hospital and noticed a variety of electronic monitoring devices in use with each critically ill newborn. Collectively these devices were capturing a considerable volume of data on the status of each infant, however only a very small portion of the data values were being sampled, usually by a nurse or physician walking over to the machines and looking at the readings.

The computer scientist thought, "We should be able to make better use of all this data," and set out to create a software solution to analyze the data more completely. One result was that they were able to start detecting the presence of a particular type of infection up to a full day earlier, which of course means the infection is less well established and is easier to combat, with less impact on the health of the infant.

We explain this is what CS is all about – using information to solve problems with real-world importance. The fact that computers are often a part of the solution is not the point; such devices are simply our best current tools for storing and manipulating data into useful information in various forms.

We go on to explain that the same is true in other disciplines. Biologists use microscopes to study organisms, but biology is not the study of microscopes. Similarly, chemistry is not the study of test tubes, and CS is not the study of computers.

At this point in the dialog we usually see plenty of head nodding and "aha" moments happening. In many cases we have found that this level of insight is effective in turning young people on to the idea of becoming a problem solver who can inject real value into a wide variety of enterprises. Grover, Pea, and Cooper report similar success using a set of videos to explain the true nature of CS [9].

Insight #4: The use of online and self-directed learning seems more prevalent for high school courses with more technology-oriented subject matter, with sometimes less-than-optimal results.

Within the many high schools we visit each year, we have never heard of courses in English literature or history being offered online or via self-directed computer-based modules. There are a few examples, however, of "Broad-Based Technology" and computer programming courses that are offered in this manner.

Given how short the attention span of young people can be, it is our opinion that such a self-directed, technology-driven mode of instruction is often likely to be less effective than traditional in-class, face-to-face instruction provided by teachers. This opinion is backed up by considerable anecdotal evidence – stories we have been told of students who are keen to take CS at university, but change their minds after having a negative experience with a self-directed CS course in high school.

The primary reason why technology courses are taught this way in our local schools seems to be primarily due to the aforementioned lack of teaching personnel who are qualified to teach technology-related subject matter. For instance, in the approximately seventy schools we visit regularly, only three are fortunate enough to have a teacher on staff who is qualified to teach Java programming. It is no surprise, then, that our local school districts rely on "pre-canned" online subject matter in order to offer instruction across a larger number of schools.

3. Stakeholder Relationships

Figure 1 shows a number of the most important stakeholders involved in typical school systems, along with the most salient existing relationships (shown as black arrows).

The primary focus of this entire discussion is the teaching relationship between school teachers and their students. The hope is that appropriate skills and knowledge can be imparted to students, consistent with the three-part rationale presented in Section 1: developing creators of technology, developing computational thinking as a foundational competency, and enhancing uptake of CS as a profession.

This direct relationship between teacher and student is the focus of movements such as Code.org and Code Kids. The basic idea is that all students should learn the fundamentals of coding. In other words, these movements strive to influence what subject matter is presented to students in schools.

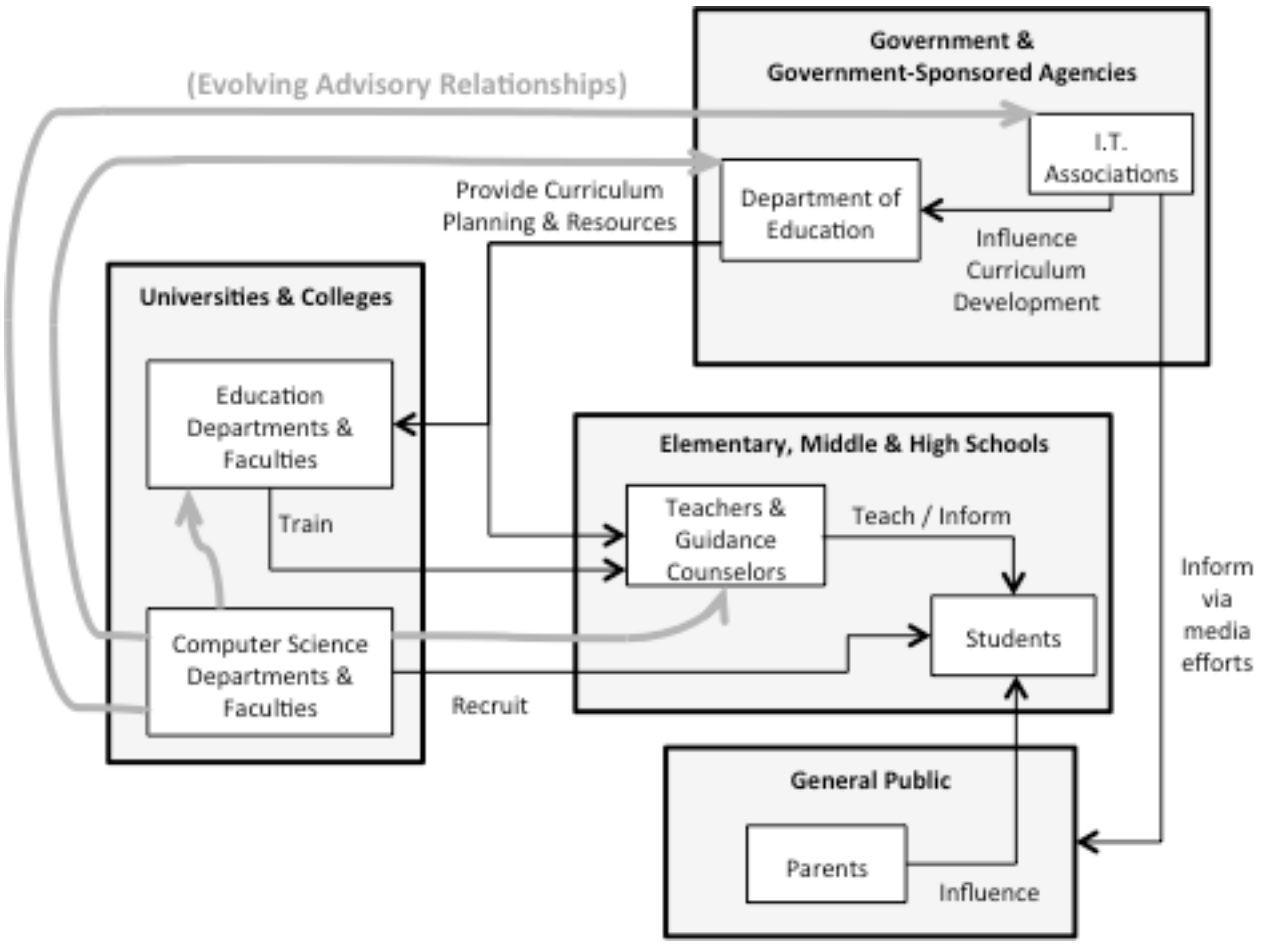

Figure 1: Educational Stakeholders and Relationships

As Figure 1 shows, however, the impacts of such changes ripple through a number of stakeholder groups:

- New teachers must be hired or existing teachers must be re-trained to be able to teach new subject matter;
- Teachers are normally trained, for example, by Education departments or faculties at universities or colleges. To implement a new foundational science across all (or even many) school curriculums, the teacher training at this level would have to evolve; and
- It is common for school curriculums to be developed centrally by government Departments of Education and provided for teachers across a given jurisdiction to deliver; this is the case in our province. Changes to curriculums impact this activity.

A number of industry- and government-sponsored associations are also on board with the need to increase exposure to CS among school students. These organizations typically use a multi-pronged approach for information dissemination, including influencing school curriculums. For example, the Information and Communications Technology Council (ICTC) of Canada conducts the Focus on Information Technology (FIT) Program. Participating provincial jurisdictions align their high school curriculums with a set of IT-related goals defined by ICTC. High school students can voluntarily sign up to participate in the program, and receive a diploma recognizing their efforts in completing relevant courses.

In many jurisdictions, the governmental Departments of Education and university-level Education departments suffer from the same challenge as the schools themselves, namely a lack of fully trained IT resources who can effectively (a) develop curriculums, and (b) train the teachers. For this reason we believe it is becoming increasingly important for university CS faculty members to provide expertise by taking on new advisory roles, which are shown as thick gray arrows in Figure 1.

At the University of New Brunswick we have already undertaken this advisory role. For instance, each year we conduct a full-day workshop for the entire graduating class of UNB's Faculty of Education. We explain how teachers can implement computational thinking into courses as diverse as English and mathematics, and the day includes participatory exercises that prepare these teachers-to-be to do exactly that.

We have also begun working with the ICTC to influence the types of IT-related knowledge encompassed by the FIT Program. Our hope is to work together to ensure students are exposed to not only just IT skills, but also material that helps dispel the most common fears and misperceptions surrounding CS.

4. Looking Ahead

Despite our progress in working with various educational stakeholders, we recognize that an ongoing and evolving road lies ahead. We are committed to serving as an effective source of help and information for continuing development of CS-related subject matter in local and national school curriculums.

At the present time we are working on addressing the limited effectiveness of rolling out CS courses in middle and high schools due to a lack of teachers with the required IT expertise. The fundamental question is this; what can we collectively do to significantly improve the information making it to all school students without requiring large numbers of qualified IT professionals to act as teachers? How can we leverage the human resource base comprised of existing teachers to improve upon the current situation, based on only a reasonably modest investment of cost, time, and effort?

We believe the answer lies in looking beyond the current efforts to teach coding to all students. Such efforts are likely to take time to implement, especially given the well-recognized shortage of teachers with IT knowledge. In the short term, we would like to help large numbers of teachers in multiple disciplines become effective advocates for CS. We have already started doing this with newly graduated teachers, and we hope to do so with existing teachers as well.

The key to this initiative is to recognize three factors:

1. Teachers can educate students about the true nature of CS using anecdotal stories of the type described in Section 2 above. Videos can also be incorporated into such presentations;
2. The ability to do so does not require that teachers have any specialized IT skills; and
3. Teachers can be prepared to take on this role in as little as a single Professional Development (PD) day. (The challenge, of course, is to convince school system officials of the value of this role, and to obtain agreement to allocate the required PD days.)

We are determined to serve as agents of this change, and we encourage other university and college CS educators to explore opportunities to take on similar advisory roles within your own local educational systems. The potential positive impacts on our IT economy and our society are sure to be well worth the effort.

5. References

[1] M.A. Bernardo, J.D. Morris, Transfer effects of a high school computer programming course on mathematical modeling, procedural comprehension, and verbal problem solution, Journal of Research on Computing in Education, 26, 4, Summer 1994, pp. 523-536.

[2] P.S. Buffum, A.G. Martinez-Arocho, M.H. Frankosky, F.J. Rodriguez, E.N. Wiebe, K.E. Boyer, CS Principles Goes to Middle School: Learning How to Teach "Big Data", *Proc. SIGCSE'14: 45th ACM Technical Symposium on Computer Science Education*, March 2014, Atlanta, pp. 151-156.

[3] L. Carter, Why students with an apparent aptitude for computer science don't choose to major in computer science, *Proc. SIGCSE'06: Symposium on Computer Science Education*, Houston, March 2006, pp. 27–31.

[4] Code Kids, *http://www.codekids.ca*.

[5] Code.org, *http://www.code.org*.

[6] J. Denner, L. Werner, E. Ortiz, Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts?, *Computers & Education*, 58, 1, Jan. 2012, pp. 240-249.

[7] W. Feurzeig, S. Papert, B. Lawler, Programming languages as a conceptual framework for teaching mathematics, *Interactive Learning Environments*, 19, 5, Dec. 2011, pp. 487-501.

[8] M.N. Giannakos, L. Jaccheri, R. Proto, Teaching Computer Science to Young Children through Creativity: Lessons Learned from the Case of Norway, *Proc. CSERC'13: 3rd Computer Science Education Research Conference*, Arnhem, Netherlands, April 2013, pp. 103-111.

[9] S. Grover, R. Pea, S. Cooper, Remedying Misperceptions of Computer Science among Middle School Students, Proc. SIGCSE'14: 45th ACM Technical

Symposium on Computer Science Education, March 5–8, 2014, Atlanta, pp. 343-348.

[10] J. Liebenberg, E. Mentz, B. Breed, Pair programming and secondary school girls' enjoyment of programming and the subject Information Technology (IT), *Computer Science Education*, 22, 3, Sept. 2012, pp. 219-236.

[11] H. Partovih, Transforming US Education with Computer Science, *Proc. SIGCSE'14: 45th ACM Technical Symposium on Computer Science Education*, March 2014, Atlanta.

[12] T. Paz, D. Levy, Introducing computer science to educationally disadvantaged high school students, *Research in Science & Technological Education*, 23, 2, Nov. 2005, pp. 229-244.

[13] E. Shein, Should Everybody Learn to Code?, *Communications of the ACM*, 57, 2, Feb. 2014, pp. 16-18.

[14] S. Yardi, A. Bruckman, What is computing?: bridging the gap between teenagers' perceptions and graduate students' experiences, *Proc. ICER'07: Third International Workshop on Computing Education Research*, Sept. 2007, Atlanta, pp. 39–50.