Introduction

Over the past decade the field of computer science has grown in importance and in popularity. As a result of the dotcom bubble in the late nineties and the expanding pervasiveness of computers in the world around them, many students decided to pursue careers in computer science. Unfortunately, with a curriculum that had not been updated since 1984[1], many students left high school ill equipped to pursue their goals of higher education or finding jobs in the field of computer science.

In March of 1999, Ontario's minister of education, Dave Johnson released a new curriculum for high schools with the aim to "set consistent province-wide standards for students' high school education to ensure they have the skills they need to succeed"[2]. For the most part Dave Johnson and the Ontario government have achieved their goal of laying down comprehensive expectations outlining the skills and concepts that students should be taught in a high school. Unfortunately, the sweeping changes ushered in by the new curriculum did not fully address *how* students are to meet these expectations.

For the most part, high schools are doing an admirable job of implementing the government recommendations and ensuring students gain a full understanding of the computer science concepts contained in the new curriculum through testing and assignments. In doing so, high schools give most students the preparation they need for university computer science programs. While this preparation is adequate, there exists a gap between what high schools are currently doing and what could be done to go beyond the basic needs of students.

Examination of curriculum and courses

The high level description of the guidelines given by the high school curriculum can be quite vague. One example of ambiguity in the new curriculum is in the teaching of programming in high schools. The curriculum set out by the Ontario government

[1] Grade 11, 12 Curriculum, Ontario Ministry of Education 2000, http://www.edu.gov.on.ca/eng/document/curricul/secondary/grade1112/english/english.html
[2] Ministry of Education News Release (March 4, 1999), http://www.edu.gov.on.ca/eng/document/nr/99.03/second.html

specifies the programming concepts to be learned, but not the language that must be used to learn them. The Ontario government is right to allow flexibility in how the concepts are taught, but without any concrete support in selecting how to teach programming, high school teachers often make choices that are less than ideal. In 2001, two years after the new curriculum was introduced a survey was conducted by the Association of Computer Studies Educators (ACSE)[3] in which 182 out of 402 high school teachers responded that they would like to see Visual Basic licensed for use in teaching programming. The same survey asked 12 universities what languages they thought were and were not suitable for teaching in high school. 5 of the 12 said that Visual Basic was *not* suitable and none of the 12 universities thought that it was suitable. Clearly there is a discrepancy here. There needs to be more support to allow high school teachers to make informed decisions about how to best implement the current curriculum.

Streaming in the Curriculum

A problem that has been noted with the new curriculum is the large overlap between the grade 10 TIK (an 'open' computer and information science course) and the grade 11 ICS (a computer and information science course geared towards university and college preparation) course. Currently there is no prerequisite for the ICS course. Some students take both courses, while others begin learning about computers for the first time in the grade 11 ICS course. The extra course that some students have taken leads to a schism between those who have a good understanding of the first part of the course because it is review and those who are learning for the first time. Those who are forced to review material may become bored and complacent while those who are learning for the first time may find their selves struggling to learn concepts that seem simple to other students. This issue is indicative of the more pervasive problem of accommodating students with varied backgrounds. One possible solution is a further splintering of the curriculum into courses with an even finer focus on material. Instead of one grade 11 ICS course there could be two in order to accommodate each group of students. However, splitting up the high school curriculum into extremely specific streams is not the answer. We are forcing our sons and daughters to make what can amount to career

[3] Ministry-Licensed Computer Programming Software Report, 2001 by ACSE (See Appendix)

choices as early as grade 10. The objective of high school should be to give as well rounded an education as possible, providing students with skills they need to succeed not only in one specific area, but in whatever endeavor they choose to pursue. Another, more pragmatic solution is embracing the varied past experience of students by offering a curriculum focused on the individual. Students with weak backgrounds can get the extra help they need and students who have stronger backgrounds are provided with enrichment materials to allow them to excel.

Inadequacy of Equipment

In 1999 the province injected $150 million into high schools in the two years following the introduction of the new curriculum. Some of the high school teachers did not feel that this was enough. Kevin O'Reilly, a computer science teacher at Forest Heights Collegiate Institute is forced to teach in an old tech class room that is not well equipped to teach a computer science course[4]. The room is so awkward that some students cannot see the teacher, who has only a small white board on which to write. If we want our students to learn the latest concepts in computer science we should equip them with the latest technology; the computers used at Kevin's school are only replaced every five years. According to Moore's law the speed of computers will increase more than three fold over that period of time. Most of today's software simply cannot be run on a computer that is 4 or 5 years old. This is another example of how the government has put forth what they deem to be a current and relevant curriculum, but does not supply the resources to correctly implement such a curriculum.

Preparing Students for CS at Waterloo

The computer science curriculum at the University of Waterloo is "centred around the study of information"[5]. The best way for high school students to prepare for the computer science program at the University of Waterloo is to develop proper analytical and problem solving skills and to be comfortable with computers.

[4] Kevin O'Reilly's lecture notes for his presentation to the CS492 class
[5] http://www.adm.uwaterloo.ca/infoucal/MATH/comp_sci.html

Problem Solving Skills

For instance, the grade 11 computer science course at Forest Heights Collegiate Institute does not do enough to encourage students to apply problem-solving skills. From an email sent by Kevin O'Reilly[6], the grade 11 computer science teacher, it seems as though he just wants to give students an exposure to programming concepts but unfortunately the course is taught in a computer lab where students are tempted to just code away when they are in front of a computer. Programming is more than just coding. It is problem solving involving the design and justification of an algorithm which is then coded. An ideal approach would be to first solve the problem on paper, then design pseudo code to solve a problem, and then finally to code the solution instead of immediately coding away in front of a computer without an fore thought. Of the mathematics and science courses that are required to enter the computer science program at University of Waterloo, they should have a pretty solid problem solving foundation.

Student Participation

It is no secret that students hate to have to sit in class and listen to teachers explain abstract concepts. If teachers present the concepts in a more appealing manner, students may pay attention and learn the concepts. The way in which the object-oriented techniques were demonstrated to the grade 11 computer science students was very effective. A visual demonstration of the object-oriented way of making a sandwich is a good way to teach a simple concept while incorporating some entertainment value. Getting the students involved in the presentation is a good idea as well.

Analytical Skills

Analytical skills are a really useful tool for the computer science program at the University of Waterloo. It is almost impossible to write a program that is free of errors. No matter how careful a person is when he or she is coding, minor glitches will always be present in the final program. These glitches may be caused by a conceptual error or are just simple syntax errors. No matter what type of error it is, students must apply their analytic skills in order to debug their programs. The grade 11 computer science course at

[6] Email was sent from Kevin O'Reilly to Kevin Regan

Forest Heights Collegiate Institute encourages students to apply their analytical skills. At the University of Waterloo coding assignments are done in pairs, embracing extreme programming paradigms. Students are forced to analyze their own code and the code of their group member in order for their program to work. Working in groups also forces students to be responsible programmers. Students will be more responsible knowing that the code they wrote may conflict with the code of their group member. Since some of the computer science courses have coding assignments that are done in groups, it is good practice for students to learn how to program as a team.

Last but not least, interest in computers is a must for any students who wish to enter the computer science program at UW. Unfortunately, there is no way for a high school to teach students how to be interested in something. If a student is interested in computers, the grade 11 computer science course offered at Forest Heights Collegiate Institute does do a good job in preparing a student for a career in computer science. While more emphasis can be put towards the theory of computer science, there is only so much material students can absorb, especially now that five years worth of learning has been crammed into four.

Appropriateness of Exams and Assignments

After reviewing some of the testing procedures and assignment material from computer science high school teacher, Michelle Vidberg it is clear that her high school is moving in the correct direction. They have successfully incorporated basic programming fundamentals, problem-solving techniques, and have also made concerted attempts to make the content entertaining for the student in order to maintain interest. Some group work is also introduced which is quite effective at introducing the student to modularized programming concepts. Societal influences like the topics studied in CS492 are also covered and directed by the students in order to help them see the positive and negative impacts of computers on society. However, the curriculum lacks several elements that would help to provide a positive learning experience in post-secondary education at the University of Waterloo, namely: learning objectives, marking subjectivity, and knowledge relevant to academic/workplace environments.

Learning Objectives

While knowing syntax and common functionality, such as interfaces to frequently used classes, is important for programming efficiency and speed, teaching a student to memorize common APIs will only help them to a certain degree. Consider an age old adage that applies quite nicely this scenario: “give a man a fish and you feed him for a day; teach him how to fish and he’s fed for life”. In the same regard, a teacher can have a student memorize syntax for native Java or Pascal functions, but it is more effective to show them how to figure out where to look for that same information. If a student is aware of where to look and how to search for the information (i.e. learning research methodology), then the task in finding out how to formulate a call to a particular function is relatively easy. Moreover, in an academic setting such as at UW, and in a work environment (co-op or full-time work), students and employees alike are not required to memorize things like syntax or common interfaces. It has limited scope and ends up teaching the student a “skill” rather than giving them the wisdom to learn it themselves; this is an incredibly invaluable lesson for any University student.

Marking Subjectivity

In order to accurately reflect a student’s understanding of the depth/breadth of a computer science course, it is necessary to incorporate as much objective marking as possible and to limit the subjectivity as much as is realistically possible. The practice of objective marking will, in the long run, more accurately reflect how well a student understands a topic. An example of subjective marking is a teacher who takes observational notes on how quickly a student generally got started on tasks, the extent of their documentation, etc. In reflection one must ask the following question: why mark a student based partially on their behaviour in a computer science class and not do the same thing in a science class while conducting lab experiments? Grades such as these may not accurately reflect a student’s proficiency in the subject, as perhaps a student lacks the proper knowledge to continue (which reinforces the aforementioned suggestion of learning how to research and learn on one’s own).

Applicable Knowledge

There are several types of categories that students fall into with respect to computer science and reasons for enrolling in those particular classes: students that "have to", students that "want to", and students that are not sure but are curious about the subject. Although it is unclear as to whether or not there are individual courses tailored to different long-term student goals, that students looking to pursue a career in computer science at university are exposed to sufficient material in order to be adequately prepared. According to the University of Waterloo, grade 11 students should be taught about repetition, selection, file I/O, and should have exposure to functions and procedure methodologies. Grade 12 students should possess the former in addition to knowledge about arrays, objects, parameter passing and designing programs using systems of interacting objects.[7] This is not to say that all students should be learning this material, because some simply learn at a slower pace or lack the motivation to learn. The curriculum examples given by the teachers met most of these different objectives, but lacked focus on method invocation and parameter passing[8].

General Recommendations

Pairing UW students with high school teachers

During his talk to our CS492 class Kevin O'Reilly had a number of complaints, such as:

- Industry people do not respond to his requests for problems for students to solve
- He has difficulties recruiting guest speakers
- People have donated machines but he is unable to use them because he does not know how to set them up

[7]http://www.cs.uwaterloo.ca/High_School_Liaison/Symposium_Presentation_V3/Symposium_Presentation_V3.ppt

[8] It is possible that this specified material was covered in due course(s), but it was not tested nor was it in the example assignments

- He does not have time to spend learning additional computer science concepts and may be presented with questions from student he is unable to answer[9].

We believe that Mr. O'Reilly and other computer science teachers could benefit from a volunteer program whereby university computer science students are paired with a high school, especially considering that the vast majority of high school teachers had no formal training in computer science.

Such a program would enable high school teachers to use their university student volunteer as a sounding board and problem solver. For instance a university student could provide a teacher with the following.

Access to guest speakers

After several work terms university students have made both a number of contacts at companies they have worked for and with professors at their schools. Student volunteers can then facilitate guest speaking engagements between a high school and an industry professional. Alternatively, upper year students could speak to the high school students themselves as they now have considerable experience.

Real world problems

Again through both their work term experiences and university courses student volunteers have been exposed to a number of real world problems which if simplified could be used in a high school setting.

Expertise in helping with special projects

While Mr. O'Reilly may not have the experience to setup his donated servers it is likely that a student volunteer either would be able to do so or could put Mr. O'Reilly in touch with another student who could.

Knowledgeable resource

Student volunteers could, time permitting, answer short questions that teachers may have time to time over email.

[9] Kevin O'Reilly's lecture notes for his presentation to the CS492 class

This program would serve to alleviate some of the stress and load that some teachers may feel while teaching computer science in high school. At the same time this program would give university students a chance to give back to their community by using their specialized skills in computer science.

Conclusion

After examining the high school computer science curriculum and its implementation at two schools in Southern Ontario, it is clear that the curriculum addresses the fundamental needs of students. There are concerns with the way the curriculum is being implemented, such as programming language choice, and relevance of the material being taught. However, considering the lack of government support faced by teachers, they are doing an admirable job.

While students, schools, and parents could lobby the provincial government for increased funding, a more pragmatic solution is to pair university computer science students with schools to give the teachers the extra support they need.

By doing so, we can help to ensure a brighter future for those, who see the relevance and importance of the field of computer science and choose to pursue it.

Bibliography

Ministry-Licensed Computer Programming Software Report, 2001 by ACSE (See Appendix A)

Grade 11, 12 Curriculum, Ontario Ministry of Education 2000, http://www.edu.gov.on.ca/eng/document/curricul/secondary/grade1112/english/english.html

Ontario Ministry of Education News Release (March 4, 1999), http://www.edu.gov.on.ca/eng/document/nr/99.03/second.html

O'Reilly, Kevin. (2003). Lecture notes for his presentation to the CS492 class

O'Reilly, Kevin. (2003). Email to Kevin Regan

University of Waterloo. Computer Science Undergraduate Calendar. http://www.adm.uwaterloo.ca/infoucal/MATH/comp_sci.html

Weber, B. Graham, S. The University Perspective on the new High School Computer Science Curriculum. http://www.cs.uwaterloo.ca/High_School_Liaison/Symposium_Presentation_V3/Symposium_Presentation_V3.ppt

Appendix A

The Association of Computer Studies Educators
Ministry-Licensed Computer Programming Software Report
2001 01 27
By Graham Smyth

The Association of Computer Studies Educators has lobbied the government to license software for use in teaching programming in the high schools. The Association is concerned about the lack of Ministry-licensed software to support the Computer and Information Science and Computer Engineering courses in the new Ministry curriculum. In the absence of Ministry-licensed programming language software for computer studies, an increasing number of schools will not be able to afford to offer theses courses for students. To this end, ACSE has sent a survey to Computer Science high school teachers asking
which language(s) they would like to see licensed by the Ministry of Education. The teachers were asked to select two of the following languages:

- C/C++
- Delphi
- Java
- Pascal
- Turing/Object Oriented Turing
- Visual Basic

Of the 403 teachers that responded to the survey, they selected the language the number of times indicated in the following chart:

Language	Number of Times Selected
C/C++	117
Delphi	32
Java	185
Pascal	30
Turing/Object Oriented Turing	199
Visual Basic	181

ACSE also surveyed 12 Universities/Colleges to determine which languages (from the same list given to high school teachers) they felt were suitable or not suitable for high school computer science instruction. Twelve university/college people responded. These results are as follows:

Language	**Suitable**	**Not Suitable**
C/C++	3	4
Delphi	1	3
Java	8	0
Pascal	5	2
Turing/Object Oriented Turing	6	1
Visual Basic	0	5

The results of this survey were sent to the Ontario Software Acquisition committee (John Taylor correspondence representative). They have responded and developed criteria for a RFP for Programming languages and hope to post very soon for evaluation in February.