New Methods in the Teaching of Computer Science

A Research Paper submitted to the Department of Engineering and Society

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Alexander Hicks
Spring, 2020

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

New Methods in the Teaching of Computer Science

Computer Science Education

It is generally assumed that the first computer, mechanically speaking, was the Difference Engine, created in 1822 by Charles Babbage (Computer Hope, 2019). Since then, computers have grown exponentially, into the complex mix of cloud and personal computers that exist today. Computer science is a growing discipline, and as the world becomes more technological, computer science has only become a larger and more important part of the world. To this end, this paper explores how computer science has been taught, specifically pertaining to the introductory classes and how the methods used can be updated through research into the state of the art of computer science education. Due to this prevalence, more people need to know at least the basics of computer science, and in order to know that, the teaching methods of the introductory courses must be adapted to avoid issues such as the persistence problem described by Leslie Cintron in 2019. In order to look into this, this paper explores technological momentum, a prominent framework in the field of Science, Technology, and Society (STS). As discussed by Hughes in 1994, technological momentum distilled down is the idea that "social development shapes and is shaped by technology," and as such fits this discussion of computer science education neatly.

How Can Computer Science Education Be Improved

Computer science is a useful tool to help solve a variety of problems, but it can require a relatively high degree of skill compared to alternative methods or solutions. Even with this impediment, computer science is one of the fastest growing engineering disciplines in the world (Kay et al., 2000). However, it must be kept in mind that in order to open the field of computer

science to more individuals and a more diverse population, several questions must be answered, including this research question of:

> How are introductory computer science classes, specifically CS1 (CS 111x) and CS2 (CS 2110), taught at the University of Virginia, and how can they be improved?

This question poses a unique problem that requires two primary research methods, documentary research methods and interviews, in order to focus on investigating how CS1 is taught at the University of Virginia, the main research method is interviewing UVA computer science professors, specifically those who currently teach or have taught an introductory computer science class here within the past five years, because they are the only source for much of this information (Seabrook, 2019, Slide 5 and 11). In addition, documentary research methods are used in order to analyze how other institutions handle teaching CS1 and CS2 education. Two important sources for this research include the journal, Computer Science Education, and public coursework from universities like Harvard and the University of California, Berkeley.

What is the Current State of the Art

Computer science education provides an interesting case study of education because of the relative youth of the discipline as well as the rate at which it is currently growing. Currently computer science is taught in several different ways, with differences including the programming language chosen, the style of the homework, group work versus individual work, in class activities versus labs, and more. This disparity between classes, in addition to other factors, has led to what is known as a persistence problem in computer science. This persistence problem is the idea that students may take one or even two computer science classes, but due to a variety of reasons including a lack of role models, new and unfamiliar content, large or impersonalized

classes, and different teaching methods than they may be used to, they will not continue further into computer science (Cintron, 2019).

Teaching computer science can also be different because the goals of the class are not always obvious or clear. For example, it is unclear whether an introductory class should teach a language or some subset of general computer science concepts (Rice and Rosen, 2017). Most courses assume no programming experience and attempt to teach both, but that leads to problems when students can associate important concepts with specific features of the language used.

Another important difference in teaching computer science is the difficulty of testing the students' knowledge. Computer science is a very team-oriented career, so ideally students should learn how to work in a team. This creates a problem where it can be hard to make sure that every student is learning the material and not getting left behind in a group. If students do not learn how to work in groups, it is easier to assess their knowledge through standard means such as tests, quizzes, and projects, but they may be missing out on learning group and teamwork related skills relevant to computer science.

One final interesting note is that learning to google is a very different skill in computer science than it is common life. The skill required to look up error messages and read programming errors is vital to being a computer scientist, but is rarely taught explicitly, especially in lower level classes. There are many other small tips and tricks such as this that are difficult to embed in traditional curriculums and many times get left out of CS1 and CS2 at universities.

TECHNOLOGICAL MOMENTUM AND COMPUTER SCIENCE EDUCATION

In order to analyze computer science education from an STS perspective, this paper will utilize technological momentum, a prominent framework in the field. Technological

momentum, according to Hughes in 1994, can be distilled down to the idea that "social development shapes and is shaped by technology." Since technological momentum is time dependent, it can be applied naturally to computer science education, which is constantly growing and expanding due to the current growth of computer science itself. In order to investigate this, technological momentum is used to analyze how computer science education has grown with computer science and how that growth has shaped computer science as well as how the world is becoming a more technological place has increased the need for software developers and has led to coding boot camps popping up. Coding boot camps are short classes that can supposedly make someone a software developer in weeks or months instead of years at a college or university. This paper will investigate how those boot camps compare to a more formal education.

Computer science is a growing discipline, and as society gets more and more technological, computer science will become more and more important in day to day life. To this end, computer science education must continue to improve and adapt as well. Developing new methods to provide a more welcoming environment within the discipline for minorities is a current problem and it will only get worse as time goes on. It is important to provide an environment for students to be able to change their mindset from the idea that they are just not "computer people" and open themselves up (Sherriff, 2016). Ideas like this and others have led to the persistence problem described above and is a widely held worry within the computer science community (Mau, 2003 and Cintron, 2019).

Technological momentum is a powerful theory, but it does have critics. Since technological momentum shares a history with both technological determinism and the social construction of technology, it shares their flaws as well as their benefits (Hughes, 1994). While

technological momentum can lead to new and greater things, it can also include unintended consequences such as, in the context of this paper, students getting left behind by more time economical teaching styles. Despite these drawbacks of the framework, technological momentum is a powerful tool for this topic.

What was found

Significant research is currently in progress in the field of computer science education. Specifically, at the University of Virginia, a new pilot curriculum is currently being tested, that involves the complete redesign of lower level computer science classes at UVA including the focus of this paper, CS 111x and CS 2110. The pilot program has focused on several goals: to better prepare students for internships earlier in their computer science careers, cut down on duplicate information, and to reorganize the program topically in order to better cover important material. This reorganization of much of the curriculum has taken up a significant amount of time and energy by the UVA computer science faculty throughout the last eight years as it has been developed and implemented and has not left much room for more change while it is being tested. The professors that were interviewed for this project thought that changing the method of instruction would increase the retention by students overall, but they have already cut a significant amount of material from courses such as CS 111x and worry that slowing down any further would leave students unprepared for their following classes.

UVA computer science is currently focused on issues such as those identified in this paper. According to UVA Professor Luther Tychonievich, the computer science faculty have been focusing on a curriculum redesign for about eight years now, which shows just how important they feel this issue is. The faculty had pinpointed several issues that they believed they needed to address in order to open up computer science to more people. Currently at UVA,

there are two computer science majors, a Bachelor of Arts (BA) from the College of Arts and Sciences (CAS) and a Bachelor of Science (BS) from the School of Engineering and Applied Science (SEAS) (Tychonievich, 2019). While computer science faculty including Professor Tychonievich and Professor Cohoon believe the BA program, which was added only in 2006, has been a huge success in opening up computer science to more people, it has come with its challenges, most importantly the movement of material in and out of classes to make sure two majors with different sets of requirements both provided a rigorous education for their respective students. This dual curriculum, along with the continuous growth of the department, especially in recent years, has led to some duplication of material across several classes, such as the introduction of Hash Tables in CS 2110 as well as CS 2150 and the discussion of NP-Completeness in CS 3102 and CS 4102, and too much material being included in others, such as CS 2150 and CS 4102 as discussed in the previous example. This duplication of material was recognized by the faculty and a new curriculum was created in order to address these issues. The new curriculum has been piloted for the 2018-2019 and 2019-2020 academic school years, before it will be reworked to implement some of the changes needed that were highlighted by the pilot program (Sherriff, 2020).

The pilot program shows both that there is current research and work going into adapting computer science education as it is needed as well as why it is important. The discipline grows so fast that many professors do not even use textbooks for their classes due to material moving so fast (Graham, 2019). This growth also occurs in industry as well, with new frameworks and programming language being created in the last few years. This industry growth requires that students take classes on updated material and learn concepts that will be relevant to their future jobs. This is another stated goal of the new computer science curriculum, to better prepare

students for internships earlier in their education. This earlier experience with internships will give students the opportunity to hold multiple internships before they graduate as well as come out of college with more experience and real-world skills that cannot be taught at UVA. This vision shows that computer science education at UVA is growing with the needs of industry, which is evidence of the STS framework of technological momentum. As computer science grows and expands, computer science education must grow and expand as well. This is clearly happening at UVA through this pilot program and is also the reason the pilot program was needed, due to this growth in individual classes being unchecked for several years before it was able to be fixed.

Another current computer science education project currently taking place at UVA is the Lighthouse for Computer Science project. This team was co-founded by UVA Professor Jim Cohoon and one of its National Science Foundation (NSF) grants explores introductory computer science education, specifically CS 1110 and CS 1112 at UVA (Cintron, 2020). Some of the focuses of the project include educating teaching assistants to provide a more welcoming experience to students, especially those who did not previously intend to follow computer science past their introductory class. According to Leslie Cintron, the Lighthouse team also has implemented a peer mentor group program for CS 1112 for students to periodically meet with a teaching assistant that has volunteered to answer questions about the class and computer science in general as well as check in on the students to make sure they are welcomed in the class. The goal of this project is to avoid issues of the persistence problem that was highlighted earlier in this paper. In his article, Robins focuses on the fragility of novice knowledge as well as complexity in programming languages, and how those concepts can be retained, one way is through more contact between course staff and students as well as more contact between students

themselves. The goal of these mentor groups by the Lighthouse team runs parallel to these ideas and is to provide a group for the students to feel comfortable with in the class to avoid leaving them behind if they run into trouble (Cintron, 2020). This group attempts to keep updated with current research into computer science education and projects happening around the country in order to find new interventions for introductory computer science classes at UVA. While interviewing Dr. Cintron, she remarked that ideas about increasing classroom time with students and slowing the pace of class would help from an education standpoint, but according to her perspective as an education researcher attempting to convince computer science faculty to change their teaching methods, the amount of material that would have to be cut from somewhere in the program makes this method prohibitively difficult to implement.

According to Professor Tychonievich, there is currently too much research and change occurring in the computer science department to even consider more change. The pilot program is currently taking feedback, but despite a reorganization of material, the program transitions from seven classes to six for the lower level component of the computer science degree, which means there is even less space to slow down material. Additionally, he also mentioned that he was already having to cut material from the pilot class that he is currently teaching and that Professor Floryan was having to do the same in the other section. Additionally, it would take away from methods such as problem-based learning, which has been shown to perform better in introductory classes for students to get a better idea of what computer science is like (Kay et al., 2000).

While this paper explores a broad topic in theory, due to the focus on UVA computer science, there is a much narrower area for possible implementation of any ideas as a result of this paper. Currently at UVA, the preexisting research projects in education are taking up too much

time from faculty to really look into new ideas currently, specifically one about slowing down the pace of instruction and leaving out material. Another problem highlighted by Professor Tychonievich and Professor Floryan was that even now computer science labs are not counted in the credit counts for lower level courses. For example, CS 2110 has lecture for three hours per week as well as a one hour and forty-five-minute lab section once a week, but only counts as three credits for a student's schedule. Even with the current system, professors are having to leave information out that they would like to teach simply for time constraints. Further, one of the reasons this new curriculum was needed was that professors were adding new topics into their classes as computer science involved and it finally reached the point where pruning of topics and reorganization was needed.

Several ideas for future research on this topic could include exploring other ways of maximizing student interaction between both other students as well as with course staff. Ideas like the Lighthouse team's peer mentor groups are paralleled in higher level courses through group projects and teaching assistants who oversee their progress, and that idea could be explored further by creating a framework for introducing students to project based learning outside of the class environment early. Additionally, another direction this research could lean in would be exploring how students respond to error messages and teaching how to interpret errors in the introductory languages. Debugging is one of the most important problems in computer science but it tends not to be taught in courses explicitly. Focusing on an interface to a compiler that would translate the errors to a more human readable format for intro students would be an interesting problem to look at. A somewhat related direction would be looking into the trade off between time in lecture and time in labs. Computer science classes at UVA are moving away from physical labs due to the nature of the material, but a future project could verify the students

are still getting the benefits from the assignments if they are not doing them in the intended environment.

CONCLUSION

UVA is currently going through a process to improve how their introductory computer science courses are being taught in several ways, the most important being the new pilot curriculum and the Lighthouse Project at UVA. These two programs have different stated goals, but both are concerned with making UVA computer science a better and more welcoming program for everyone. They do have their differences however, including the fact that the pilot program is focusing on movement of the current material into a more reasonable and easily covered structure while the Lighthouse project is looking at ways to change the current options for introductory computer science, CS 111x and CS 2110 in order to make them more open to students who might be undecided about pursuing computer science as a major or career option prior to taking the class. With both of these projects happening concurrently, introductory computer science professors are focused on making sure that the basic concepts that need to be covered are not lost in the restructuring. To this end, the general response was that any attempt to slow down the courses, which is what transitioning to a new, slower in any way, experience would do, would be counterproductive to students going forward in their computer science coursework, because there would not be time to cover enough material. While the specific idea explored in the research question was not directly applicable to UVA computer science in particular, it could have applications to high school computer science, especially as that discipline grows and more and more students are taking their introductory class in high school instead of college, there does exist several projects that are currently ongoing to improve the introductory computer science courses at UVA and that is one reason there is no room for more

experimentation at this time. Computer science education is a complex topic and requires a delicate balance of providing a class that can be understood by students not pursuing further computer science education and a class with the requisite rigor to prepare students planning to major in computer science for their next level classes.

REFERENCES

Cintron, Leslie (2020). Lighthouse for Computer Science. http://www.cs.virginia.edu/lighthouse/team/

Computer Hope. (2019). When was the first computer invented. https://www.computerhope.com/issues/ch000984.htm

Graham, Daniel (2019). Algorithms. https://danielggraham.com/algorithms/

Horton, Tom (2020). CS BA Major in the College of Arts & Sciences. https://engineering.virginia.edu/departments/computer-science/academics/computer-science-undergraduate-programs/ba-computer-science#accordion67815

Hughes, Thomas (1994). Technological Momentum https://collab.its.virginia.edu/access/content/group/484deb3f-d8f1-405a-8b1c-5541ca7dd540/Readings/Hughes%20-%20Technological%20Momentum.pdf

Kay, Judy, Barg, Michael, Fekete, Alan, Greening, Tony, Hollands, Hollands, Kingston, Jeffrey H. & Crawford, Kate (2000) Problem-Based Learning for Foundation Computer Science Courses, Computer Science Education, 10:2, 109-128, DOI: 10.1076/0899-3408(200008)10:2;1-C;FT109

Mau, W. C. (2003). Factors that influence persistence in science and engineering career aspirations. The Career Development Quarterly, 51(3), 234-243.

Rice, John R. and Rosen, Saul (2017). History of the Department. https://www.cs.purdue.edu/history/

Robins, Anthony, Rountree, Janet & Rountree, Nathan (2003) Learning and Teaching Programming: A Review and Discussion, Computer Science Education, 13:2, 137-172, DOI: 10.1076/csed.13.2.137.14200

Sherriff, Mark (2020). UVA CS 2020 Curriculum Pilot. http://pilot.cs.virginia.edu

Tychonievich, Luther (2019). Department of Computer Science Undergraduate Handbook. https://github.com/uva-cs/ugrad-handbook

Appendix

Interview Questions

Why is the computer science department reorganizing the curriculum?

How will this reorganized curriculum better include and prepare students for their next steps?

What steps are being taken to focus on issues such as the persistence problem in introductory computer science classes?

Do you think changing from a TuTh or MWF schedule to a MTWThF schedule would help students learn in CS 111x?

What content would you like to include in CS 111x and what content would you like to remove from CS 111x?

What current things are being done to open up UVA computer science to more students?

How much learning from CS 111x happens in class versus how much learning happens outside of class through homeworks and other activities?