Recommendations for UVA Computer Science Curriculum to Create Industry Ready Graduates

A Technical Report submitted to the Department of Computer Science

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

Andrea Jerausek

Spring, 2023

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

Briana Morrison, Department of Computer Science

ABSTRACT

The computer science curriculum at the University of Virginia could be improved to help students build more practical skills to ensure they are industry ready. Meta analysis of research relating to the topic of creating industry ready developers during undergraduate studies is examined. In this evaluation of the current curriculum, I define success based on the results of third-party studies. More specifically, I compare non-traditional curriculums, such as programs that include open-source projects or that were created for experimental purposes, to UVA's traditional computer science program. Due to the rapidly changing and expansive nature of technology, it is impossible to create a curriculum that will fully satisfy all levels of industry; however, there are improvements that traditional curricula can make to create better software engineers, including increasing focus on soft skills such as collaboration and communication.

1. INTRODUCTION

Software engineering is a growing field with extensive technology that is being rapidly invented or updated resulting in a high demand for skilled developers. This means that a high supply of competent new graduates is essential to maintain and continue to keep pace with technological developments. The need for competent developers makes one ask what kind of curriculums produce the most industry ready software engineers? How does UVA's computer science program prepare students for industry in comparison to curriculums that utilize open source or multi-semester Capstone projects. Further, how do bootcamps, programs that attempt to get individuals technically proficient quickly, compare to four-year degree programs?

One should also ask whether industry demands too much from young developers. Expertise in any field requires years of experience, and asking for high levels of expertise from new graduates may be more counterproductive than helpful. Specifically, setting excessively high expectations for entry level roles may needlessly discourage people from entering the field.

2. SURVEY OF LITERATURE

Industry has certain expectations for new developers during interviews and a different set of expectations for the developers during their long-term work at a company. Lunn (2021) points out that most of the time, developers are expected to perform well on technical interviews that "place emphasis on specific topics." These kinds of interviews can only gauge a candidate's success in the field to a certain degree. In fact, hiring managers often complain that boot camp recruits do better in the interview process than those who have completed a four-year degree program. However, Wilson (2017) finds that graduates from universities tend to outperform boot camp recruits after six months despite their initial ineptitude. This is most likely due to each student's training focus. Bootcamp students tend to focus on learning specific frameworks and doing interview prep whereas college students tend to focus on developing general critical thinking skills.

Lunn identifies a number of skills that industries desire in new hires, including "competencies in knowledge and abilities" for hard skills and friendly personalities, critical thinking, and communication for soft skills. Both Lunn and Almi note that primary complaints from industry about university education include graduates' skill gap due to curricula placing heavy emphasis on theoretical concepts rather than practical skills. According to Almi, industry also complains that graduates have "choosy attitudes" which influence their willingness to learn new technologies that they may have not considered yet. He finds this problematic because adaptability, one of the primary desired skills in any profession, is particularly relevant in a profession where tools used for work are constantly changing.

3. ANALYSIS AND DISCUSSION

Before discussing various curricula, differences between software engineering versus computer science majors need to be considered, because there is a distinction between program styles even though both typically prepare students for future careers as software engineers. The distinction can be best explained by looking at another set of similar majors: physics and mechanical engineering. Mechanical engineering core principles are based on concepts from physics; however, the focus of the mechanical engineering major is the application of physics (Offutt, 2013). This distinction for physics and mechanical engineering is so well established now that when one majors in physics, they do not become mechanical engineers by default since they are not well equipped to do that job (Offutt, 2013). The same concept applies to computer science and software engineering majors. However, computer science curricula often attempt to teach practical aspects of the theory; therefore, most people with that major do move on to become software engineers in this case despite an emphasis on theory. Because a computer science major is more theory-based, homework is usually an individual task; thus, collaboration is discouraged, which may negatively affect students when working in industry due to the highly collaborative nature of the software engineering profession (Offutt, 2013).

Next, four different methods of curriculums will be addressed: traditional, innovative, open-source project oriented, and a short term boot camp.

3.1 Traditional

UVA's curriculum can be described as traditional. Characteristics of this program include learning material in class and applying it in a short-term assignment via exercise, lab or tutorial (Alasbali, 2015). In this type of curriculum, students typically have a couple courses in which a semester-long project is introduced, and seniors in the program also get Capstone projects that either span either half or their entire final year (Gary, 2008).

3.2 Innovative

An innovative curriculum is defined at Arizona State University Polytechnic, where students have two one year long Capstones that emulate enterprise projects (Gary, 2008). Students at this university take a course, try to apply it to their project, and then describe why the concept learned in the course does or does not apply (Gary, 2008). This program structure helps demonstrate true industry experience by projects involve "evolution, testing, analyzing, designing, and managing" of larger and continuous pieces of software (Gary, 2008).

3.3 Open Source

Having students participate in open-source projects forces them to participate in "authentic learning" (Alasbali, 2015). The primary goal of authentic learning is forcing students to complete tasks related to the real world in an environment they would feel motivated in (Alasbali, 2015). This type of learning should get students industry-ready as it makes students work on projects with existing code and with people in the OS project's community (Alasbali, 2015). Adding onto existing software and working/ communicating with others are both industry-relevant skills.

3.4 Bootcamps

The primary goal of bootcamps is to get individuals technologically proficient quickly (Wilson, 2017). Bootcamps typically choose to teach students about niche technologies/software that universities do not cover, and their style of learning is project-based (Wilson, 2017). This helps with hiring developers in specific areas of expertise; however, after six months this initial expertise soon becomes irrelevant when comparing performance of recent graduates of a 4-year program (Wilson, 2017).

4. EXPECTED BENEFITS

Benefits of developing a curriculum that will create the most industry-ready students have expected benefits for individual students, universities, and industries. Individuals will benefit from a better constructed curriculum because their career prospects will improve without needing to exert extra effort outside of working on their university coursework. In other words, students will be able to minimize the time they spend doing well with coursework while also preparing themselves to be well versed in practical skills that industry looks for.

In an ideal world, this would indicate that when students are completing coursework for official classes, they are also working towards developing practical skills in industry. A better-defined curriculum would also decrease the slope of the learning curve new graduates encounter during their first year in industry. This preparedness should translate into later career successes as graduates are able to contribute more valuable work to projects early on.

Curriculums that prepare graduates for industry benefit universities by creating more career opportunities for students. This increases a university's value, consequently increasing prospective students' interest in attending that university due to its potential to afford students greater career opportunities.

Industry would also benefit from a better structured curriculum, as that would create more prepared graduates. Better-prepared graduates would require fewer company resources to train new hires, suggesting a lower cost to onboard a new hire. Not only that, but work-ready graduates would contribute monetarily by producing more worthwhile work sooner.

5. CONCLUSION

This past summer I had the opportunity to work at Lutron Electronics on the Azure cloud team. While there, I worked with a variety of common tools used in industry during development, including–Azure Cloud, Visual Studios, BitBucket, JIRA, Postman, and others. I was also exposed to more advanced software architecture used when dealing with code that might be used to interact with a large number of users. Such architecture included concepts related to dependency injection and inversion of control.

At the time, I had limited exposure to the concept of the cloud and web development. With the UVA CS curriculum, most students will have zero exposure to the cloud and how it works unless they choose to take the cloud class (CS 4740) as an elective. In essence, there is limited access to gain knowledge about this essential component to developing software. Since developing software using the cloud is such an integral part of modern-day software development, this technology should be taught as a general education requirement. Knowing how the cloud works and how to optimize its usage is an essential skill that most students will need to have no matter what discipline of software engineering they pursue. This is especially important since according to Buloa (2022), platform as a service (PaaS) and infrastructure as a service (IaaS) are

predicted to grow by about 26% and 30% respectively this year.

Next, learning how to make software scalable is another essential skill I learned during my internship that the UVA CS curriculum could improve on. During my time at UVA, I often wrote code that would pass tricky test cases, and I am trained to consider various user inputs and how an individual user will break code that is implemented. However, what is greatly lacking is considering what happens if we run our code and a lot of users are attempting to use the software at once. This is especially relevant in industry where there might be thousands of users interacting with software simultaneously. In those instances, developers must consider if any latencies are introduced based on how they choose to implement features.

Both of these skills, cloud computing and scaling code, could be taught by structuring the curriculum to have group projects that span multiple semesters. A long-term homework assignment forces students to make smart decisions during development and to deal with problems introduced with their code sooner rather than later.

This method would better simulate a work environment. Not only that, but it would cause each student to learn how to deal with coworkers over a significant amount of time. Collaboration will be forced since no single student should be able to carry the workload of an entire project, and the need for cohesive software will make students learn to develop technical communication skills. Technical communication skills will be promoted in instances where students help each other. Essentially, a curriculum that is structured similar to one that Gary (2008) described for Arizona State University would greatly benefit students at UVA.

6. FUTURE WORK

To ensure competent developers are produced through 4-year degree programs, universities should reconsider the structure of their curriculums. In particular, the CS department at the University of Virginia should consider adding Cloud Computing as a required course on its curriculum. Teaching students how to scale code is another essential skill that would greatly benefit students when they enter the workforce. Both of these skills together could be dually applied in multi-semester long projects. Overall, it is beneficial to both students and employers if universities would consider teaching skills that are essential to the workforce.

REFERENCES

[1] Alasbali, N. and Benatallah, B. 2015. Open source as an innovative approach in computer science education a systematic review of advantages and challenges. *2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE)* (2015). DOI:http://dx.doi.org/10.1109/mite.2015.7375330

[2] Almi, N., Rahman, N., Purusothaman, D., and Sulaiman, S. 2011. Software engineering education: The gap between industry's requirements and graduates' readiness. *2011 IEEE Symposium on Computers & Informatics* (2011). DOI:http://dx.doi.org/10.1109/isci.2011.5958974

[3] Bulao, J. How many companies use cloud computing in 2022? all you need to know. Retrieved October 21, 2022 from https://techjury.net/blog/how-many-companies-use-cloud-computing/#gref

[4] Gary, K. 2008. The Software Enterprise: Practicing Best Practices in Software Engineering Education. (January 2008). Retrieved October 18, 2022 from https://www.researchgate.net/publication/263602679_The_Software_Enterprise_Practicing_Best_Practices_in_Software_Engineering_Education

[5] Lunn, S. and Ross, M. 2021. Ready to work: Evaluating the role of community cultural wealth during the hiring process in computing. *2021 Conference on Research in Equitable and Sustained Participation in Engineering, Computing, and Technology (RESPECT)* (2021). DOI:http://dx.doi.org/10.1109/respect51740.2021.9620686

[6] Offutt J. 2013. Putting the engineering into Software Engineering Education. *IEEE Software* 30, 1 (January 2013), 96–96. DOI:http://dx.doi.org/10.1109/ms.2013.12

[7] Wilson, G. 2017. Building a new mythology: The Coding Boot-camp Phenomenon. *ACM Inroads* 8, 4 (2017), 66–71. DOI:http://dx.doi.org/10.1145/3132706