Evaluation of University Curriculum in Internship Preparation

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

Binh An Dang Nguyen

Spring, 2022

Advisor

Daniel Graham, Department of Computer Science

Evaluation of University Curriculum in Internship Preparation

CS4991 Capstone Report, 2022

Binh An Nguyen
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
bdn4ef@virginia.edu

Abstract
Entry-level jobs are no longer geared toward new graduates looking for their first-real world experience. Standards for software engineering positions have risen and a degree is no longer enough to secure an offer. Summer internships provide computer science students an opportunity to expand their skills and garner needed experience. However, internships require countless interviews, a great deal of preparation, and a deep understanding of algorithms.

Classes at the University of Virginia helped me build the foundational basis for learning new skills and languages. In addition, the experience further developed my soft skills and improved communication. Although UVA offers material enriched courses, these courses have flaws in material and structure that restrict student growth. Courses can further be improved to help students gain more knowledge and better provide experiences for their future careers.

1. Introduction

Recent computer science graduates are facing a rising unemployment rate "of seven percent" [5]. Even though "many companies claim to have a severe shortage of tech skills," newly graduated computer science students still struggle to find a job [5]. A degree is not sufficient to land a job, but a bare minimum. In fact, the job market has become increasingly competitive with "more young people fleshing out their resumes before they even leave university" [1]. Entry-level jobs are not geared towards newly-graduated students who are looking for their first real-world experience, but for more "qualified developers with exceptional coding skills" [5]. Companies want someone with "real-time practice experience."

Internships provide students a way to "develop practice experience" [3]. Not only do internships showcase students' skills to employers, but also their ambition and potential growth. While "theory and academia" concepts learned from school "serve as a strong backbone for "topics such as Front-end, Back-end, and Mobile development," experience is just as important. UVA provided computer science students with a strong foundation with a heavy focus on theory and academics. However, with such a strong focus on conceptual computer science topics, students are not presented with enough opportunities to apply their newly learned skills in class.

Companies that look for interns or new grads want students who already have practical projects and coding skills. UVA's computer science program stresses general education on theories and concepts but fails to offer students opportunities to grow their technical and soft skills.

2. Related Works

The gap between academia and industry computer science has been a well-documented. Despite, the fact that "computer science, information systems, and information technology educators often do an exemplary job of preparing their students for jobs in industry or for further education, there are still many areas where these students do not possess the necessary skills or knowledge based on the expectations of employers or academia" [7]. Many computer science curricula struggle to prepare their students to meet industry expectations. There have been several studies and proposals on how to reduce this disparity between "graduates' skill and industry expectations" [2].

Most "recent graduates often fail to meet industry expectations when they first enter the workforce" [2]. Software engineering and other entry level roles demand a "large number of skills" for recent graduates [2]. University curricula have trouble "reproducing industry-like scenarios in academic settings" with many courses pushing students to work on projects that do not have industry value.

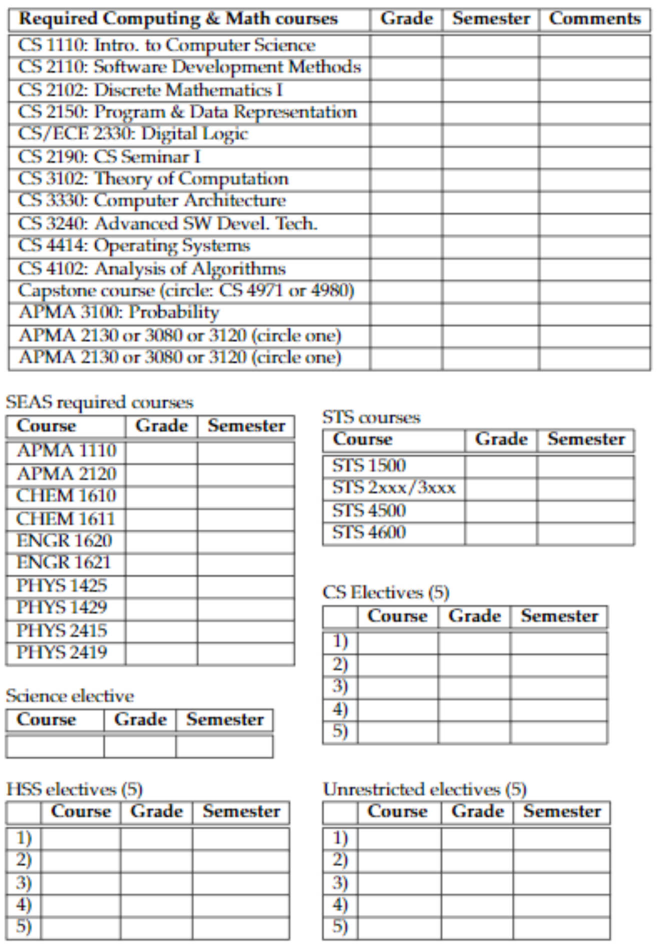
The research conducted by Conde proposed FIWARE Open-Source Initiative. FIWARE is “a framework of opensource components, called Generic Enablers (GEs), that ease the implementation of smart solutions” [2]. It focuses on “(1) the integration with third-party systems, including interfaces with IoT devices and robotics; (2) management of context data; and (3) visualization, analysis and processing of context information enabling the smart behavior of applications” [2]. This allows students to be able to “develop the necessary skills to contribute to an open-source project” [2]. This highlights a proposed class structure that helps students gain a better understanding of industry goals.

Radermacher (2012) identified and analyzed the factors for knowledge differences in newly graduated students. He discovered that “students lack proficiency with software development tools, knowledge of software testing, and teamwork and communication skills” [4]. This research gives insight into where students feel they are lacking so universities can better prepare their students for these skill gaps. Rademacher identified knowledge deficiencies in students. By knowing these deficiencies, we can change the curricula to narrow these gaps.

3. UVA Course Evaluation

The technical paper will detail suggestions for improving the current computer science curriculum, discuss a transition in the core focus of the program as well as review the strengths of the courses. An evaluation and suggested improvements of computer science classes at UVA will be presented on discussed are Discreet Math, Software Development Methods, Algorithms, and Advanced Software Development Methods.

3.1 UVA Course Load



Required Computing & Math courses	Grade	Semester	Comments
CS 1110: Intro. to Computer Science			
CS 2110: Software Development Methods			
CS 2102: Discrete Mathematics I			
CS 2150: Program & Data Representation			
CS/ECE 2330: Digital Logic			
CS 2190: CS Seminar I			
CS 3102: Theory of Computation			
CS 3330: Computer Architecture			
CS 3240: Advanced SW Devel. Tech.			
CS 4414: Operating Systems			
CS 4102: Analysis of Algorithms			
Capstone course (circle: CS 4971 or 4980)			
APMA 3100: Probability			
APMA 2130 or 3080 or 3120 (circle one)			
APMA 2130 or 3080 or 3120 (circle one)			

SEAS required courses

Course	Grade	Semester
APMA 1110		
APMA 2120		
CHEM 1610		
CHEM 1611		
ENGR 1620		
ENGR 1621		
PHYS 1425		
PHYS 1429		
PHYS 2415		
PHYS 2419		

STS courses

Course	Grade	Semester
STS 1500		
STS 2xxx/3xxx		
STS 4500		
STS 4600		

CS Electives (5)

	Course	Grade	Semester
1)			
2)			
3)			
4)			
5)			

Science elective

Course	Grade	Semester

HSS electives (5)

	Course	Grade	Semester
1)			
2)			
3)			
4)			
5)			

Unrestricted electives (5)

	Course	Grade	Semester
1)			
2)			
3)			
4)			
5)			

Table 1. Required BSCS Degree Requirements

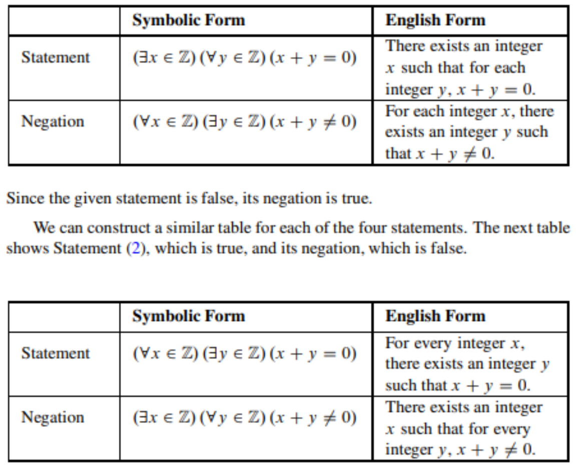
Table 1 depicts the UVA BSCS undergraduate requirements. To complete an undergraduate degree in computer science, students take at least 126 credits. The bulk of these required credits come from science, humanities, and math courses. There are 11 required computing classes, leaving students only 10 unrestricted electives. Students are unable to explore different branches of computer science fields and programs.

Most students graduate in four years making it difficult for students to expose themselves to a variety of classes. The UVA curriculum prioritizes a strong foundation in theoretical and general education, which allows students to gain strong footing in basic programming regardless of prior background. However, it also inhibits students from exploring interested CS topics, which communicates their curiosity and readiness to future employers. The new UVA curriculum should allow for more flexibility for BSCS majors.

3.2 Discrete Math (CS 2102)

The objective of this course was to "introduce students to discrete mathematical structures, including constructive logic, formal and informal proof construction and aspects of computational complexity, and some of their applications." Discreet helps students form problem-solving techniques and better understand algorithms.

This class struggled to develop student understanding of proof construction. The main problem was the use of Lean, "an interactive theorem prover" and programming language. With Lean, CS2102 became a functional programming class rather than a proofs class. Using this software over-complicated learning. Not only do students have to learn logic, construct proofs, and understand them, but also understand how to work with this new language and its toolsets.



	Symbolic Form	**English Form**
Statement	$(\exists x \in \mathbb{Z})\,(\forall y \in \mathbb{Z})\,(x + y = 0)$	There exists an integer x such that for each integer y, $x + y = 0$.
Negation	$(\forall x \in \mathbb{Z})\,(\exists y \in \mathbb{Z})\,(x + y \neq 0)$	For each integer x, there exists an integer y such that $x + y \neq 0$.

Since the given statement is false, its negation is true.

We can construct a similar table for each of the four statements. The next table shows Statement (2), which is true, and its negation, which is false.

	Symbolic Form	**English Form**
Statement	$(\forall x \in \mathbb{Z})\,(\exists y \in \mathbb{Z})\,(x + y = 0)$	For every integer x, there exists an integer y such that $x + y = 0$.
Negation	$(\exists x \in \mathbb{Z})\,(\forall y \in \mathbb{Z})\,(x + y \neq 0)$	There exists an integer x such that for every integer y, $x + y \neq 0$.

Figure 1. Screenshot From Math3000 Textbook

Learning proofs taught in Figure 1 was more effective because it uses both the symbolic form and the English form. The traditional method promotes understanding of logic. Lean resulted in program bugs, increasing time spent learning syntax and code rather than logic behind these theorems. This course can be improved by relying on a textbook and the traditional method of teaching mathematical proofs. Learning logic with Lean caused unnecessary time debugging code rather than learning what proofs we are trying to solve. To maintain an active learning style, professors can utilize the flipped classroom method where students learn topics outside of class through lecture videos and do practice problems in class.

The class helps simplify logic functions when coding. For internship projects, discreet math techniques create cleaner code by simplifying logic

3.3 Software Development Methods (CS2110)

CS2110 was one of the most impactful. This class focused on teaching students JAVA and basic programming principles. Since I had limited coding experience I took this class, CS2110 was an excellent introductory course that gives students a chance to build a strong foundation in JAVA. During my internship, I learned JAVASCRIPT and Ruby, but because of my strong foundation in coding, transitioning to these new languages was easier.

Although CS2110 succeeds in teaching basics in programming, it can be improved with better homework and project application. The homework reinforces in-class topics, but it struggles to combine topics together. For example, with threading, the homework on concurrency and locks felt more separate from other topics we learned in class. I understand how threads and concurrency work, but not how to use it in my personal projects.

An end-of-the-year group project will allow students to learn how to "adapt their knowledge across" [2] different areas. Students gain a better understanding of implementation methods for problems they want to solve. An end-of-the-semester project inspired by company internship projects help students reach a better understanding of "issues facing practitioners" [2] and expectations for their future careers. Additionally, teachers will be able to see what topics students struggle with and which aspects of class can be enhanced. Students can add projects that are in line with industry standards on their resume, revealing their development skills to future employers.

3.4 Algorithms (CS4102)

Algorithms prepared me for job interviews. This class taught problem solving techniques and ways to optimize these solutions. Technical interviews cover many different programming concepts with a strong focus on algorithms and data structures. In these interviews, given a problem description and input examples, I was asked to implement an optimal solution

CS4102 teaches various problem-solving methods and time and space complexity for each solution. Topics covered in class showed up constantly in job interviews. Algorithms can be improved through more implementation-based homework. The homework relies on pseudo-code, but since coding interviews require candidates to implement the solution as well as discuss it, there should be lab time or more implementation-based assignments.

3.5 Advanced Software Development Methods (CS3240)

This class best stimulated my experience during my internship. Over the Summer, I had to create a website based on client needs. My team tasked me with interviewing engineers about requirements for a server search feature. Through these interviews, requirements were developed, and a server design was created before implementation. We went through several sprint product development phases and several review sessions before I finished developing the product. The steps taken during my internship were replicated in this class. This class tries to narrow the "several common knowledge deficiencies" in students such as "software testing, programming ability, teamwork, oral communication, written communication, requirements gathering, analysis, problem-solving ability, software design, project management, user interface design and configuration management" [6].

A common problem in project-based classes is poor group creation. Differing motivation and effort levels among team members cause tensions and unbalanced workloads. Group composition affected my group projects and learning experience. Utilizing more descriptive surveys helps teachers compose a better group. Question about prior knowledge experience and skills, motivation, and personality are needed to build an effective team. Additionally, considering students' familiarity with each other will allow for more effective group work and communication. Ensuring accountability through peer and self evaluations during each phase of the project will act as an incentive for unmotivated students.

Another problem with the class was the assignments of specific roles: Scrum Master, Requirements Manager, Testing Manager, DevOps, UX designer. Although roles divide work, it does not encourage students to learn about tasks other roles perform. Students end up only completing their assigned instead of learning other role requirements. Switching roles from week to week encourage students to learn the roles equally and participate throughout all aspects of software development.

My manager often tasked me with additional work such as software testing outside of my project, and I found this change beneficial Learning about wireframes and understanding user interaction was emphasized at the workplace. A stronger focus on good design composition should be an added topic to the class.

4. Results

With these class proposals and recommended changes, I hope that students will feel more prepared for real-world opportunities. The increased flexibility in curricula allows students to explore interests more, leading to meaningful project development. Students will have an easier time talking about their coding skills during interviews and have a more full-bodied resume.

From the added implementation and coding portion in algorithms, there is an expected significant increase in students' ability to solve coding interview questions. The students should be able to better explain their code and reach the optimal solution faster than students in previous algorithms classes. Through these circular changes, students will be able to learn more, design more satisfying programming projects, and better prepare for real-world opportunities.

5. Conclusions

A more competitive job market stresses the importance of a full-bodied resume and interview skills. Many newly graduated students fail to meet the company's expectations due to a lack of experience. To narrow these gaps in knowledge, the UVA curriculum should become more flexible to allow students to explore specialized skills and topics. Courses in the UVA system can be improved through added group work, improving assignments, and course structure.

6. Future Work

Future work should focus on further methods to reduce this academia-to-industry gap. There are still many approaches to prepare students for their real-world careers. A way to reduce this is for teachers, students, and businesses to work together to make students and teachers more aware of the skills that are most vital in the workplace and the quality of projects expected by employers. After the implementation of these changes, data analysis can be conducted to see if these changes produced a significant impact on student employment rate and career readiness.

REFERENCES

[1] BBC. (n.d.). Why inexperienced workers can't get entry-level jobs. BBC Worklife. Retrieved

November 21, 2021, from https://www.bbc.com/worklife/article/20210916-whyinexperienced-workers-cant-get-entry-level-jobs.

[2] Conde, J.; López-Pernas, S.; Pozo, A.; Munoz-Arcentales, A.; Huecas, G.; Alonso, Á. Bridging the Gap between Academia and Industry through Students’ Contributions to the FIWARE European Open-Source Initiative: A Pilot Study. Electronics 2021, 10, 1523. https://doi.org/ 10.3390/electronics10131523

[3] Palacios, A., 2022. Computer Science Internships Are Important.. Here's Why.. [online] Linkedin.com. Available at: <https://www.linkedin.com/pulse/computer-science-internships-important-heres-why-alejandro-palacios> [Accessed 23 February 2022].

[4] Radermacher, A., 2022. EVALUATING THE GAP BETWEEN THE SKILLS AND ABILITIES OF SENIOR UNDERGRADUATE COMPUTER SCIENCE STUDENTS AND THE EXPECTATION OF INDUSTRY. [online] Library.ndsu.edu. Available at: <https://library.ndsu.edu/ir/bitstream/handle/10365/26744/Evaluating%20the%20Gap%20between%20the%20Skills%20and%20Abilities%20of%20Senior%20Undergraduate%20Computer%20Science%20Students%20and%20the%20Expectations%20of%20Industry.pdf?sequence=1&isAllowed=y> [Accessed 23 February 2022].or (Ed.). 2007. The title of book one (1st. ed.). The name of the series one, Vol. 9. University of Chicago Press, Chicago. DOI:https://doi.org/10.1007/3-540-09237-4.

[5] The Sass Way. 2022. What Is The Unemployment Rate For Computer Science Majors?. [online] Available at: <https://thesassway.com/what-is-the-unemployment-rate-for-computer-science-majors/> [Accessed 23 February 2022].

[6] Radermacher, A. and Walia, G., 2013. Gaps between industry expectations and the abilities of graduates. Proceeding of the 44th ACM technical symposium on Computer science education - SIGCSE '13,.

[7] Teimzit Amira, Mahnane Lamia, and Hafidi Mohamed. 2019. Flipped classroom for algorithmic teaching. In Proceedings of the 2nd International Conference on Networking, Information Systems & Security (NISS19). Association for Computing Machinery, New York, NY, USA, Article 59, 1–4. DOI:https://doi.org/10.1145/3320326.3320393