A Redesign of the *Meals on Wheels of Charlottesville* Volunteer Portal
(Technical Paper)

A Curricular Analysis of Computer Science at the University of Virginia
(STS Paper)

A Thesis Prospectus Submitted to the

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

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

Michael Benos
Fall 2019

Technical Project Team Members
Alex Hicks
Kyle Leisure
Kevin Naddoni
Maxwell Patek
Joshua Santana
Nathanael Strawser

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.

Advisor
Dr. Ahmed Ibrahim, Department of Computer Science

Advisor
Kent Wayland, Department of Engineering and Society

General Research Problem

Software engineers must have a solid educational foundation in computer science (CS) before being able to write software effectively. At the University of Virginia (UVA), most computer science students must engineer a real-world piece of software for a client in order to graduate, and the success of these projects is predicated upon completing most of the courses in the UVA CS curriculum. Additionally, many CS students go on to develop software professionally. As a result, the CS curriculum, as part of the UVA engineering program, must adequately prepare students to apply CS fundamentals to software development. This prospectus will explore a technical application project focusing on Charlottesville *Meals on Wheels* and will present a science, technology, and society (STS) research topic the state of the UVA CS curriculum, aiming to better understand how recent changes to teaching methods and content work toward constructing an ideal curriculum. By connecting an application-based project with a research-driven analysis, this prospectus will connect the two in order to draw conclusions regarding how successful the UVA CS curriculum's efforts have been to survey the landscape of the rapidly developing field.

Technical Research Problem: Improving and Future-proofing the Volunteer Portal of the Charlottesville Chapter of Meals on Wheels

How should the current system be changed to address shortcomings and optimize the portal's future performance?

Despite America being one of the richest countries in the world, an estimated eight million of its aging citizens face the threat of hunger (World Bank, 2019; NCOA, 2015). Meals on Wheels is America's oldest and largest organization dedicated to mitigating this issue through community chapters (MOWA, 2019). The non-profit's local chapter delivers meals to disabled or

elderly people in the Charlottesville-Albemarle area who cannot cook or buy food themselves. With the help of volunteers, the organization packs, labels, and distributes meals to customers via various delivery routes. In addition, volunteers drive a few shuttle routes to deliver meals to locations outside of the Charlottesville-Albemarle area (A. Dudley, personal communication, September 27, 2019).

A small team of paid staff administers Charlottesville's Meals on Wheels chapter by managing delivery routes, maintaining current and prospective customer information, and ensuring that all daily jobs are filled by at least one volunteer (S. Bayker, personal communication, September 13, 2019). These administrative tasks can get rather complex due to a combination of daily, weekly, biweekly, monthly, and one-time volunteer shifts and customer needs. Also, most of the meals delivered are sourced via donation, which makes predicting supply difficult.

The greater U.S. Meals on Wheels organization sells professional software to help staff manage the complexity of their tasks; however, the Charlottesville office cannot afford it (A. Dudley, personal communication, October 11, 2019). Thus, staff managed volunteers, customers, and routes by hand until approximately three years ago, when a University of Virginia computer science capstone team created a web portal for them. Adopting this web portal gave Meals on Wheels' staff more time to focus on essential tasks by automating physical reports and tedious manual tasks.

A subsequent capstone team updated the web portal to its current state, but Meals on Wheels' staff still desires improvements. First, staff complained that the web application has become increasingly slow over time. After examining the current codebase, we believe this slowness is likely due to its cluttered, unclear data storage and the use of an inexpensive hosting

solution. Second, staff identified several organizational oddities within the app layout, making some tasks take longer than necessary. Finally, staff requested the addition of new features, including historical report generation and general search functionality.

It is clear that the system needs an update; however, the technical debt[1] accumulated by the separate capstone teams developing features over a two-year period necessitates a rewrite. Our capstone team's goal, therefore, is to write a new application that satisfies Meals on Wheels' needs and has a more reasonable and maintainable backend for long-term deployment, including state-of-the-art modularity via Docker, normalized database models[2], and cost-effective cloud deployment via Amazon Web Services. By redesigning and modernizing from the ground up, our project should enable Meals on Wheels to operate at lower costs and function more quickly; the organization should have more time and money to help customers in need. We aim to provide a minimum viable product including features such as account creation, assigning volunteers to routes, and volunteer substitution by the end of the current year. We plan on releasing the complete, working product by May 2020.

Requirements for a Minimum Viable Product

- All Users
    - As a user, I should be able to create my own account (including custom username), so I can log in and see personalized information.
    - As a user, I should be able to request to change my password in case I forget it.

[1] Technical debt is "when engineers take shortcuts that fall short of best practice" (Allman, 2012).

[2] Normalization is the process of organizing a database to eliminate redundancy and inconsistent dependency (Microsoft, 2017).

- Volunteers
    - As a volunteer, I should be able to release my route on a day, so someone else can substitute for that job.
    - As a volunteer, I should be able to pick up a released route on a particular day, so no routes go without a volunteer.
    - As a volunteer, I should be able to pick up a new route that has not been assigned to any volunteer, so I can plan my hours in advance.
- Staff
    - As staff, I should be able to create clients, so I can accommodate a growing client base.
    - As staff, I should be able to generate reports, so I can prepare daily operations.
    - As staff, I should be able to manually create delivery routes, so I can customize the volunteer's tasks.
    - As staff, I should be able to manually delete delivery routes, so I can avoid cluttering the portal with unused routes.
    - As staff, I should be able to assign volunteers to recurring routes, so I can plan delivery.
    - As staff, I should be able to substitute one-time volunteers for jobs, so I can ensure that all necessary jobs are filled.
    - As staff, I should be able to release volunteers from their recurring routes, so I can assign another volunteer to the recurring route.
    - As staff, I should be able to one-time release volunteers from their routes, so I can allow other volunteers to substitute.

- As staff, I should be able to print reports that have been generated by any staff, so can have physical report copies.
- As staff, I should be able to see who is volunteering on a particular day, so I can stay organized and communicate as necessary.

STS Research Problem: An Assessment of the Practical Applicability of the UVA CS Program

How do the recent changes in the UVA CS curriculum achieve a more modern approach to teaching computer science?

The UVA CS curriculum has recently undergone drastic changes. A pilot program was unrolled in 2017, providing an alternative path to graduation for students. This new pilot program has restructured the old curriculum almost entirely, removing old course requirements and replacing them with fewer courses. We need to know how the slimmed down and modified content, as well as whether the practical applications in the classroom, are actually helping students to learn effectively and develop a thorough foundation in computer science. This will involve delving into the differences between the old and new curricula, analyzing the depth and breadth of topics covered, in an attempt to understand how the changes are adapting the curriculum to better explore new developments in the rapidly changing field. This is significant because upon graduation, students must be prepared to take the next step to either graduate level studies, or to apply their knowledge to real-world problems, both of which place students in a situation where they must have the ability to comprehend the state-of-the-art, not just the antiquated, components of the field.

Background and Theoretical Framework

The most important groups to investigate are students, faculty, and administrators who experienced the CS program prior to the implementation of the pilot program, as well as the students, faculty, and administrators who have experience with the program since it underwent changes. Investigating student perspectives and comparing them across time will likely be most interesting. This will help to comprehend the current state of and sentiments toward the new

curriculum, as well as give a more nuanced understanding of how the new and old curricula differ. The differences that this research seeks are less focused on the conspicuous changes (e.g. the fact that there are fewer courses) and more so on the subtleties of how the new curriculum is embracing modern developments in the CS field.

In addition to investigating stakeholders, literature will be explored relating to the past, present and future states of computer science, and how these are represented in the UVA current and pilot curricula. One element of the literature review will be to research computer science fundamentals, such as those described by Proulx et al., to benchmark the old and pilot UVA curricula. Another element of the literature review will include synthesizing the various teaching methods currently utilized in computer science education. When learning new skills in general, there is evidence that a greater focus on practical application can help students not only learn effectively, but also better translate complex and abstract topics to other real-world challenges (Basawapatna, 2010). As a result, some methods are potentially superior to others, such as constructivism, which is "theory of learning which claims that students construct knowledge rather than merely receive and store knowledge transmitted by the teacher" (Ben-Ari, 2001). Despite evidence that constructivism is an effective teaching methodology, especially in STEM fields, it is largely lacking from the old UVA CS curriculum. The old curriculum relies heavily on lecture-based methods to communicate information to students, while the methods of the pilot program are unknown. The pilot program's methods will be investigated by inspecting its coursework, and then will be compared to those of other reputable CS programs such as that of Carnegie Mellon University to identify the UVA curriculum's strengths and weaknesses. Overall, an in-depth literature review on CS curricula, effective teaching and the history of CS will be included as part of the final project.

Evidence and Data Collection

In terms of evidence and data collection, this STS research problem will collect data mainly via interviews and surveys with UVA CS students and faculty. Other stakeholders to be examined could include administration, other professors, and previous students at UVA, as well as accreditation boards (ABET), education researchers, other universities, national societies, and employers of graduates. Interviews will target established UVA CS professors who have had critical roles in the creation of the new pilot program, with the goal of understanding the benefits of new content or structure that the pilot program provides. The surveys may ask broad questions about how effectively the curriculum is preparing students, what the general response has been to the recent trimming down of the content in the curriculum, and topics that were and were not covered adequately by the curriculum. When taken together, the interviews, surveys and literature review hopefully will shed light on the strengths and weaknesses of the UVA CS curriculum.

Overall, this STS project will help analyze the efficacy of the UVA CS curriculum, as well as present an overview of how the UVA CS curriculum has evolved over time and the direction that the program can take for the future. Given the ever-changing nature of the field, this research is significant in helping answer the broader inquiry about whether the CS program has evolved in a direction that best prepares students.

Conclusion

The combination of the technical and STS research problems unites practice and theory. The technical portion will be an implementation-based project to improve the nonprofit *Meals on Wheels Charlottesville* volunteer portal, while the STS research portion will focus on the evolution of the UVA CS curriculum. Although distinct, an in-depth study of each topic will

hopefully illuminate how UVA has approached modernizing its CS curriculum, with the ultimate objective of better understanding the CS department's future goals.

References

Allman, E. (2012). Managing technical debt. Communications of the ACM, 55 (5), 50. https://doi.org/10.1145/2160718.2160733

Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010). Using scalable game design to teach computer science from middle school to graduate school. *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education - ITiCSE* '10, 224–228. https://doi.org/10.1145/1822090.1822154

Ben-Ari, M. (2001). Constructivism in Computer Science Education. *Journal of Computers in Mathematics and Science Teaching*, 20(1), 45-73.

Carnegie Mellon University, Computer Science Department. (2018, November). Bachelors Curriculum - Admitted 2018. Retrieved September 27, 2019, from https://csd.cs.cmu.edu/academic/undergraduate/bachelors-curriculum-admitted-2018

Kirschner, P. A., Sweller, J., & Clark, R. E. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist*, 41(2), 75–86.

Microsoft (2017). Description of the database normalization basics. https://support.microsoft.com/en-us/help/283878/description-of-the-database-normalizationbasics

MOWA. (2019). Meals on Wheels America. National Office. https://www.mealsonwheelsamerica.org/learn-more/national

NCOA. (2015, June 4). National Council On Aging. Facts About Senior Hunger. https://www.ncoa.org/news/resources-for-reporters/get-the-facts/senior-hunger-facts/

World Bank (2019). GDP per capita. https://data.worldbank.org/indicator/ny.gdp.pcap.cd

Proulx, V. K., Rasala, R., & Fell, H. (n.d.). Foundations of Computer Science: What are they and how do we teach them? [Scholarly Publication, Northeastern University]. Retrieved from http://www.ccs.neu.edu/home/vkp/Papers/Foundations-iticse96.pdf