Open Source Software Practices in CS2
(Technical Paper)

Ethics Education for Responsible Computer Scientists
(STS Paper)

A Thesis Prospectus Submitted to the

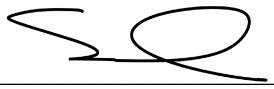
Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

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

Emma Choi
Fall, 2021

Technical Project Team Member
Lisa Meng

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

Signature  Date 12/12/2021
Emma Choi

Approved _____ Date _____
John R. Hott, Department of Computer Science

Approved _____ Date _____
Richard D. Jacques, Ph.D., Department of Engineering and Society

Introduction

The modern-day stereotype of a computer science (CS) student invokes an image of an antisocial male furiously typing cryptic codes to build the next Facebook. Outsiders of the discipline often assume that CS is full of zealous computer nerds with excessive workloads and mundane programming assignments. This rigid outlook deters and drives away bright students with a sense of detachment, alienation, and disillusionment (Biggers et al., 2008). Especially female and minority students, already underrepresented in STEM fields, are negatively affected by those factors (Fisher et al., 1997). In fact, the traditional CS curriculum perpetuates harmful stereotypes and fails to prepare students for their responsibilities as socially conscious engineers. Instead, it should support, engage, motivate, and inspire students to become capable problem-solvers in the real-world.

Educators should integrate realistic contexts in lessons to provide a well-rounded curriculum instead of assigning exercises that regurgitate technical concepts. According to Layman et al., students can gain insights and aspirations on how theoretical knowledge can be applied to solve practical problems in society through meaningful assignments (2007). More specifically, the "context—the area outside direct computer science, or the circumstances surrounding the technical course content—would provide a source of examples, motivations, and project ideas throughout the course" (Cooper & Cunningham, 2010, p. 5).

In order to realize this goal, the technical research will address how open source software projects can be integrated into lower-level CS courses. Depending on classroom needs, I will outline different strategies for instructors to help students gain experience in solving practical problems. I will suggest concrete plans of action to illustrate real-world software development

practices, exercise teamwork, and interact with professional communities. On the other hand, the STS research will explore a broader reformation of CS as a discipline. I will investigate how contextual learning has the potential to expand students' worldview as socially conscious engineers beyond technical accomplishments.

Technical Topic

Traditional lectures and impractical programming assignments often fail to capture the interests of students. Many students who change majors from CS describe the discipline as unrewarding, endlessly coding/debugging, lacking real-world utility, and irrelevant to other fields (Biggers et al., 2008). In response, Cooper and Cunningham emphasize the importance of contextual teaching to motivate, challenge, and engage students (2010). More specifically, a context-based CS course would introduce the environment in which a technical topic exists. Besides fundamental principles, students would engage with their applications, possibly in fields outside of CS, and find connections to different communities.

Layman et al. propose adopting more meaningful assignments with real-world utility or power to improve society (2007). One such implementation is integrating Open source software (OSS) projects in classrooms. OSS are specially licensed products with publicly available source code that can be further built upon and distributed by other developers. Open source communities, comprising of programmers, project managers, designers, documentation writers, etc., support collaboration within software development (Red Hat, n.d.). Gokhale et al. present a case study from a software engineering course where students contributed to OSS (2013). As students had to analyze existing software, complete with an established user base and a history of modifications by other community contributors, they gained a first-hand understanding of quality

architecture, documentation, and maintenance practices. By interacting with real-world code, they were able to learn from contextualized examples.

Students who participate in OSS projects as part of similar software engineering courses often report benefitting from developing technical and social skills, interacting with the wider open source community, gaining confidence, and upgrading their portfolios (Pinto et al., 2019). However, incorporating OSS projects into classrooms poses many logistical difficulties for instructors. For example, instructors face the challenge of filtering through millions of OSS available on online repositories to find projects that are appropriate for the size, level, and curriculum of courses. Moreover, researchers report that not every CS class can support OSS projects (Gokhale et al., 2012). Without prior knowledge of software engineering and relevant tools/frameworks, novice students can be overwhelmed with learning both course content as well as how to contribute to OSS. Therefore, Postner et al. suggest that instructors may find software engineering and capstone courses most suitable to integrate such projects (2019).

In response to the aforementioned obstacles restricting the diverse learning opportunities that students can acquire from OSS, my technical research will investigate methods of introducing OSS concepts into lower-level CS classes. While other research mainly focuses on projects for software engineering and senior capstone courses, Biggers et al. emphasize the need for interactive and engaging introductory classes in order to retain students (2008). Therefore, I will specifically target fundamental data structures and algorithms courses and provide guidelines for meaningful assignments with varying degrees of interaction with open source communities. With a research partner and an advising CS professor, I will design a rubric to help instructors determine how to incorporate existing projects or open source development practices

depending specific classroom needs. The suggested curriculums will be published for free online, in the true spirit of open source movement.

STS Topic

According to Cech, engineers discover their roles and responsibilities as part of their education in a process called professional socialization (2014). Due to their technical expertise, they are uniquely able to reflect on the ethical and social impact of their work. However, the author argues that higher institutions often fail to inspire consideration for public welfare in students by upholding a culture of disengagement. More specifically, students learn to value technical skills and disregard social implications of technology. Especially within the field of CS, students commonly esteem accomplishments and programming knowledge over creative reasoning and intellectual exploration of ideas (Lewis et al., 2010). Those who are disenchanted by this rigid and competitive attitude often quit without recognizing their potential to change the world as computer scientists (Biggers et al., 2008).

Toward the goal of providing a more well-rounded education bridging the study of technology with social sciences, curriculums should frame the big picture of CS and establish its utility in the real world. Instructors can demonstrate theoretical concepts in action by introducing specific examples of their implementations. For instance, instead of using sorting algorithms to arrange a random list of numbers in order, they can challenge students to queue patients waiting to receive vaccines based on age, exposure risk, and medical history. Fisher et al. hypothesize that women especially value computers as a "tool to use within a broader context of education, medicine, communication, art and music" (1997, p. 108). Then perhaps more marginalized students will be motivated to study CS when they realize how computing can be pertinent to

diverse fields while engaging with meaningful assignments. On that account, I will explore how educators can adopt context-based teaching to train empowered and mindful computer scientists.

At Purdue University, engineers can fulfill community service by providing modern technical solutions to nonprofit organizations. The program demonstrates a productive example of how students and communities can mutually benefit by working together. Through collaboration, students can practice theoretical knowledge learned in classrooms in the context of real-world problems within their community. Most importantly, they learn "many valuable lessons in citizenship, including the role of community service in our society; the significant impact that their engineering skills can have on their community; and that assisting others leads to their own substantial growth as individuals, as engineers, and as citizens" (Coyle et al., 2005, p. 3).

Alternatively, instructors do not need to coordinate complex projects with external communities in order to provide meaningful contexts. In fact, any programming assignment can accommodate practical or socially relevant context. For example, instructors can provide template code, hiding complex details, so that beginner programmers can build valuable software regardless of their lack of experience. Or, they can reword problem statements and variable names of assignments in order to demonstrate potential applicability of programming concepts toward real-world problems (Layman et al., 2007).

Even the smallest change in existing assignments can inspire CS students to recognize their potential to serve and improve the greater society. As part of my STS research, I will apply the Actor Network Theory (ANT) in order to comprehend how instructors can introduce contextual learning to transform the network of CS education and train students to become

socially conscious computer scientists. I will analyze how different actors affect students' development during higher education with evidence from case studies, which have been published for CS education conferences, that highlight different strategies of context-based teaching.

Conclusion

Computer science as a discipline should foster students' goals to serve as impactful engineers. Layman et al. suggest that meaningful assignments, with real-world contexts, can better engage students and develop their worldview (2007). Further building upon their idea, I will investigate how contextual learning can be realistically implemented in classrooms and how it can transform the culture of CS education. The technical research will recommend plans of action, in the form of rubrics and curriculums, for instructors to introduce open software practices to novice students. On a broader scope, the STS research will analyze how a reformed CS curriculum can help students better connect with the wider world. Overall, I will explore how instructors can better provide a well-rounded experience, balancing technical concepts with social studies, and mentor future generations of responsible engineers.

References

Biggers, M., Brauer, A., & Yilmaz, T. (2008). Student perceptions of computer science. *ACM SIGCSE Bulletin*, *40*(1), 402–406. https://doi.org/10.1145/1352322.1352274

Cech, E. A. (2014). Culture of disengagement in engineering education? *Science, Technology, & Human Values*, *39*(1), 42–72. https://doi.org/10.1177/0162243913504305

Cooper, S., & Cunningham, S. (2010). Teaching computer science in context. *ACM Inroads*, *1*(1), 5–8. https://doi.org/10.1145/1721933.1721934

Coyle, E. J., Jamieson, L. H., & Oakes, W. C. (2005). EPICS: Engineering Projects in Community Service. *International Journal of Engineering Education*, *21*(1), 139–150. https://www.ijee.ie/articles/Vol21-1/IJEE1549.pdf

Fisher, A., Margolis, J., & Miller, F. (1997). Undergraduate women in computer science. *Proceedings of the Twenty-Eighth SIGCSE Technical Symposium on Computer Science Education - SIGCSE '97*, 106–110. https://doi.org/10.1145/268084.268127

Gokhale, S. S., Smith, T., & McCartney, R. (2012). Integrating open source software into software engineering curriculum: Challenges in selecting projects. *2012 First International Workshop on Software Engineering Education Based on Real-World Experiences (EduRex)*, 9–12. https://doi.org/10.1109/edurex.2012.6225697

Gokhale, S., Smith, T., & McCartney, R. (2013). Teaching software maintenance with open source software: Experiences and lessons. *2013 IEEE Frontiers in Education Conference (FIE)*, 1664–1670. https://doi.org/10.1109/fie.2013.6685121

Layman, L., Williams, L., & Slaten, K. (2007). Note to self. *Proceedinds of the 38th SIGCSE Technical Symposium on Computer Science Education - SIGCSE '07*, 459–463. https://doi.org/10.1145/1227310.1227466

Lewis, C., Jackson, M. H., & Waite, W. M. (2010). Student and faculty attitudes and beliefs about computer science. *Communications of the ACM*, *53*(5), 78–85. https://doi.org/10.1145/1735223.1735244

Pinto, G., Ferreira, C., Souza, C., Steinmacher, I., & Meirelles, P. (2019). Training Software Engineers Using Open-Source Software: The Students' Perspective. *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training (ICSE-SEET)*, 147–157. https://doi.org/10.1109/icse-seet.2019.00024

Rosmaita, B. J. (2007). Making service learning accessible to computer scientists. Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education - SIGCSE '07, 541–545. https://doi.org/10.1145/1227310.1227493

Red Hat. (n.d.). *What is open source software?* Retrieved April 5, 2021, from https://www.redhat.com/en/topics/open-source/what-is-open-source-software