Ethics Education for Responsible Computer Scientists

An STS 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

Emma Choi

Fall Semester 2021

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

Date 12/12/2021

Emma Choi

Date 12/07/2021

Richard D. Jacques, Ph.D., Department of Engineering and Society

Introduction

Computer science (CS) is a versatile discipline that can empower students to realize the unlimited potential of its application in every domain of knowledge. However, while it brings visionary technology from science fiction closer to reality, it also gives rise to dystopian nightmares where privacy is relinquished for security or malicious artificial intelligence programs wreak havoc. Therefore, students must learn to reflect on the ethical implications of their work and commit to uphold moral responsibilities. In order to achieve this goal, CS departments can introduce a comprehensive ethics education and implement contextual teaching with projects involving local communities and meaningful assignments. As a result, a reformed curriculum will better engage, inspire, and prepare future generations of socially conscious computer scientists.

Part I: A Culture of Disengagement

There is no doubt that computer technology is rapidly shaping our world and software engineers are at the forefront of the digital revolution. Thus, in order to ensure that this profession remains beneficial and respected, the joint task force on Software Engineering Ethics and Professional Practices, established by the Association for Computing Machinery (ACM) and the Institute for Electrical and Electronics Engineers Computer Society (IEEE-CS), promotes a code of ethics and professional practice. Its eight principles describe how engineers should uphold public interest and welfare while maintaining professional standards as employees, managers, colleagues, and lifelong learners. According to the task force, such codes of conduct should empower and guide members through practical challenges (Gotterbarn et al., 1997).

While well-intentioned and promising, the extent of its application in the real world is more dubious.

Oftentimes, technology companies also define a code of conduct, promising the public that they will bear social responsibility and make a positive contribution to society. Keep building better. Reimagine a better future. However, the language of corporate responsibility is often distorted with a heavy accent of self-interest. Whether in the form of an ethics advisory board, employee training on ethical behavior, or a strategically formulated code of ethics, companies use ethics as a "façade that justifies deregulation, self-regulation or market driven governance". The term, "ethics washing", refers to similar acts of trivializing the merits of ethical work by focusing on its desired results alone (Bietti, 2020). This narrow reasoning hinders productive discussions on upholding moral principles in tech industries.

Ethics is the reflection on worldviews and applications of moral values. Due to widely varying human experiences and cultures, a "universal" code of ethics cannot account for every possible moral dilemma nor will everyone interpret or apply it in a similar manner. Therefore, Burton et al. assert that "the idea that a single code of laws or duties would solve all problems, and that our responsibility as teachers is to transmit those laws to students, is appealing but ultimately false" (2018, p. 56).

Students discover the roles and responsibilities of successful engineers as part of their education in a process called professional socialization (Cech, 2014). However, if the professional society undervalues responsibility and accountability to the public, similar oversight can be observed in higher education institutions. Computer science, like other STEM fields, is a discipline with a robust practical domain and a prevailing set of technical concepts. Thus, instructors often resort to a transmission of absolute facts rather than encouraging creative

reasoning (Burton et al., 2018). In fact, based on a survey given to faculty members and students, Lewis et al. discovered that engineers commonly esteem technical skills and accomplishments more than explorations of ideas and discussions on the social impacts of technology (2010). This cultural emphasis on outcomes leads to several disadvantages. It can "condition [students] to interpret what they learn in terms of an authority-based view of 'truth' that in turn leaves them unequipped to reason about situations involving no single correct answer or think cogently about ethical trade-offs" (Burton et al., 2018, p. 56). It also upholds a culture of disengagement where a CS curriculum fails to cultivate a consideration for public welfare in students (Cech, 2014). Those who are disenchanted by this rigid and competitive attitude often abandon their studies without recognizing the potential to impart a positive influence on the world as computer scientists (Biggers et al., 2008).

Ethics is also a mode of inquiry for the purpose of gaining a deeper understanding of sociotechnical issues and designing better technology policies and regulations to improve the world (Bietti, 2020). It should serve as a framework to empower students to reflect on a given situation and weigh its consequences, while considering diverse viewpoints (Burton et al., 2018). In other words, instructors should not teach what to think, but how to think.

Part II: Expanding Perspectives

The Accreditation Board for Engineering and Technology (ABET) requires CS curricula to cover topics discussing "local and global impacts of computing solutions on individuals, organizations, and society" (Criteria for Accrediting Computing Programs, 2020 – 2021, n.d.). According to a survey of 50 higher education institutions, 55% of CS departments offer a computer ethics course, 30% incorporate discussions about ethical and social implications of

technology in existing CS courses, and 15% require students to take ethics courses in other departments, namely philosophy. While many of them report that designing a custom ethics course within the CS department is most efficient to fulfill ABET standards, students can gain unique insights from this approach. When CS professors teach ethics and social responsibility, they can lead more relevant discussions about real challenges faced by professions in the computing field. Moreover, "they serve as role models who demonstrate that contemplating the ethical dimensions of everyday decisions is something that everyone can and should do—not just those with a Ph.D. in philosophy" (Quinn, 2006, p. 337).

Technology cannot be considered independently from the society that adopts it for use. For this reason, Martin and Weltz advocate for a more extensive ethics education for computer scientists. Separate courses in philosophy and sociology or a perfunctory review of a general code of ethics are not enough to prepare students to address ethical issues in the real world. Instead, lessons on ethics must be embedded throughout the CS curriculum. They present a progressive path where students learn each topic incrementally, starting from recognition of social issues, evaluation of problems, to responsible action for resolution. In order to explore the interaction between technology, ethics, and society, instructors should introduce topics early, continuously lead discussions on computing ethics, and integrate them within technical course concepts (1999). Overall, such a comprehensive integration of sociotechnical analysis will equip students with relevant skills to tackle difficult challenges in the future.

As an alternative approach, researchers from Stanford University present a multidisciplinary computing ethics course with the belief that students from any discipline should consider ethical issues in a technological society. Faculty staff from computer science, philosophy, and social sciences provide diverse perspectives to "challenge students to internalize a commitment to their

responsibilities as innovators, designers, coders, engineers, policymakers, citizens, and/or consumers" (Reich et al., 2020, p. 297). Assignments ranging from code evaluation to reflective essays kindle contemplation for conflicting ethical issues and trade-offs in the context of realistic case studies, such as algorithmic bias and data privacy (Reich et al., 2020).

Part III: Practice makes perfect

Unfortunately, daunting myths and shortcomings of the CS curriculum chase away bright students from the discipline. According to a survey from Georgia Tech, negative stereotypes portray the field as programming-centric, unrewarding, and lacking real-world connections. Many introductory CS courses reinforce this demoralizing image by limiting the classroom scope to foundational programming concepts (Biggers et al., 2008). It is especially detrimental to female students who often "contextualize their interest in computer science, instead, within a larger purpose: what they can do in the world" (Fisher et al., 1997, p. 108). Other minority students report interest in socially relevant projects as well (Layman et al., 2007). As an alternative, the CS curriculum should introduce meaningful contexts in order to support and motivate the next generation of responsible engineers.

Cooper and Cunningham advocate for contextual teaching, where relevant sources or applications of technical concepts are integrated into lessons. Contexts can provide a more accessible approach to understanding course materials and prepare students to confidently employ their knowledge when faced with similar real-world challenges (2010). As a result, students gain valuable insight on how theoretical abstractions can be applied to solve practical problems with a social impact through meaningful assignments (Layman et al., 2007).

One of the most effective ways to practice social responsibility is to participate in projects that involve the community. One such implementation is to adopt open source software (OSS) projects in classrooms. OSS is specially licensed products with publicly available source code that can be further built upon and distributed by other developers. Open source communities, which may include the likes of programmers, project managers, user interface designers, and technical writers, collaborate with the selfless goal of developing quality software free for anyone to use or modify (Red Hat, n.d.). Moreover, OSS documentation, version history, pending issues, and discussions amongst contributors provide valuable insight into professional software development. Numerous case studies show that students who contribute to OSS as part of CS courses benefit from unique opportunities to improve technical and soft skills, interact with open source communities, gain confidence as accomplished developers, and elevate their resumes (Pinto et al., 2019).

At Trinity College, a course titled free and open source software (FOSS) 101 introduces students to software development principles in the context of OSS. Besides programming lessons, it facilitates discussions on the impact of the open source movement on the wider society. Furthermore, in the true spirit of the open source philosophy, the cutthroat and competitive stereotype of computer scientists is dispelled by encouraging students to collaborate and share solutions. Through the program, students utilize CS principles to give back to the community and leave a positive impact on society (Morelli & de Lanerolle, 2009).

Contributors who work on OSS designed with philanthropic goals, called humanitarian and free open source software (HFOSS), provide community service while exploring CS principles in real-world applications. Ellis et al. advocate for course assignments involving HFOSS to demonstrate the capability of computer scientists to succeed beyond the purely

technical domain and solve socially relevant problems (2007). For example, the Trinity Sahana HFOSS project, a disaster recovery coordination software, provides students an opportunity to work alongside faculty members, corporate associates, and developers across the globe. Students benefit from exploring the utility of CS in diverse applications, understanding social responsibility of engineers, and acquiring a new, positive perspective on the discipline (Hislop et al., 2009).

Another productive example of students and communities cooperating for mutual benefit is the Engineering Projects in Community Service (EPICS) Program at Purdue University. By providing modern, technical solutions for nonprofit organizations, students can fulfill community service. They gain useful insight by utilizing theoretical knowledge learned in classrooms in the context of real-world challenges within their local community. More 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. 141).

Alternatively, instructors can embed practical contexts within existing lessons in order to train socially conscious engineers. Instead of coordinating elaborate projects with external communities, they can introduce meaningful contexts within programming assignments. Instructors can provide template code, hiding complex details, so that even novice programmers can build valuable software regardless of their lack of experience. For instance, rather than writing sorting algorithms to arrange a list of numbers in order, they can challenge students to queue patients waiting to receive vaccines based on age, exposure risk, and medical history. Or, they can reword problem statements and variable names of a program to demonstrate the

applicability of technical concepts towards 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 world.

Conclusion

Due to the highly technical expertise of computer scientists, they are most capable of reflecting on the social impact of their work. However, within the culture of disengagement, they often neglect to conscientiously reflect on ethical questions. As the influence of technology on every facet of society grows, it is imperative for a reformed CS curriculum to train proficient, accountable, and responsible engineers. Burton et al. affirm that "the responsibility of an ethics instructor is to train students to engage in understanding and reasoning. The students are thus prepared to navigate situations that offer no clean solutions and engage other computer science practitioners in discussion about what and how to choose" (2018, p. 56). Thus, through well-designed ethics education, contextual teaching, and meaningful assignments, they will be empowered to contribute to society as responsible engineers.

References

Bietti, E. (2020). From ethics washing to ethics bashing: a view on tech ethics from within moral philosophy. FAT* ’20: Proceedings of the 2020 Conference on Fairfness, Accountability, and Transparency, 210–219. https://dl.acm.org/doi/abs/10.1145/3351095.3372860

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

Burton, E., Goldsmith, J., & Mattei, N. (2018). How to teach computer ethics through science fiction. Communications of the ACM, 61(8), 54–64. https://doi.org/10.1145/3154485

Cech, E. A. (2013). 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, 139–150. https://www.researchgate.net/publication/228337426_EPICS_Engineering_Projects_in_Community_Service

Criteria for accrediting computing programs, 2020 – 2021. (n.d.). ABET. Retrieved November 8, 2021, from https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-computing-programs-2020-2021/

Ellis, H. J. C., Morelli, R. A., de Lanerolle, T. R., Damon, J., & Raye, J. (2007). Can humanitarian open-source software development draw new students to CS? ACM SIGCSE Bulletin, 39(1), 551–555. https://doi.org/10.1145/1227504.1227495

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

Gotterbarn, D., Miller, K., & Rogerson, S. (1997). Software engineering code of ethics. Communications of the ACM, 40(11), 110–118. https://doi.org/10.1145/265684.265699

Hislop, G. W., Ellis, H. J., & Morelli, R. A. (2009). Evaluating student experiences in developing software for humanity. ACM SIGCSE Bulletin, 41(3), 263–267. https://doi.org/10.1145/1595496.1562959

Layman, L., Williams, L., & Slaten, K. (2007). Note to self: make assignments meaningful. ACM SIGCSE Bulletin, 39(1), 459–463. https://doi.org/10.1145/1227504.1227466

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

Martin, C. D., & Weltz, E. Y. (1999). From awareness to action: integrating ethics and social responsibility into the computer science curriculum. ACM SIGCAS Computers and Society, 29(2), 6–14. https://doi.org/10.1145/382018.382028

Morelli, R., & de Lanerolle, T. (2009). Foss 101: engaging introductory students in the open source movement. ACM SIGCSE Bulletin, 41(1), 311–315. https://doi.org/10.1145/1539024.1508977

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

Quinn, M. J. (2006). On teaching computer ethics within a computer science department. Science and Engineering Ethics, 12(2), 335–343. https://doi.org/10.1007/s11948-006-0032-9

Reich, R., Sahami, M., Weinstein, J. M., & Cohen, H. (2020). Teaching computer ethics: a deeply multidisciplinary approach. Proceedings of the 51st ACM Technical Symposium on Computer Science Education, 296–302. https://doi.org/10.1145/3328778.3366951

What is open source software? (n.d.). Red Hat. Retrieved November 8, 2021, from https://www.redhat.com/en/topics/open-source/what-is-open-source-software