Integrating Ethical Design into Computer Science Assignments

TILMAN DINGLER, The University of Melbourne, Australia

The growing impact and the unintended consequences of computing have given rise to a series of problems and concerns: biased algorithms, self-driving car crashes, and the spread of misinformation online have led to a recent awakening and questioning of the "*Move fast and break things*" doctrine. While other engineering disciplines have had quality controls for decades, Computer science is still wrestling with standards and control mechanisms. Ethical design considerations are rarely more than an afterthought in curricula. In this article, we discuss the importance and opportunities of integrating ethical design into assignments and projects of existing courses. We present a case study using a final project assessment to get students into the habit of reflection on their design choices and capitalize on the engaging nature of ethical challenges.

CCS Concepts: • **Human-centered computing → HCI theory, concepts and models**.

Additional Key Words and Phrases: Education, Computer Science, Ethics

ACM Reference Format:
Tilman Dingler. 2018. Integrating Ethical Design into Computer Science Assignments. In *Yokohama '21: 2021 ACM CHI Virtual Conference on Human Factors in Computing Systems, May 08–13, 2018, Yokohama, Japan.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Computing and algorithms permeate all aspects of modern life. The modern technology industry has churned out services and products that impact billions of lives around the globe. With a seemingly insatiable appetite for new CS graduates, the industry's infamous motto of "*Move fast and break things*" has inspired whole generations of programmers to chase innovation without much regard for its impact and often unintended consequences on society. Few days go by without reports about programming errors, data leaks, security holes, and biased algorithms [8], the effects of which go as far as causing catastrophic accidents [9] and even undermining our democratic processes and institutions [3].

While engineering disciplines have had strict and systematic quality controls for decades, CS is still struggling to establish standards and procedures to ensure quality and safety [2]. Security, privacy, and ethical considerations are often little more than an afterthought in courses about programming and software engineering. Recommendation algorithms that send users down a rabbit hole of ever more radical content and autonomous cars that mistake a high-clearance truck for blue sky have made it clear that ethical considerations and prospective thinking should be left no longer to a postscript. Dedicated subjects, such as Digital Ethics, are slowly being introduced into the CS curriculum, yet there are countless opportunities to integrate ethics into assignments and projects of existing courses.

In this article, we make a case for seeking out opportunities in existing CS syllabi to 1) instil a reflective mindset in students early on and 2) capitalize on the engaging nature of cutting-edge challenges and their technical and ethical natures.

2 ETHICAL DESIGN AS A FORM OF ACTIVE LEARNING

Active learning involves students actively and experientially in the learning process [4]. It encourages engagement beyond passive lecture content processing. Problem-based assignments with a connection to current technological developments and the ethical questions that accompany them build a higher-order thinking task for students as it requires analysis, synthesis, and evaluation [10]. To include current topic assignments means to engage students and to connect their classroom learning to real-world problems.

Students bring different motivations and purposes for learning into the classroom, with varying backgrounds, expertise, and skillsets. Baking ethical dilemmas and attempts to their solutions into problem sets, therefore, has the potential to utilize the variety of perspectives to trigger diverse discussions. Hattie [7], in asking '*What works best?*' in higher education student learning, found that 50% of the variance in student learning is a direct result of what students bring to the class. By engaging students with current technology challenges and ethical dilemmas, we have an opportunity to crowdsource their solutions and spark new ways of understanding.

Thorndike's idea of **connectionism theory** [11] is the basis of an active learning theory that promotes learning by forming associations. The integration of his *Law of Readiness* and the *Law of Effect* involves the experiential and emotional reaction of the learner. In accordance, learning will always be more effective when a feeling of accomplishment and reward accompanies or is a result of the learning process. Bringing a set of real-world ethical design challenges into the CS curriculum may allow students to build engagement through a sense of responsibility and form associations between design decisions and their consequences down the line.

To demonstrate the integration of ethical design into an existing syllabus, we turned to one of the classics of ethical dilemmas.

3 CASE STUDY: MORAL MACHINES

For a final project assignment of an introductory Master-level course on *Programming and Software Engineering*, we devised a set of problems stemming from the *Trolley Dilemma*, a fictional scenario presenting a decision-maker with a moral dilemma: choosing "*the lesser of two evils*". The scenario entails an autonomous car whose brakes fail at a pedestrian crossing. As it is too late to relinquish control to the car's passengers, the car needs to make a decision about whose lives should be saved in a particular scenario based on the facts available. This particular variant of the problem was conceptualized by Awad *et al.'s* [1] famous crowdsourcing experiment *Moral Machines*[1].

Figure 1 shows an example scenario in which two groups of people with varying characteristics are present: car passengers and pedestrians at a crossing. In the left scenario, the car will continue ahead and drive through the crossing resulting in the death of the depicted pedestrians. If the car swerved, it would crash into a concrete barrier resulting in its passengers' death.

In this particular assignment, students are required to create a program designed to generate and explore different scenarios, build an algorithm to decide between the life of the car's passengers *vs.* the life of the pedestrians, audit their decision-making algorithm through simulations, and allow program users to manually judge the outcomes themselves. The assignment covers the basics of object-oriented programming, including encapsulation, inheritance, polymorphism, file i/o, and interactive elements.

The essence of the assignment, however, consists of three main aspects: 1) the design of a decision algorithm, 2) the implementation of an algorithm audit, and 3) a reflection of how the algorithmic decision choices affect the outcome.

[1]https://www.moralmachine.net/

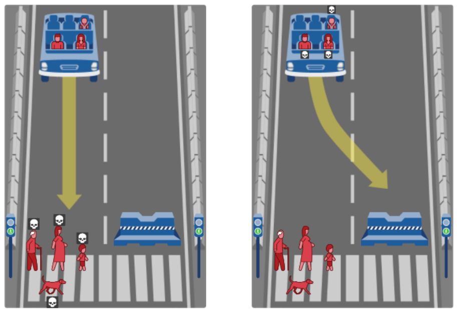

Fig. 1. Scenario example: a self-driving car approaches a pedestrian crossing but its breaks fail. Students need to design an algorithm that decides who survives a particular scenario (image source: http://moralmachine.mit.edu/).

3.1 Decision Algorithm

The heart of the program is the design of a decision algorithm that chooses for each given scenario whom to save and whom to condemn. To make this decision, the algorithm needs to consider the characteristics of the characters involved (e.g., age, gender, body type, profession, lawful employment, the presence of pets) as well as the situation (*i.e.*, whether it's a legal crossing). The problem description emphasizes that—for the sake of the assignment—there is no right or wrong in how the algorithm is designed. Execution is what matters, so students need to make sure their code meets the technical specifications. They are explicitly encouraged, however, to think about the consequences of their algorithmic design choices.

3.2 Algorithm Audit

An audit is an inspection of an algorithm with the goal of revealing inherent biases that may be built-in as a (un)intended consequence. By implementing an algorithm audit, students must create and simulate a variety of scenarios and run them through their decision algorithm. The audit is then initialized with a variable number of either randomly generated scenarios (e.g., N=1000) or is conducted on a given set of scenarios from an external data source (e.g., a local file) to be able to compare different algorithms' outcomes. The audit results in a statistic, which accounts for the different scenarios' characteristics and their overall rate of survival.

3.3 Reflection

Reflection is a process of explicitly noticing our experiences and translating them into useful insights that will help us understand a topic as well as ourselves better. This self-knowledge is the key to deeper understanding and learning [5].

The final task of the assignment is an optional part where students can score bonus points. It gives them the opportunity to reflect on and describe the reasons they applied when designing their program's decision-making, as well as the consequences revealed by auditing their algorithm. For example, what were inherent biases they might have become aware of by running the simulations? Were there any surprises? What were the consequences of design choices that they took as a programmer in general?

3.4 Observations

Two cohorts of students (N=534) have now gone through this exercise. The project was designed as an exam replacement during the COVID-19 pandemic and was administered online. Students had five weeks to complete the project, and automated testing was provided on university servers to make sure the students' programs successfully compiled before final submission.

Although challenging in its nature, the project managed to elicit the students' interest in complex design questions beyond purely performance-driven choices. The online forum, which accompanied the course, ended up containing a thread with more than 900 student comments as a result of vivid discussions about algorithm design and persons' varying characteristics. Bringing a cutting-edge topic (the design of autonomous cars) into the assignment allowed students to connect and apply the often abstract lecture content to a concrete and up-to-date problem set. The algorithm design challenged students to conceptually ponder their assignments beyond technicalities and, as we hope, managed to trigger thoughts about programmers' responsibilities of modern technology design.

4 INTEGRATING ETHICAL DESIGN INTO CS EDUCATION

Barely any systems operate in a space where they do not affect humans in one way or another. Hence, there is an opportunity to integrate ethical design into any computer science course. Ethics should be more than a mere afterthought in a syllabus and can instead be thought of as a point of engagement and application. Algorithms impact our lives on countless levels, from collating our news feed to determining whether we are fit for a specific job or eligible for a certain insurance plan. Each of these challenges offers opportunities for students to engage with their moral complexities and develop a mindset for ethical consideration. New generations of CS graduates should not only be masters of the hard programming skills but be able to interrogate, analyze, and make decisions based on reflection and prospective thinking of the (un)intended consequences of their creations. Said generation might be better equipped to find a compromise between the dizzying speed of innovation and the need to "*Move slow and fix things.*".

5 ACKNOWLEDGEMENTS

While having gone through a few CS curricula void of ethics education as a student myself, I would like to thank the inspiration I received as a teacher from studying initiatives, such as *Embedded EthiCS* by Barbara Grosz and her colleagues [6]. I would like to further thank Evan Peck for his educational blog post on *The Ethical Engine*, which formed the basis of the case study assignment.

REFERENCES

[1] Edmond Awad, Sohan Dsouza, Richard Kim, Jonathan Schulz, Joseph Henrich, Azim Shariff, Jean-François Bonnefon, and Iyad Rahwan. 2018. The moral machine experiment. *Nature* 563, 7729 (2018), 59–64.

[2] Adam Barr. 2018. *The problem with software: Why smart engineers write bad code.* MIT Press.

[3] Giorgio Bertolin and Giorgio Bertolin. 2017. *Digital hydra: Security implications of false information online.* NATO Strategic Communications Centre of Excellence.

[4] Charles C Bonwell and James A Eison. 1991. *Active Learning: Creating Excitement in the Classroom. 1991 ASHE-ERIC Higher Education Reports.* ERIC.

[5] David Boud, Rosemary Keogh, and David Walker. 2013. *Reflection: Turning experience into learning.* Routledge.

[6] Barbara J Grosz, David Gray Grant, Kate Vredenburgh, Jeff Behrends, Lily Hu, Alison Simmons, and Jim Waldo. 2019. Embedded EthiCS: integrating ethics across CS education. *Commun. ACM* 62, 8 (2019), 54–61.

[7] J Hattie. 2015. The applicability of Visible Learning to higher education. Scholarship of Teaching and Learning in Psychology, 1 (1), 79–91.

[8] Keith Kirkpatrick. 2016. Battling algorithmic bias: How do we ensure algorithms treat us fairly? *Commun. ACM* 59, 10 (2016), 16–17.

[9] Nancy G Leveson and Clark S Turner. 1993. An investigation of the Therac-25 accidents. *Computer* 26, 7 (1993), 18–41.

[10] Alexander Renkl, Robert K Atkinson, Uwe H Maier, and Richard Staley. 2002. From example study to problem solving: Smooth transitions help learning. *The Journal of Experimental Education* 70, 4 (2002), 293–315.

[11] Edward L Thorndike. 1898. Animal intelligence. *Nature* 58, 1504 (1898), 390–390.