I believe that teaching is an iterative process, and that the most effective way to become a better instructor and mentor is to regularly solicit feedback, act on it, and continuously make improvements. At Khoury, I have been lucky enough to be able to iteratively develop a software engineering curriculum (CS4530) in collaboration with a fantastic set of co-instructors (including Mitch Wand, Frank Tip, Adeel Bhutta, John Boyland and Ferdinand Vesely) and with feedback from dozens of TAs and hundreds of students. At GMU, I developed undergraduate and graduate courses in distributed systems, web application development, and program analysis. I integrate research advances into my teaching by designing new lectures on topics like continuous integration, open source ecosystems, and software testing. I also integrate my teaching activities into my research agenda, using the classroom as a testbed for emerging technologies [1].

I have co-organized workshops on software engineering education, and discuss software engineering curriculum design with colleagues at other institutions. I make all of my lecture and project materials publicly available under an open source license (CC-BY-SA), which have been adopted in part or whole elsewhere at Northeastern (Ryan Rad's CS5500 in Vancouver), at CMU (Michael Hilton and Chris Timperley's CS 17-313) and at NJIT (Martin Kellogg's CS490). My teaching at GMU was recognized with a university-wide Teacher of Distinction award in 2020.

Teaching Philosophy

Teaching senior-level programming courses that engage *all* of the diverse students in the class is a challenge. Some students enter my courses with no prior experience using the languages, tools, and frameworks that are used in the class. At the other extreme, others have been on *multiple* co-ops where they have used those same technologies, and are eager to apply their skills to something new. Through my lectures, projects, and activities, I strive to provide every student with an opportunity to engage with the material, learn something new, and demonstrate that growth. I also integrate continuous evaluation into my teaching activities, soliciting and acting on student feedback throughout the semester in order to improve my teaching.

On-boarding students to new technologies. I carefully design the programming projects in my courses such that they begin with extremely detailed specifications and instructions, closely supporting students who haven't yet had exposure to the languages and frameworks used in the project. However, by the end of the course, the students implement a project that is entirely of their own specification and design. This approach allows all of the students to express their creativity and demonstrate mastery of the technologies, while still supporting them with a sufficient runway to learn these skills.

This philosophy is embodied in the project sequence that I designed for CS4530 (Fundamentals of Software Engineering). The course culminates with a team project, where students propose, design, implement and deploy a new feature for *Covey.Town*. Covey.Town is an open source video collaboration platform that I created, inspired by the "Gather.Town" platform that launched at the start of the pandemic. Users log in to join a "town," which is a 2D arcade-style world that they can explore. When multiple users come near each other, they can see and hear each other through an integrated video call. The project makes use of a large complex web of open source packages and infrastructure, including Typescript, React, NodeJS, Twilio's Programmable Video platform-as-a-service, Heroku and Netlify. Since I began at Northeastern, enrollment in this course has ranged from 90 to over 400 students, with a team of up to four instructors and 20+ TAs.

Before they reach the team project, each student completes an *individual project*, implementing and testing a new feature for Covey.Town, following the instructors' specifications and design. For example, in Fall 2022, the individual project was to implement a new abstraction to support interactable objects in the world, along with a "viewing area," which allows for social, synchronized playback of YouTube video streams in the town.

The individual project provides students with a gentle on-ramp to these technologies and codebase. I organize the individual project into two deliverables, and design these such that all students start from the

same codebase for each deliverable. I provide students with a partial test suite to evaluate the correctness of their implementation, and ask them to implement the missing tests to strengthen that test suite. The individual project is graded almost entirely graded automatically, using a thorough suite of unit and integration tests to check functionality, mutation analysis to check for test adequacy, and linters to check for coding style. To avoid students spamming the grading server with submissions, I allow students to resubmit their code up to five times per-24-hours. I provide explicit instruction on debugging, and how students should attempt to test and validate their fixes before submitting to the grading server.

This assignment sequence helps to set expectations for the scope of team project, where students propose and implement their own feature. In groups of four, students work closely with a TA (one TA per 4-5 groups) to refine a project concept into a work plan and eventually a delivered feature. Students are encouraged to make their projects publicly available under an open source license and to include links to their deployed feature and codebase on their résumés. Select student projects from Fall 2022 are highlighted on our course website. As of May 2023, this project sequence has been used in four iterations of 4530: Spring 2021, Spring 2022, Fall 2022, and Spring 2023.

I have discussed this concept with colleagues who have experience designing reusable course projects (in particular, NCSU's iTrust project, which survived 10+ years of evolution and use [9]), and am excited to see this project continue to grow and evolve. This project concept is being reused in Spring 2023 (while I am not teaching the course), this time asking students to implement "poster areas," where a user can upload a poster and "present" it on an easel in the game. This project concept has also being reused at NJIT[1].

Soliciting and Acting on Student Feedback. I gather feedback from my students throughout the semester and use that data to help me improve my teaching. For instance: on the day that each assignment is due, we have an in-class poll where I ask students three simple questions: (1) how hard was this assignment? (2) how fair was this assignment? (3) how long did you spend on this assignment? We review the results of the poll in class, and have an open discussion about the parts of the assignment that worked well for students, and those which might have been unintentionally tricky. In addition to discussing the results of this poll with the class, I also adjust the scope of future assignments based on those results. At the end of the semester, I reserve at least 45 minutes of the last class for a post-mortem of the course. These discussions have yielded valuable insights, particularly around how the lectures met students' needs for understanding how to complete their term projects. I have a similar post-mortem meeting with my TAs and co-instructors.

I also recognize the difficulty of engaging *all* students during class time — which is critical to allow me to gauge how well the class is absorbing new material. Particularly when teaching larger sections (e.g. 60+), I regularly integrate instant online polls (via PollEverywhere) into my lectures. I use find this approach particularly useful for brainstorming and reflection activities (where students share ideas/insights with the class), as even students who are more shy can share their thoughts with the whole class (anonymously).

Curriculum Development

Working with my colleagues (Frank Tip, Mitch Wand, John Boyland, Adeel Bhutta and Ferdinand Vesely), we have designed and continued to refine the undergraduate CS4530. Developing this course has truly been a team project, although there are some lectures that I feel most directly responsible for. In addition to the Covey.Town projects (described above), I developed new lectures on topics including software process, React, distributed systems, continuous integration, security, technical debt, open source culture, and ethics.

I have also designed a new graduate course on software engineering, "Advanced Software Engineering," which mixes lecture material with research articles, requiring students to critically analyze the state-of-the-art. I created this class to provide a venue for students to engage with software engineering research, and to apply advanced software engineering practices to a project of their choice. Previously, at GMU, I substantially refreshed the curriculum of three courses: SWE 432 "Web Application Development," CS475

[1] https://web.njit.edu/~mjk76/teaching/cs490-sp23/projects/ip1.html

"Distributed and Concurrent Systems" and SWE622 "Distributed Software Engineering." I make all of my course materials publicly available under an open source license, and engage with colleagues at other institutions to improve them further.

Integrating Research and Teaching. As demand for computing education continues to grow — both in Boston and across the network — we need new approaches to support both teachers and students. I actively engage with colleagues in the international software engineering education community and have organized workshops on the topic in 2022 and 2023. I have learned first-hand that ensuring uniformity and excellence from a staff of 20 TAs is a significant challenge. One ongoing research project focuses on identifying team communication issues, and involves researchers at Bowdoin, CMU and NCSU. We administer a weekly collaboration survey to each team, building on my collaborators' recent research [10]. To extend this work, I built an instructor-facing dashboard that summarizes each team's responses and sentiment over time. The dashboard provides an artifact for TAs to refer to when meeting regularly with their teams, and for instructors to refer to when meeting regularly with their TAs.

I have a long-term objective of making it easier for instructors to teach testing practices and to evaluate student tests in programming classes throughout the entire curriculum. One challenge in grading student test suites is that it can be a tedious process, often relying on TAs to manually determine the adequacy of the test suite. My research objective is to design an approach that requires limited TA and instructor effort in order to provide students with useful feedback on the adequacy of their test suite. Mutation analysis, which automatically seeds potential faults in an implementation, is a possible alternative to manual test suite evaluation approaches. However: a common criticism of mutation analysis is that these synthetic faults may not be representative of real faults (making it an invalid proxy metric). Using a dataset of 2,711 student assignment submissions from three institutions, we empirically evaluated whether mutation score is a good proxy for manually-seeded fault detection rate and faulty student implementation detection rate [1]. Our results show a strong correlation between mutation score and manually-seeded fault detection rate and a moderately strong correlation between mutation score and faulty student implementation detection. Mutation analysis is now regularly used in grading student test cases for CS 4530, and an ongoing project (supported by a Khoury Teaching Innovation Grant) is studying the applicability of the approach in Racket, to potentially deploy it in Fundies.

Mentoring and Advising

I currently advise five PhD students (three at Northeastern and two who remain at my former institution, GMU) and two MS students. During my career thus far, I have also advised a total of 19 undergraduates and 8 masters students. I have also mentored a postdoctoral scholar at GMU, Dr. Luìs Pina, who went on to become an Assistant Professor at the University of Illinois-Chicago. Many (but not all) of these mentorships were successful, which I measure from both the student's newfound knowledge and appreciation for software engineering and from tangible artifacts (papers, reports, and code). I have built systems and published articles with my students [1]–[6] and have helped them to make contributions to popular open source projects (e.g. Apache Maven [7], [8]). Although it has taken some time to make these connections across the lab, I have also begun to collaborate with other faculty's students. For example: I have supported Arjun's students Donald and Federico in their recent software engineering foray into open source ecosystems (including [11] and some ongoing work), and am currently also working with Frank's student Satya on asynchronous flaky tests. With each collaboration, I gain new perspectives on advising and mentorship, and look forward to future projects and students.

Referenced Publications

[1] J. Perretta, A. DeOrio, A. Guha, and J. Bell, "On the use of mutation analysis for evaluating student test suite quality," in *Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis*, Acceptance rate: 27%., 2022.

[2] K. Hough, G. Welearegai, C. Hammer, and J. Bell, "Revealing injection vulnerabilities by leveraging existing tests," in *Proceedings of the 2020 International Conference on Software Engineering*, 2020.

[3] J. Kukucka, L. Pina, P. Ammann, and J. Bell, "Confetti: Amplifying concolic guidance for fuzzers," in *Proceedings of the 2022 International Conference on Software Engineering*, 2022.

[4] K. Hough and J. Bell, "A practical approach for dynamic taint tracking with control-flow relationships," *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 2, 2021.

[5] A. Alshammari, C. Morris, M. Hilton, and J. Bell, "Flakeflagger: Predicting flakiness without rerunning tests," in *Proceedings of the 2021 International Conference on Software Engineering*, 2021.

[6] A. Celik, P. Nie, M. Coley, A. Milicevic, J. Bell, and M. Gligoric, "Experience report: Debugging the performance of maven's test isolation," in *Proceedings of the 2020 International Symposium on Software Testing and Analysis*, 2020.

[11] D. Pinckney, F. Cassano, A. Guha, J. Bell, M. Culo, and T. Gamblin, "Flexible and optimal dependency management via max-smt," in *Proceedings of the 2023 International Conference on Software Engineering*, 2023.

Referenced Open Source Contributions

[7] M. Coley and J. Bell, *Surefire-1584: Add option to rerun failing tests for JUnit5*, https://github.com/apache/maven-surefire/pull/245, 2019.

[8] M. Coley, *[SUREFIRE-1711] Support ParameterizedTest for JUnit 5 test reruns*, https://github.com/apache/maven-surefire/pull/252, 2019.

External References

[9] S. Heckman, K. T. Stolee, and C. Parnin, "10+ years of teaching software engineering with itrust: The good, the bad, and the ugly," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*, 2018, pp. 1–4.

[10] K. Presler-Marshall, S. Heckman, and K. T. Stolee, "What makes team[s] work? a study of team characteristics in software engineering projects," in *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1*, 2022.