ThanhVu Nguyen: Teaching and Mentoring

1 Teaching

At UNL I have taught Automata (twice, ≈40 students), Software Engineering 2 (once, ≈30 students), Software Verification seminar (SV, 5 times, ≈8 students), Software Testing, Verification, and Analysis (TVA, twice, ≈70 students), and Compilers (twice, ≈20 students). My recurring courses, which I have designed from scratch, are SV, TVA, and Compilers. SV is a graduate-level course and the other courses are for both graduate and (mostly) undergraduate students.

Goal and Strategies My teaching goal is to have students finishing the class understanding the subject and being able to apply it to real life. I use several strategies to achieve this:

Motivations. I put extra efforts in creating motivating and real-world to helping students appreciate and remember difficult CS concepts. For example, when teaching about *fuzz testing* in TVA, I tell the story of Barton Miller conceiving the idea in 1988 by noticing that thunderstorm generates noise on the telephone line, which in turn creates bad inputs to remote Unix commands, causing them to crash. This semester I also use this recently reported bug in LinuxMint[1] to motivate fuzzing "*A few weeks ago, my kids wanted to hack my Linux desktop, so they typed and clicked everywhere, while I was standing behind them looking at them play... when the screensaver core dumped and they actually hacked their way in! wow, those little hackers...*"

Practical tools. Seeing how learned concepts are implemented and used in the real-world helps the students solidify their knowledge and understand the subject better. Thus, I introduce students to many practical program analysis tools such as Facebook Infer, which checks various problems in C and Java code in apps such as Instagram and Messenger, and MyPy, which provides static type checking to Python. In their final projects, TVA and SV students learn how to use existing and popular tools and apply them to large real-world programs. Several students expressed that they were not aware of these tools but were amazed by their capabilities, and they intend to continue learning and using these tools in their development environment.

Programming Assignments (PA's). I assign challenging but interesting PA's to strengthen and sustain students' understandings. For example, my TVA students enjoyed using their genetic algorithm (GA) implementation to guess long, secret passphrases[2]. Moreover, to encourage students to think outside-the-box, my PA's often ask students to apply covered techniques on problems in emerging real-world domains. For example, PA4 in TVA'20 requires students to implement and apply *symbolic testing* and *theorem proving* techniques to analyze Deep Neural Networks[3] (DNN's). These students were quite interested DNN's and thus ended up learning about formal specifications and properties of DNN's and mastering the popular Microsoft Z3 constraint solver so that they can use it efficiently to analyze complex DNN's.

Interaction Book I plan to turn my course lectures and PA's into an open-source *interactive book*, which has a similar Jupyter/Python-based format as the Fuzzing Book for software testing but focuses on popular static analyses. The dynamic format of the book can make learning more

[1] https://github.com/linuxmint/cinnamon-screensaver/issues/354
[2] https://nguyenthanhvuh.github.io/class-tva/pa2
[3] https://nguyenthanhvuh.github.io/class-tva/pa4

enjoyable as students can edit code and experiment with their effects live on the web browser. Students can also contribute by extending the Markdown text or Python code through Github.

Many of my collaborators teach similar SE/PL courses and thus are potential users of the book (e.g., Sebastian Elbaum at UVA has expressed desires to contribute and use particular modules of the book in his course). This idea has been proposed in my recent NSF submissions and received praises from reviewers. I also have an REU proposal to involve undergraduates in the project.

Feedback I continuously refine my class materials based on feedback from peers (whom I invite to my classes to observe and evaluate my teaching) and students. For example, when I first taught TVA in Fall'19, students mentioned that the PA's were not clear on specifications and expectations. Since then, I have significantly improved the PA's and include detailed specifications and examples, grading rubrics, etc[4] (for Compilers, I even include Youtube video's demonstrating how to do the PA's[5]). Also, when I first taught Compilers last year, students observed that test cases are useful to debug their works. Over the recent winter break, I have created hundreds of tests and even automated scripts for students in the current Compilers class. I also ask students to provide suggestions (i.e., words of wisdom) to other students planning to take the course[6].

2 Mentoring

Currently I am working with 2 Ph.D., 1 MS, and 3 undergraduate students. My senior Ph.D. student has published in total 4 full top-tier conference papers (PLDI, ICSE, etc) and multiple short conference and workshop papers on the invariant generation and program repairs. He plans to graduate in Fall 2021 and seeks a postdoc or faculty position. One of my undergraduates, who is a sophomore and just worked with me for about a year, has already published 2 refereed conference papers (a full ICSE '21 a short ICSME New Idea and Emerging Results paper) and submitted a top-tier TSE journal paper. He plans to apply to Microsoft research this summer (he already interned at Facebook last summer) and continues with a Ph.D. study after graduation.

In the past 4 years at UNL, I have involved *nine* undergraduate research assistants and volunteers (most are first-generation students, two are freshman, six have received departmental or university awards). I also strongly believe in broadening participation and have been recruiting students from underrepresented groups. So far I have worked with two female undergraduates and one of my Ph.D. students is a minority.

My goal is to provide a welcoming and diverse environment for students to join and stay in my group. I have been using the SCORE group meeting method (developed at UMD where I was a postdoc), which is an all-hands status meeting 3 times a week for 15 minutes around noon. This approach favors frequent and brief group status updates together with ad hoc, one-on-one meetings based on demands. This way everyone receives frequent interaction and is aware of each other's projects, progress, and issues. To further increase group spirit and productivity, my group also has "retreat" days where once a month we meet outside of campus (typically at my house) and spend an entire day focusing on just writing or coding[7].

[4] https://nguyenthanhvuh.github.io/class-tva/pa3

[5] https://nguyenthanhvuh.github.io/class-compilers/pa1/

[6] Feedback and suggestions for Compilers and TVA are at `https://nguyenthanhvuh.github.io/class-compilers/class_feedback.txt` and `https://nguyenthanhvuh.github.io/class-tva/class_feedback.txt`

[7] During the pandemic we still have our status meetings and retreat days through Zoom and Slack.