Aaditya Naik: Teaching Statement

The integration of AI code-assist agents and Large Language Models (LLMs) into the development lifecycle represents a paradigm shift in software engineering. However, it introduces a critical vulnerability: the reliance on probabilistic generation for deterministic engineering tasks. The primary challenge for the next generation of engineers is no longer merely writing syntax; it is effectively orchestrating AI agents to build robust, verifiable systems. The competence of the modern engineer now depends on wielding these powerful tools without compromising fundamental correctness. My teaching agenda addresses this gap by training students to develop software using sound engineering practices, enforcing rigor, especially when AI is generating the code.

1 Teaching Philosophy

My teaching philosophy is shaped by the need to train highly effective software engineers who can adapt to the ever-increasing suite of AI-assistants at their disposal. I aim to do so by balancing AI proficiency with Engineering Rigor. While AI can rapidly generate code, it cannot guarantee the correctness, reliability, or maintainability of the code. I thus want to train students to be expert "conductors" of code generators, treating them as junior developers that require detailed specifications, rigorous supervision, and thorough code review. By the end of my course, they should be able to take complex software goals, break them down into tasks achievable by LLMs, and apply sound verification and testing techniques on the produced code.

In my classroom, I intend to bridge the gap between traditional software engineering and modern AI-assisted coding. Whether it is in a core undergraduate course or a graduate elective, I want to emphasize core software engineering concepts that are key to producing high-quality software, touching on principles such as property-based testing, formal specifications, and modular design, to guide AI assistants. I will also focus on traditional software development strategies like scrum and agile development, and discuss what they look like in the age of AI-driven assistants.

I also intend to bring cutting-edge software engineering research into my courses, highlighting areas of innovation in developing and testing software and AI. My goal here is to develop in students an inquisitive mindset towards the practice of software engineering, encouraging them to deeply understand the tools of their trade and their shortcomings in order to use them to their full potential.

2 Teaching Experience

I have been heavily involved in teaching and mentoring since my days as an undergraduate, spanning the full spectrum of computer science.

During my PhD studies at the University of Pennsylvania, I served as a teaching assistant for the Software Analysis course (CIS 5470) five times, twice as a Head TA, for which I won an Outstanding Teaching Award. This course focuses on building static and dynamic analyses of programs using industrial-strength compiler libraries like LLVM and tools like AFL and Z3.

As a TA, I guided students through the notorious labyrinth that is the LLVM documentation, helping them realize abstract concepts they learned in the course as runnable tools. I also led a fundamental redesign of the course exercises, streamlining the autograder pipeline and aligning them more closely to the course material. By containerizing the environment and simplifying the LLVM build, I enabled students to focus more on implementing concepts rather than struggling with lab harnesses.

Prior to my graduate work, I also built a strong foundation in teaching fundamentals. As an undergraduate active in my student ACM chapter, I designed and taught an in-depth course on advanced C++, spanning across core concepts like pointer mechanisms and modern libraries like STLs. My experience taught me how to break down complex concepts into accessible and manageable components.

My teaching experience has thus prepared me to teach undergraduate courses like Software Engineering and Compilers, as well as graduate electives like Automated Verification and AI-Augmented Software Development.

3 Mentoring

I view mentorship as a vital extension of teaching. I have mentored a diverse group of researchers, including three undergraduates, two graduate students, and a professional engineer. Collaborating with these talented individuals has refined my ability to operationalize complex research problems into achievable milestones.

I began mentoring undergraduates during my first year of doctoral study. I worked with Jonathan Mendelson on a novel program synthesis approach for predicate invention, guiding him through experimental design and publication. This work appeared in AAAI 2021.

Subsequently, I mentored Jason Liu and Amish Sethi, undergraduate researchers who were instrumental in scaling our neurosymbolic learning frameworks on GPUs. This collaboration resulted in a publication at ICML 2025. Amish is now actively applying to top-tier PhD programs to launch his own research career.

Most recently, I mentored Claire Wang, a professional engineer seeking to pivot into academic research. We worked to transition her industry expertise into a rigorous research mindset, leading to her contributing significantly to our ICML 2025 work and joining our lab as a PhD student.

I find mentoring to be deeply fulfilling, as it offers the opportunity to collaborate with exceptional students. As a faculty member, I look forward to building a research group that empowers the next generation of researchers to tackle the grand challenges of reliable AI.