Richard Rutledge

Teaching Statement

I believe an effective teaching approach in Software Engineering includes the following four principal objectives.

1) Help students achieve a solid understanding of core, fundamental concepts.
2) Engage with students to develop problem-solving skills, enabling the application of fundamentals.
3) Maintain an inclusive learning environment and encourage all curious minds.
4) Mentor students to give them the insight to make critical academic and career choices.

In the following, I expand upon these objectives after summarizing my teaching experience.

Experience

Georgia Institute of Technology: Spring 2021 - Current
Head Teaching Assistant for CS-6300: Software Development Process

> A course offered three semesters per year through the Online Master of Science in Computer Science (OMSCS) program. The class size ranges from 600 students in the summer to 800 students in spring and fall.

Georgia Institute of Technology: Spring 2019
Teaching Assistant for CS-6301: Advanced Software Engineering

> A traditional, on-campus course that is typically offered once per year. It contains a substantial project component, with a typical class size of 50 graduate students.

Georgia Institute of Technology: Spring 2014
Instructor of Record for CS-4912: Computational Media Senior Design

> This is the senior project class for students in the Bachelor of Science in Computational Media program. I lectured on software development processes, from Requirements Engineering to Maintenance, while student teams each developed a project for an outside client.

University of Georgia: Spring 1981 - Fall 1983 Undergraduate Teaching Assistant: Math Lab

> An environment for TAs to provide extra instructional assistance to other undergraduate students in calculus and linear algebra.

Fundamentals

Even though software engineering is often characterized by frequently evolving tools and techniques, there are fundamental concepts that remain relevant. And a thorough understanding of these fundamentals can make the difference between an acceptable developer and an exceptional one. I saw the importance of a firm grounding as long ago as when I TAed in the UGA Math Lab. For example, when students were stuck solving integration word problems, they invariably wanted

2091 Stanrich Ct, Marietta, GA
678.467.6600 • *rrutledge@gatech.edu*
https://cc.gatech.edu/~rrutledg/

to know which of several equations to use. Rather than answer the question directly, we would jointly diagram the problem on a chalkboard, partition the diagram, recognize a Riemann sum, and finally derive an integral equation. This type of student interaction may take more time initially. But typically, those students that learned the fundamental application of an integral could work the remaining problems themselves. Those that did not, resorted to memorization of equations. Software engineering has its own fundamentals. Techniques such as 'Delta Debugging' and 'Category-Partition' will remain valuable approaches as long as programs exhibit undesirable behavior and consume inputs, irrespective of future programming language or frameworks.

Engagement

As an instructor, I engage with the students in the class. Students learn best when lecture material is reinforced through active learning, interweaving the lesson with student participation such as projects, class discussion, student presentations, etc. In the Computational Media Senior Design class that I taught at the Georgia Institute for Technology, I adopted a highly interactive teaching style, which I applied to all lectures. Each topic point in the lecture would consist of a 10 to 15-minute lesson to present the point. I then engaged the students in a short class discussion on some aspect of the topic.

Inclusivity

"I am, somehow, less interested in the weight and convolutions of Einstein's brain than in the near certainty that people of equal talent have lived and died in cotton fields and sweatshops." - S. Gould

Students will be reluctant to engage without a sense of belonging to the class and amongst their fellow students. Instructors can work to cultivate this sense by encouraging a diversity of perspectives, while regarding each perspective only upon its own merits and without regard to the identity (including race, gender, class, etc.) of its originator. Implicit bias, such as gender-bias in Science, Technology, Engineering, and Math fields, can have a corrosive effect on a student's perception of acceptance and belonging. In her forth grade, my daughter asked me why her school had placed her in advanced math. When I asked if that was not where she should be, she replied, 'No, I'm a girl.' Of course, I was horrified. But her story has a happy ending; she graduated with dual degree degrees in mathematics and genetics. In CS4912, I coached a few students still coping with the residual effects, some clearly anxious about participation and some too quick to judge an idea by its source. A single term is woefully short, but I sincerely hope I make a difference. Inclusivity is the right approach for the student, but it's also the right approach for society. Software development is difficult and challenging; our field needs more of the best and brightest.

Mentoring

An instructor should strive to be a role model for students. As an undergraduate mathematics major, I was fortunate to be mentored by John Hollingsworth. From him, I learned how to think about topics ranging from calculus to algebraic topology and, along the way, how to reason abstractly about a problem. As an aside, I still introduce myself with 'Greetings' in recognition of Hollingsworth; though perhaps without his booming style. Mentorship was an essential part of my practitioner role as a lead developer/architect. Both newly graduated and experienced hires need to continue their professional development and mentoring advances both parties in the relationship. Applied consistently, a strong mentoring network even benefits the organization by supporting an inclusive workplace/academic culture promoting learning, engagement, and collaboration.