Teaching Statement

Gregory James Gay
Department of Computer Science & Engineering, University of South Carolina
3A66 Swearingen Engineering Center, 315 Main Street, Columbia, SC 29208
(803)-777-9479, greg@greggay.com

Addressing the problems posed to modern computer science requires both of the traditional roles of the professor—researcher and educator. Researchers discover solutions and propose techniques to address the challenges that face our field, but it is not enough to solve these problems in isolation—it is only through teaching that a new generation of computer scientists will recognize the challenges that exist in computer science, learn how to apply the knowledge acquired in their university courses to such problems, and discover new and innovative solutions of their own. I strongly believe in the importance of the educator role of the professor, and through teaching and outreach, I hope to inspire new generations of engineers, scientists, and innovators.

My teaching philosophy has been shaped by my experiences on both sides of the student & instructor divide, as well as the realities of the rapidly evolving field of computing. The fast pace of advances in the state of the art require that students become adaptable, learning both the core principles of computer science and skills that are relevant to and competitive in today's job market. It is not sufficient that students memorize a static set of slides; students must also gain hands-on experience with real-world problems and discover how to apply the knowledge learned in lectures to solve them. As a Ph.D. student, I have had the opportunity to put my teaching philosophy into practice while serving as a teaching assistant for four semesters, on two different software engineering courses. For both courses, I stepped beyond the standard responsibilities of a teaching assistant and shaped course assignments, exams, and guest lecture opportunities around my own teaching philosophy:

- I shaped course projects around a realistic development cycle, with each subsequent assignment building on previous efforts. This approach gave students hands-on experience with different stages of software development (requirements elicitation, design, implementation, and testing) and allowed the students to apply the knowledge from lectures to an example of a real-world industrial system.
- Rather than basing assignment and test questions around recitation of material, I instead designed them as open-ended applications of course knowledge to realistic problems and case examples. When possible, students were allowed a large degree of freedom in the design and implementation of projects. These assessments promoted sound engineering principles while challenging students' creativity, individuality, and problem solving abilities.
- I developed lectures and assignments to draw on the students' own experiences and be relevant to current trends and techniques in both industry and research.

My educational vision is designed to provide the foundational knowledge required to develop quality software while at the same time being **relevant, attractive,** and **connected**:

- **Relevant** to the future career of my students. A solid education in the fundamentals of computer science and hands-on experience with problems in the field will prepare students for the challenges in the workplace and inspire them to become lifelong learners.
- **Attractive** so that the excitement and intellectual content of computer science will inspire students to move from memorization of content, to the application of that content, and finally, to the discovery of new approaches and techniques.
- **Connected** to local industry, government, and professional organizations through partnerships and integrated activities (such as site visits or guest lectures).

I intend to keep courses **relevant** by updating lecture material to cover new trends and techniques in industry and research, while maintaining a solid education in the fundamentals of the courses. Students must develop a solid understanding of the cornerstones of software engineering, but new techniques and discoveries can be disseminated to the students in terms of these fundamental concepts. I will always strive to prepare students to answer the questions *"How does it work?"*, *"Why does it work?"*, and *"Where does it work?"*. By answering these questions, students will be well equipped to learn, evaluate, and deploy tools and techniques that are relevant in today's job market as well as those that emerge in the future.

I aim to keep courses **attractive** through my passion for teaching and by covering the material in an engaging manner. In many areas, computer science is a practical subject—and, as a result, some of those courses are seen as less interesting—but with the proper instruction, all aspects of our field can be engaging and inspiring. Lectures should be grounded in realistic case examples and students should be given opportunities to apply the lessons through in-class active learning activities. Students should work in groups to develop an understanding of the human aspects of software development and to encourage collaborative learning. Ultimately, it is my responsibility to provide the right classroom environment to best teach the principles of computer science, and it is my aim to keep students interested through engaging delivery of lectures and hands-on exercises. My teaching style has been very well recieved by the students, and comments received on formal evaluations such as the ones below are common:

- "Thank you! One of the best TA's I've had here at the U."
- "Good job at lecturing!"
- "Very approachable and willing to help students understand."
- "Good communicator. Knowledgeable on subject matter. Makes good assignments."
- "Provides great help for students to understand problems and materials. Wants us to succeed in course."

Finally, I aim to keep courses **connected** to local industry, university, government, and professional organizations. I will invite industry representatives to give views "from the trenches" so that students develop an understanding of how the material from the course impacts real-world software development and develop an appreciation for how the lessons learned in class will be important in their future careers as software designers. I encourage students to become involved with both local industry—through internship, co-op, and other work study opportunities—and professional organizations so that they will learn of new developments and be better prepared for their chosen profession. My teaching and mentoring is not restricted to the classroom:

- As the senior student in my research lab, I have taken on an unofficial advisory role where I have assisted students with their research, educated them in areas that I am knowledgeable in, worked with them to develop their research and writing skills, and helped them meet their academic progress goals.
- Throughout my M.S. and Ph.D. programs, I have eagerly taken on opportunities to give lectures to fellow students on both areas relevant to my research and other topics in computer science ranging from basic Unix skills to the philosophy of open-source software development.
- As the president of the West Virginia University chapter of the Association for Computing Machinery, I sponsored lecture series to connect professors with the student body and inspire students to become interested in ongoing research efforts. I also led community outreach programs to teach computing skills and plant an interest in computer science in the minds of local youths.

To summarize, I believe strongly in the importance of teaching, and I view teaching as an integral part of my long-term career goals. I am passionate about inspiring new generations of engineers, and I aim to produce graduates who both understand the core concepts of software engineering and are able to creatively apply classroom knowledge to real-world problems.

Teaching Interests

I am interested in teaching a variety of courses at the graduate and undergraduate levels, including—among others—software engineering, software testing, data mining, artificial intelligence, programming languages, and data structures.