Teaching Statement

Brittany Johnson (bjohnson@cs.umass.edu)

My teaching philosophy is informed by my experiences as both a student and professor, along with research on active learning and other ways to engage students in the classroom and Computer Science curriculum. One concern, both as a previous student and future faculty member, is the strong focus on lecturing, textbook terminology, and "typical CSC problems," such as Towers of Hanoi, in undergraduate and graduate courses [1, 2]. While these are beneficial to learn, these are not catching or keeping students' engaged and retained. This is especially true for minority students [2, 3]. The principles that inform my teaching philosophy are *practical engagement*, *culturally-relevant lessons*, and *communication skills through interactivity.*

Engaging Students with Culturally-relevant, Practical and Interactive Teaching as Career Preparation

Practical Engagement in the Classroom

Rather than teaching CS concepts like a laundry list of definitions, which can be memorized and easily forgotten, I find it more useful when lessons are grounded in real world tools, practices, and research. Especially when for many students, the way CS is taught leads to misconceptions regarding the applicability of CS to their interests and future careers.

Consider a student whose interest in CS came from its applicability to her work in Biology. She learned how CS could improve her day-to-day activities, training that few biologists have. Had she majored in CS initially, there is no telling if she would have maintained the same interest. Once she's interested, we can teach with the goal of adding to, and possibly correcting, her programming mental models. As students learn and explore programming concepts they can realize the applicability, get excited about the material, and thereby go beyond memorization to building better mental models [4].

One way to engage students, especially minority students, is to incorporate research and real-world examples into coursework. This could lead to interest in undergraduate research experiences and a career in CS [3]. I can speak from personal experience, as my decision to stay in CS and continue to a graduate degree stemmed from a combination of my experiences doing research and in the classroom. My research focuses on software engineering tools and practices, which allows for incorporation of tooling and research relevant to the courses I would be teaching (e.g., introductory programming, human factors in software engineering) and approachable engagement opportunities inside and outside the classroom. For those with backgrounds outside of CS, it may be useful to incorporate interdisciplinary research that showcases the value of CS skills outside of a purely CS context [4].

I also believe in the early introduction and frequent use of program analysis tools and development environments in the classroom. There is a demand for more qualified software engineers, so we need curriculum that can support the development of capable and productive engineers. We can use tools to improve developer productivity but often developers don't use these tools due partially to inexperience with these tools [5, 6]. One potential solution to this problem is to teach tool use as part of the software development process, along with other traditional topics like writing and testing code. Therefore, my courses will often include use of relevant tooling and environments.

Culturally-Relevant Lessons & Activities

Through my teaching experiences, I have found that culturally-relevant lessons and activities can increase student engagement, thereby increasing their interest in course content. Culturally-relevant teaching involves using students' culture, experiences, and background to teach concepts of interest. This pedagogy has been found to be particularly effective for minority students [7]. Numerous panels comprised of minority students that I've attended and been a part of mention cultural relevance as one of the most important aspects of curriculum that keeps them engaged in the classroom.

My teaching experiences that incorporate culturally-relevant ideas have proven successful in more than one context and in more than one way. I have organized numerous Python workshops where I taught programming to non-CS students in the context of scientific computing. This was a topic that everyone was interested in prior to participating in the workshops with respect to each student's own research or coursework. Some students returned to future workshops, noting in their evaluations and in person that my teaching style, and incorporating concepts relevant to their non-CS majors and interests, helped them grasp the fundamentals and acquire a desire to learn more.

I also co-led programming camps for minority students at the Wade Edwards Learning Lab in Raleigh, NC using game and mobile apps as the method of delivery. Though many students did not know what Computer Science was entering the camp, they signed up from interest and experience in playing games and using apps themselves. To increase cultural-relevance in students' projects, I use examples from the real world (e.g., pop culture references and examples). I encouraged students to do the same when creating their own games and mobile applications.

From the four different camps I led, two students (one male and one female) have since reached out to me to inform me they have continued on to CS degree programs. Both students came into the camp unsure of their desire to pursue CS, based mostly uncertainty regarding what CS is and how it could apply to their own interests and career goals. It was rewarding to see their attitudes towards CS change and beyond gratifying to have both credit my teaching for this transition. The female student, hearing about my experiences with research, was inspired to explore research opportunities for herself. The male student returned to volunteer at the camps that inspired him to pursue a degree in CS.

Most recently, I designed and led a course on the great contributions made by women in CS. One of the main goals while designing this course was to incorporate women from varied backgrounds. By doing this, no matter the culture, experiences, or backgrounds of the students, each student (male and female) found at least one story they could relate to, and perhaps be inspired by. In an attempt to strengthen the cultural relevance of each lesson, I allowed students to pick who they wanted to research and present on. Often, students selected someone that, whether they realized it up front or not, they could relate to on some level – many students explicitly noted this and how inspiring it was in the presentation reflections they submitted.

Communication Skills through Interactivity

Having a liberal arts background, I believe it is important to have a balance of technical and soft skills, such as communication, to stand out in today's job market. Many developers and hiring managers I have spoken with from industry feel highly technical schools prepare students for highly technical positions but not as much for collaborating and communicating their work and ideas to others. Communication skills are important whether you are a researcher or industry developer.

Therefore, I will incorporate communication skills via course interactivity. Exercising both communication and technical skills regularly is important for building each skill individually and show how they can and should work together. In the camps, workshops, and courses I've taught, I facilitate communication skill building through interactivity by frequently polling the class for information such as the progress they've made that day. When possible, I also integrate presentation skills into my lessons (for all age groups). In any multi-day course or workshop I've taught, there has always been at least one portion of a lecture devoted to how to present their work and a required end of course presentation on their accomplishments. I found this helps further strengthen the connection between the ability to do research or create technology and then communicate about it to others.

Contributing to the Curriculum

Below are a list of courses, based on my research and background, that I could teach, with a brief description of each course:

- **Software Engineering** An introductory or advanced course (undergraduate and graduate) that teaches foundations and best practices in software engineering.
- **Human Computer Interaction** An introductory or advanced course (undergraduate and graduate) that teaches foundations and best practices in human computer interaction.
- **Software Engineering in a Data Centric Society** A special topics course (undergraduate and graduate) that explores how big data and machine learning does, can, and should impact software engineering research and practice.
- **Hidden Figures: Contributions to CS by Underrepresented Groups** A first year seminar for undergraduate students that explores contributions made to the field of computer science by individuals from diverse backgrounds.
- **Empirical Research Methods** A special topics course (undergraduate and graduate) that introduces empirical research to students and provides an opportunity to get experience using a method of their choice to solve a problem.

References

[1] Felder, R. M., & Brent, R. (1996). *Navigating the bumpy road to student-centered instruction.* College teaching, 44(2), 43-47.

[2] Shaw, M. (2000, May). *Software engineering education: a roadmap.* In Proceedings of the conference on The future of Software Engineering (pp. 371-380). ACM.

[3] Aspray, W., & Bernat, A. (2000, March). *Recruitment and retention of underrepresented minority graduate students in computer science.* In Report on a Workshop by the Coalition to Diversity Computing.

[4] Forte, A., & Guzdial, M. (2005). *Motivation and nonmajors in computer science: identifying discrete audiences for introductory courses.* IEEE Transactions on Education, 48(2), 248-253.

[5] Bruckhaus, T., Madhavii, N. H., Janssen, I., & Henshaw, J. (1996). *The impact of tools on software productivity.* IEEE Software, 13(5), 29-38.

[6] Johnson, B., Pandita, R., Smith, J., Ford, D., Elder, S., Murphy-Hill, E., & Sadowski, C. *A Cross-Tool Communication Study on Program Analysis Tool Notifications.* In 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE).

[7] Ladson-Billings, G. (1995). *But that's just good teaching! The case for culturally relevant pedagogy.* Theory into practice, 34(3), 159-165.