TEACHING STATEMENT

GEOFFREY CHALLEN

04/04/2017

I believe that teaching is as important as research. As more students are exposed to computer science and learn our powerful tools, they will use those capabilities to change the world. Teaching also provides opportunities for computer scientists to build and evaluate effective education tools using our domain knowledge. Please peruse my teaching portfolio for more information about the courses I teach, including syllabi, online resources, and student evaluations.

Note that this statement focuses on my contributions to curriculum and course development and student outreach, rather than my approach to teaching specific courses. A more detailed overview of the courses that I have taught at UB can be found in my course improvement statement.

1 — COURSES TAUGHT

At UB my primary teaching responsibilities are a large flipped introductory course on the internet (CSE 199), an introduction to operating systems (CSE 421/521), and a graduate seminar on mobile systems (CSE 72x). For a more detailed description of my approach to teaching these classes and pedagogical innovations I have deployed, please review my course improvement statement.

2 — CURRICULUM IMPROVEMENT

From Fall 2015–Fall 2017 I chaired a subcommittee of the department's Undergraduate Advisory Committee. Our charge was to revise the entire CS curriculum. The last review was many years ago, so an update was overdue. Several other faculty members and a number of student volunteers participated in the process. We also solicited feedback from alumni and other department faculty as needed.

We focused on our department's Bachelor of Science degree in Computer Science, since this is by far the most popular undergraduate degree. There were two key problems that we identified with the current curriculum. First, over the years the number of required courses had slowly grown, causing the number of electives to dwindle. Students had too few electives to appreciate the breadth of computer science, or specialize in areas like artificial intelligence or databases. There were also unable to reach more advanced courses in many areas due to the excessive number of requirements. Second, because students were not taking very many electives the department was not offering many. We lacked undergraduate courses in a variety of important computer science subareas.

We tackled each of these problems in turn. We began by focusing on increasing the flexibility in our current curriculum by eliminating unnecessary or inappropriate requirements. A key part of this effort was coding our curriculum and the curriculum of many other schools for the purposes of comparison. The output of the first phase was a set of changes to our curriculum that can be implemented immediately and will have a large impact quickly. These changes are summarized here, and included:

- **A new three-course programming sequence**, including two new introductory programming courses and a new third-semester system programming class. (See Section 3 for more details.)
- **A new three-course theory sequence**, condensed and improved from four original courses after a large amount of work from the theory faculty.
- **Removal of several unnecessary or inappropriate requirements**, including courses on computer operating systems, programming languages, and circuit design.
- **New distribution requirements** applied to an increased number of electives.

The end result was a significantly modernized curriculum with twice as many electives despite having the same number of total requirements. We were also able to meet two significant external constraints: ABET accreditation, and SUNY seamless transfer requirements. Department faculty approved these changes in Fall 2016, and they will begin taking effect in 2017–2018.

The original plan was to follow these initial changes with more sweeping recommendations about how to modernize our entire program. We anticipated that these changes would be implemented over a longer time horizon and involve new courses and hiring. However, at this point I am no longer involved in the curriculum reform process.

3 — INTRO COURSE DEVELOPMENT

As part of the curriculum overhaul described in Section 2, I also became involved in the process of designing new introductory programming courses. I had seen during my time at UB how large an impact these courses have on education in the department. They introduce students to computer science and provide a chance for us to attract and retain a large and diverse group of students. They should provide a solid programming foundation and overview of CS that will support students in later courses.

My experience teaching OS caused me to realize that our current introductory courses were not meeting these goals. Students emerged with weak programming skills, unprepared for downstream courses or internships. This weakness persisted even until senior year, where it made it difficult for them to succeed at my OS assignments.

As a member of a committee devoted to redesigning these courses, I worked with Jesse Hartloff to propose a new introductory series. We eventually produced a long design document as well as shorter proposals for both first-semester (P1) and second-semester (P2) programming courses. These proposals outline content coverage on a week-by-week basis, course assignments, language choices and rationale, content delivery, and many other design choices required to implement a new introductory course. Implementing them should be a great deal of work, but straightforward given the design.

Our design goals for P1 and P2 included:

- **Rigor:** programming is a skill, and students need enough practice to get good at it. Our aim is for students to spend 10–15 hours a week outside of class on programming assignments. Particularly in P1, assignments are released on a weekly basis, which allows us to cover many topics while ensuring that students do not get too far behind or stuck working on an assignment they don't enjoy.
- **Excitement:** lectures and assignments are intended to excite students about computer science while introducing them to the conceptual breadth of the field. P1 covers many core topics—including algorithms, data structures, storage, and networking—at a high level to help hook students. P2 assumes a more committed audience and begins to tackle deeper how and why questions—such as compilation, concurrency, and object orientation.
- **Accessibility:** to ensure continued growth and relevance, computer science needs to attract people from outside our traditional core constituency. New groups are being drawn to CS because if it's ubiquity in our daily lives, and due to its potential to solve problems. P1 and P2 make explicit attempts to connect with the students motivations for studying computer science. Our goal is to ensure that students realize that CS provides some of the most powerful tool for solving any problem—and that they are prepared to harness that power by becoming strong programmers.

We have also designed both P1 and P2 to continue the large flipped classroom model that was pioneered at UB by my new course on the internet. All broadcast content will be delivered through short videos prepared and curated by the course staff. Our video delivery platform is designed to make it easy for multiple faculty and course staff to provide, upgrade, and reinforce video content. Class time will be spend on hands-on exercises that are related to the programming assignment that is due that week.

All course assignments will be automatically graded to reduce load on the course staff and allow iterative improvement through multiple submissions. The design document also incorporates an increased focus on code reading exercises, with quizzes generated and delivered by a new online tool.

We made our proposal public and received a great deal of useful feedback from current students and alumni. With help from Carl Alphonce and Luke Ziarek, our course proposal was approved as part of the broader curriculum overhaul. Preparations are underway to teach the course for the first time in 2017–2018, and roll it out for all CS majors by 2018–2019.

4 — OUTREACH

Complementary to my teaching are ongoing efforts at improving diversity within my field and exposing undergraduates to research. My diversity efforts are described in more detail in my service statement.

4.1 — Undergraduate Research

I love introducing undergraduates to research. I have had the privilege of working with 18 undergraduates on a variety of projects, several of which have produced publications. Nick DiRienzo built a system called PocketMocker allowing smartphone users to conceal their true activities by injecting fake data. Nick presented a poster at HotMobile'14 which won the best poster award and published a PocketMocker paper at MobiCASE'14. Frank Rossi assisted in the development of the PocketLocker system which creates personal storage clouds from multiple personal devices. John Cherry worked on file system analysis as part of the SUNY Louis Stokes Alliance for Minority Participation (LSAMP) summer program.

My group currently includes 9 undergraduates working on projects including improving user quality of experience (Brijesh Rakholia), improving smartphone energy management (Kyle Schoener), investigating the interaction between Android and "modder" communities (Edwin Santos), benchmarking Android databases (Grant Wrazen and Lakshmi Ethiraj), using crowdsourcing to optimize Wifi networks (Grant Wrazen and Vighnesh Iyer), and `internet-class.org` content and infrastructure (Greg Bunyea, Aishani Bhalla, and Wesley Csendom). I integrate undergraduates into my group by having them work alongside us in the lab, giving them tasks appropriate to their training, and providing graduate student mentors.