Teaching Statement

Swapneel Sheth

During my Ph.D. at Columbia University, my interest in teaching led me to apply for the Preceptor (Graduate Student Instructor) position towards the end of my third year in the Ph.D. program in late 2010. From Spring 2011 to Fall 2013, I had been one of several Preceptors in the department and taught four undergraduate classes and one graduate class. At the University of Pennsylvania, I've been a Lecturer from 2014-2017, a Senior Lecturer since then and I've taught nine undergraduate and eleven graduate classes so far. Starting from Spring 2018, I'm also the CIS Masters Chair and CIS/MSE Program Director.

I have been very honored to receive teaching awards at Columbia and Penn: the *Andrew P. Kosoresow Memorial Award for Excellence in Teaching and Service* in 2011 at Columbia, which is given to a Ph.D. student for "outstanding contributions to teaching in the Department and exemplary service to the Department and its mission."; the *Hatfield Award for Excellence in Teaching in the Lecturer and Practice Professor Track* in 2019 at Penn, which recognizes "outstanding teaching ability, dedication to innovative undergraduate instruction, and exemplary service to the School in consistently inspiring students in the engineering and scientific profession."

Teaching Experience at Penn — graduate and undergraduate classes

I've taught a variety of graduate and undergraduate classes here at Penn. They are described below.

I've taught NETS 150 — Market and Social Systems on the Internet six times. The is a core class for the Network and Social Systems program and covers a variety of topics including graphs and social networks, the Internet and the World Wide Web, search engines, document search, recommender systems, game theory, matching markets, and auctions. When I first taught it in Spring 2014, the enrollment was 61 students; this semester, it's currently closed at 189 students.

I've taught CIS 120 — Programming Languages and Techniques two times. CIS 120 introduces students to computer science by emphasizing the design aspects of programming and includes topics such as data types and data representation, abstraction, interfaces, and modularity, test-driven development, programming patterns (recursion, iteration, events, call-backs, collections, map-reduce, GUIs, etc.), and functional and object-oriented programming. When I last taught CIS 120 in Fall 2018, it was the largest class we've offered in CIS that semester with an enrollment of 374 split across two sections.

I've also taught CIT 591 — Intro to Software Development three times and CIT 594 — Data Structures and Software Design two times. These are both core classes for the MCIT program. 591 is an introduction to fundamental concepts of programming and computer science. The class starts off with an introduction to modern programming languages and aspects such as basic data types, loops, and conditionals. The class covers features of Object-Oriented programming languages including objects and classes, inheritance, and interfaces. In addition to programming, this class also focuses on best practices and aspects of software development such as software design and software testing. 594 focuses on data structures, software design, and advanced Java. The class starts off with an introduction to data structures and basics of the analysis of algorithms. Important data structures covered include arrays, lists, stacks, queues, trees, hash maps, and graphs. The class also focuses on software design and advanced Java topics such as software architectures, design patterns, networking, multithreading, and graphics. The enrollment for these classes has been 30-50 students.

I've taught CIS 557 — Programming for the Web three times (formerly CIT 597). The first half of 557 focuses on the basics of the Internet and the Web, HTML and CSS, and basic and advanced Ruby. The second half focuses on Ruby on Rails. Teams (of size 2-3) build a web application in the second half of the

semester as the class project. Through Rails, we explore the "culture" of web programming such as agile methodology, testing, key aspects of software engineering, using web services and APIs, and deploying to the cloud. Throughout the semester, teams learn and use the same world-class tools that are used to deploy and build real world web apps such as Ruby on Rails, GitHub, Pivotal Tracker, Cucumber and Capybara, RSpec, PostgreSQL, Heroku, and New Relic.

I've taught CIS 559 — Programming and Problem Solving three times. This is a unique class that focuses on Problem-based Learning and uses the Active Learning pedagogy. This is described in more detail in the following section. Due to the nature of this class, it is capped at 25 students.

Active Learning in CIS 559

The goal of Computer Science (CS, henceforth) is to use algorithms to solve problems. Problem solving in CS is a collaborative activity that involves analyzing and communicating solutions, not just implementing them. CIS 559 — Programming and Problem Solving focuses on problem solving in CS and is based on Knuth's popular seminar in the 1970's and 80's at Stanford University. In the class, students work together to use programming techniques to solve open-ended problems from the domains of optimization, simulation, etc. and develop problem solving skills using techniques that they have learned during their CS training. There are no "correct" answers to these problems; rather, the focus is on the four steps of the problem solving process: algorithmic thinking, implementation, analysis, and communication.

For this class, we use the Active Learning and Problem-based Learning pedagogies. This is a very unusual class (compared to typical CS classes) as there are no lectures, assignments, or exams. Students work on four open ended projects in teams of 3-4 for roughly three weeks each. Everyone works with their team outside of class and the class meeting times are used for discussing their approach and solutions, the progress they've made, getting feedback, comparing one's approach to that of other teams, etc. Hence, the focus of the class is not just on coming up with good solutions, but analyzing them, communicating them to the rest of the class, and collaborating with everyone on improving it.

We published a paper at SIGCSE in 2016 with authors from Penn, Columbia, and NYU where similar versions of the class have been taught for over 15 years. The paper is available here [9] along with a website[1] that has additional details. The website for the version taught at Penn is here[2].

Using Real World Data, Open-Ended Projects, and Projects with Customers

A common theme in most of my classes is the use of real world data for homework assignments. There have been a lot of Open Data Initiatives such as OpenDataPhilly[3], U.S. Government's Open Data[4], and NYC Open Data[5]. These datasets make homework assignments much more interesting and make the students appreciate the practical applications of CS. I've used these kinds of homework assignments in almost all my classes. Here are a few examples: For NETS 150, I provided them with Facebook data that has 4000 nodes (indicating people) and 80000 edges (indicating friendships) and students have to write code and implement algorithms learnt in class (such as BFS and DFS) to find out how far certain people are from each other and whether the graph is connected or not; For CIT 591, I provided them with Philly Bike Share data that had data on 230k trips in the third quarter of 2016 and students had to write code to analyse the data and find out information such as the most common starting station; For CIS 557, students had to scrape Wikipedia and use APIs from the NYTimes and ProPublica to answer questions on topics such as the Rio Olympics and US Election Finances.

In addition to this, almost all of my classes have an open-ended project at the end of the semester. Depending on the class, this varies from a 2-3 week project to an 8-week project. In all of these projects,

[1] `http://programming-and-problem-solving.github.io/`
[2] `http://www.cis.upenn.edu/~cis559/`
[3] `https://www.opendataphilly.org/`
[4] `https://www.data.gov/`
[5] `https://nycopendata.socrata.com/`

I encourage students to create projects that are more real-world and practical by leveraging these public datasets.

For the longer projects, I also give students the option of working with real-world customers, which are typically people from the local Penn/Philly community. For example, in Fall 2016 I taught CIS 557 — Programming for the Web, which involves an 8-week project. The total enrollment was 76 students and there were 26 teams. Out of these, 12 teams worked with customers. The customers included Nalaka Gooneratne (Associate Professor of Medicine at HUP), Peter Cobb (Lecturer in Archaeology/Classics/Penn Museum), and Arjun Parasher (an MD and fellow at Perelman). There are many advantages of these customer-based projects: students work on a real-world problem; what they build will end up being used by the customers at the end of the semester (rather than being thrown away as in most classes); they get to interact with non-CS/non-technical people, which is a very valuable skill to develop; since the customers don't typically have a CS background, what they want in the project can be unrealistic, change over time, etc. which mimics the real-world in industry very well.

We've published a paper at SIGCSE in 2017 on projects with real-world customers. The paper is titled "A Two-Course Sequence of Real Projects for Real Customers" [5] and was awarded as a SIGCSE Exemplary Paper that year.

One of the teams also participated in a novel project: the team at Penn collaborated with a team at the University of Hamburg in Germany working on a project for Daimler AG. This would give students experience with global teams and learning how to work remotely, dealing with time zones, cultures, etc., which is increasingly common at most large software companies. We plan on publishing papers about our experience with global teams.

CS Education Research

I am very interested in CS Education Research and am interested in exploring new techniques and pedagogy for teaching. I have been very active in attending conferences and publishing papers both here at Penn and also while I was at Columbia. I have also been a paper reviewer for SIGCSE for the last three years and I was an Associate Program Chair this year. I will be on the organizing committee for SIGCSE 2020. I have described two papers that we recently published in the preceding sections. In addition to this, I also published some papers while at Columbia.

In Spring 2012 at Columbia, I taught COMS 1007, which is intended as a second class for majors in CS. The total enrollment was 129 students. Students develop/improve their programming, problem solving, and design skills via a rigorous treatment of object-oriented concepts using Java as an example language.

I introduced novel competitive and collaborative aspects to the curriculum. For example, introductory CS classes do not focus on software testing and a lot of students' mental model is "if it compiles and runs without crashing, it must work fine." There have been many attempts at introducing software testing early in various CS programs [2, 4] but students remain averse to it [3]. To address this, we built a system called HALO — "Highly Addictive socially Optimized Software Engineering" [1, 6] that uses game-like elements and motifs from popular computer games to make software testing more engaging and social. We used HALO in the class assignments and this resulted in students getting an early insight into software testing.

Most introductory CS classes at Columbia (and other universities) have individual assignments and allow little or no collaboration. On the other hand, most graduate classes and real world projects are typically done in large teams. To get early exposure to working in teams, for the last assignment, students could optionally work in a team of up to three students and define their own project subject to a few constraints. This resulted in a lot of a interesting, socially relevant projects that the students were passionate about.

Another novel aspect in 1007 was an in-class tournament to motivate learning good design principles — students' participation in the tournament would be contingent on making the correct use of Java Interfaces in their code. Finally, I used an informal, fun, and collaborative classroom environment to ensure that stereotypes about CS being "boring" are not reinforced. We published a paper [7] and a book chapter [8] that describes in detail the approach that was taken with the class.

Departmental Responsibilities

As teaching faculty, in addition to teaching classes, I feel it is important to be a good departmental citizen. Towards this end, I have advised 6 senior theses and 14 senior design projects so far totalling 51 students. I have served on the Lecturer Search Committee several times in the last few years. I'm currently serving on our department's undergrad curriculum committee and on the Engineering Masters Awards Committee. I also helped run the Undergraduate Summer Research Program in CIS in the summer of 2016 [6]. Along with Joe Devietti, I've organized department-wide Demo Days for 6 semesters for a variety of undergrad and grad classes that have a significant project component.

Diversity, Inclusion, and Outreach

Diversity and inclusion in CS is something that's very important to me. Given recent events in the tech (and other) industries, I think we need to give even more thought to ensure that our classes, programs, and departments are diverse and inclusive to people from varying backgrounds.

This could involve a range of small to big changes like carefully thinking about and deciding which examples and homework assignments to use in class, creating an inclusive classroom environment where everyone feels comfortable to participate in, and looking overall at systemic things to improve upon for the department. One example of the latter that I was involved in was the CIS Diversity Summit. Along with several faculty, students, and staff, we organized the first diversity summit in Spring 2018 with the goal for students to tell us things that are working well and things that could be improved upon within the department as a whole. Student leaders representing First Generation/Low Income students (FGLI), The National Society of Black Engineers (NSBE), Out in Science Technology, Engineering and Mathematics (oSTEM), Society of Hispanic Professional Engineers (SHPE), and Women in Computer Science (WiCS) participated in the planning and execution of the Summit. It was very eye-opening for us (and others!) to hear about the experiences of our students and we got a lot of excellent feedback on things to improve in the short-, medium-, and long-term.

Another event I'm heavily involved in is Penn's representation at the Grace Hopper Celebration of Women in Computing (GHC, henceforth). For several years now, Penn has been a Gold or a Platinum sponsor for GHC and sent around 20-30 women to attend. I've been helping organize this for the last two years. Some of the recent changes I've helped with include a more fair and inclusive process for choosing who gets to go to GHC (via an application process that's evaluated using a double-blind review), more structure for first-time attendees (with a pre-departure orientation where people who've attended before share their experiences and dos and donts), and trying to get more sponsorship so we can send even more people in the future.

Beyond Penn, I think it's also important to contribute to and involve other local schools in these efforts. Towards this, I've been on the organizing committee for the Capital Region Celebration of Women in Computing conference in 2017 at Georgetown University[7] and in 2019 at James Madison University[8]. I was also on the organizing committee for the first Philadelphia Region Celebration of Women in Computing (PHICWIC) conference in 2018 here at Penn[9]. What I particularly enjoy about these regional conferences (as a contrast to GHC) is that it's a great opportunity for students to get to know, learn from, mentor, and be inspired by their peers who attend nearby institutions. For example, at PHICWIC, we had over 250 attendees from institutions including Penn, Drexel, Swarthmore, Camden Community College, James Madison University, Rowan University, and University of Mary Washington, and also several local high schools.

Finally, I would like to add that I have found being a teaching faculty member at Penn to be a very rewarding experience. Interacting with students is definitely my favorite aspect. I've realized that I've learnt a lot during this whole process — about students, about teaching (what to do, what not to do), and about myself. About students, I think it's important to keep in mind, and draw upon my own experiences as a

[6] http://www.cis.upenn.edu/about-research/ug-summer/index.php

[7] https://cra.org/cra-w/events/capwic-2017/

[8] https://capwic.org/

[9] https://cra.org/cra-w/events/2018-acm-philadelphia-region-celebration-of-women-in-computing-conference/

student, that students are often overwhelmed and more often than not, they are very interested in learning the material. About teaching, I think the most important thing is adaptability — adapting to new classs, new students, new techniques and methods of teaching and learning, and varying expectations from students every semester is key. I feel that with teaching, no matter how new or experienced you are, it is important to constantly keep improving and learning how to make oneself better.

References

[1] Jonathan Bell, Swapneel Sheth, and Gail Kaiser. Secret Ninja Testing with HALO Software Engineering. In *Proceedings of the 4th International Workshop on Social Software Engineering*, SSE '11, pages 43–47, New York, NY, USA, 2011. ACM.

[2] Stephen H. Edwards. Rethinking computer science education from a test-first perspective. In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, OOPSLA '03, pages 148–155, New York, NY, USA, 2003. ACM.

[3] Sebastian Elbaum, Suzette Person, Jon Dokulil, and Matt Jorde. Bug Hunt: Making Early Software Testing Lessons Engaging and Affordable. In *Proceedings of the 29th international conference on Software Engineering*, ICSE '07, pages 688–697, Washington, DC, USA, 2007. IEEE Computer Society.

[4] Ursula Jackson, Bill Z. Manaris, and Renée A. McCauley. Strategies for effective integration of software engineering concepts and techniques into the undergraduate computer science curriculum. In *Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education*, SIGCSE '97, pages 360–364, New York, NY, USA, 1997. ACM.

[5] Christian Murphy, Swapneel Sheth, and Sydney Morton. A two-course sequence of real projects for real customers. In *Proceedings of the 48th ACM Technical Symposium on Computing Science Education*, SIGCSE '17, New York, NY, USA, 2017. ACM.

[6] Swapneel Sheth, Jonathan Bell, and Gail Kaiser. HALO (Highly Addictive, sociaLly Optimized) Software Engineering. In *Proceeding of the 1st International Workshop on Games and software engineering*, GAS '11, pages 29–32, New York, NY, USA, 2011. ACM.

[7] Swapneel Sheth, Jonathan Bell, and Gail Kaiser. A competitive-collaborative approach for introducing software engineering in a cs2 class. In *2013 26th International Conference on Software Engineering Education and Training (CSEE&T)*, pages 41–50. IEEE, 2013.

[8] Swapneel Sheth, Jonathan Bell, and Gail Kaiser. A gameful approach to teaching software design and software testing. *Computer Games and Software Engineering*, 9:91, 2015.

[9] Swapneel Sheth, Christian Murphy, Kenneth A. Ross, and Dennis Shasha. A course on programming and problem solving. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, SIGCSE '16, pages 323–328, New York, NY, USA, 2016. ACM.