CS education in higher education

Amy J. Ko, Ph.D.
Professor, The Information School, University of Washington
June 15th, 2020

Information School
UNIVERSITY of WASHINGTON

About me

‘98–’02: CS and Psychology at Oregon State

‘02–’08: CS Ph.D. at Carnegie Mellon

‘08–’12: Pre-tenure professor at UW Seattle

‘12–’14: CTO at AnswerDash (SaaS B2B)

‘14–now: Tenured professor at UW Seattle

Note: I’m in **The Information School**, and CS by “courtesy” (all the privileges, no responsibilities)

The UW campus, where I do:

- 40% research
- 40% teaching
- 20% service to UW and academia
- 20% other things, like helping Microsoft or lobbying for CS education reform.

My CS education expertise

I've published more than 50 research papers about CS education, lead one of the largest and most prolific CS education research labs worldwide, and lead many CS education research community efforts (conferences, journals, national and state K-12 policy, doctoral student pipelines).

A summer 2019 group photo of students in my lab eating donuts without me.

I believe in impact

Research is essential to envisioning the future of CS education.

But it doesn't matter unless people know about it.

That's why I'm here talking to you, and why I blog, tweet, network with teachers, principals, department chairs, politicians, journalists.

Amy J. Ko, Ph.D. *she/her/hers*
Associate Professor
Informatics Program Chair
The Information School
Computer Science & Engineering (courtesy)
University of Washington, Seattle
CV, Commitments

Projects Lab Papers Blog Talks Classes Books **Impact** Tips Bio Funding Contact

I actively share my research and expertise with the world.

The public. The press often reports on my research, interviews me, or consults with me on my expertise.

- I consulted with APM's Marketplace Tech on automation (2019)
- I wrote for the Huffington Post about the software industry's need for computing education research (2016).
- LiveScience interviewed me about the efficacy of coding toys (2016).
- NewScientist featured my work on Frictionary (2012)
- IDG news featured our work Facebook's design culture (2012)
- Seattle's King5 News interviewed me about software failures (2010)
- PC Magazine featured my research on debugging tools (2005).
- The Associated Press featured my research on the Whyline (2004).

The software industry. My research often impacts policy and processes in software companies; I have also co-founded and consulted with startups.

- Our research on Lemonaid, our crowdsourced help tool, was the basis for AnswerDash, a company that I co-founded with Jacob Wobbrock and Parmit Chilana that offers an instant self-service answers product for web applications and e-commerce sites. The company has impacted other company's product offerings, including Zendesk's self-service technologies.
- Our research on bug reporting has been read widely in industry, including by product teams at Microsoft, Adobe, Google, and ABB.

This talk will be dire.
I decided to keep it raw, so we can wrestle with reality.
It's not a time to remain silent about injustice or exclusion in CS Education.

Five topics for today

1. Higher education
2. CS education in higher education
3. Why are CS students failing?
4. CS education during the pandemic
5. How Microsoft can help

How is U.S. higher education evolving?

A bit of history.

19th century: White, western, elitist, religious

- A liberal arts education of logic, ethics, rhetoric, literature
- A slow embrace of science (which was only a hundred years old)
- A pedagogy of discipline, repetition, and subservience
- For White wealthy men, with explicit policies excluding women, people of color, and the poor
- No computer science, just math

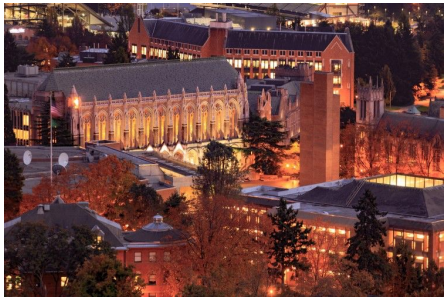
Higher education was for entry into the elite.

20th century: public, anti-poverty

- Democratization of liberal arts education
- A full embrace of science and engineering
- An unchanged pedagogy of discipline, repetition, and replication
- Majority White men, but policies shift from anti-diversity to anti-poverty, with the 1944 GI Bill and LBJ's Higher Education Act of 1965 greatly expanding 2-year colleges, grants, loans, libraries
- CS breaks off from mathematics, forming a new discipline

Higher education was for "everyone" (except Black and Indigenous Americans, and many women).

21st century: progressive but privatized



- Erosion of liberal arts and public funding
- Dominance of privately funded science and engineering
- Slow pedagogical revolution from rote learning to active learning, driven by the rise of education research
- Majority White, Asian, wealthy, but politics shift from anti-poverty to anti-racist, with a recognition of structural injustices
- As CS dominates industry, it dominates academia, becoming the most popular discipline, and begins transforming other disciplines.

Higher education becomes computational, privatized, progressive.

The future: higher education in crisis

- Public university funding has dramatically declined since Reagan.
- Pandemic-induced budget cuts and college bankruptcies are coming.
- Many project 20% of colleges and universities to go bankrupt in the next two years because of the pandemic. Some already have, some are closing less popular liberal arts and science departments.
- The only parts of academia that are fiscally resilient are CS, business, and more popular science and engineering disciplines, due to philanthropic support.

Higher education is slowly collapsing.

How is CS education evolving?

A bit more history.

1960-1980: CS is applied math for White men

- The historical foundation of CS was formal, and so theory of computation, algorithms, and data structures was the core
- Pedagogy was problem-centered and unguided, expecting students to be independent, clever, and resourceful
- Educational technology was mainframes, timesharing, punch cards
- Women, students of color, and especially Women of color were deterred from participation by elitism, sexism, racism, and safety concerns of late night lab use
- Innovations and graduates of CS begin to transform the U.S. government and white collar industries.

1980-2000: CS is engineering for White + Asian men

- CS expands to include software engineering, human-computer interaction, graphics, NLP, robotics, security, privacy
- Pedagogy unchanged: lectures and problem sets with little direct instruction, feedback, or support.
- Educational technology was desktops and command lines, but research began on IDEs, more learnable programming languages
- Women and students of color are still deterred from participation by elitism, sexism, racism, and divorce of CS from problems of the world
- CS and its graduates begin to dominate U.S. economy, creating demand for graduates, and new revenue streams for CS.

2000-2010: CS is the internet

- The rise of the internet from its government roots to disruptive economic force begins to transform higher education CS.
- Pedagogy unchanged: lectures and problem sets with little direct instruction, feedback, or support, but all in larger classes.
- Educational technology was labs, IDEs, and professional programming adapted for CS education
- Women and students of color are still deterred from participation by elitism, sexism, and racism, but CS departments begin to accept their role in exclusion and start to attempt with structural change.

2010-2020: CS is everything

- The dominance of CS in industry translates to the dominance of CS in academia, transforming all other disciplines.
- 25% of students at many campuses are CS majors, increasing teaching at scale from 50 students to 100's, 1,000's.
- These trends lead to an explosion in CS education research on methods for teaching, assessment, broadening participation.
- Educational technology is laptops, web services, APIs, Stack Overflow, automated testing and grading.
- Broadening participation efforts begin to inspire women and students of color to pursue CS; progress is slow but measureable.

Why are students "failing"?

Five discoveries from 20 years of CS education research.

1. Students believe programmers are born, not made

Most students come to CS with a “fixed mindset”, believing that some people are born with the capacity to code, others are not.

CS classes and CS faculty tend to reinforce this mindset, causing students with “growth mindsets” to permanently change to a fixed mindset (not just for CS, but for all subjects).

Fixed mindsets erode motivation, persistence, resourcefulness, and interest, explaining most of the attrition from CS classes and programs.

2. There’s little room for students’ interests in CS

The culture of academic computer science prizes abstraction, cleverness, and novelty, leaving little room in the curriculum for the many passions that students bring to CS (health, education, social justice, etc.)

Even if students can learn to code, they can’t see how programming will help them solve the problems in the world that they care about, so they leave.

3. Students face sexism, racism, ableism, elitism

Many students find a way to pursue their interests in CS, and thrive in programming, but encounter sexism, racism, ableism, and a daily barrage of microaggressions by faculty, staff, and peers.

The lack of women and people of color in CS, as well as the culture of independence, also limits the availability of mentorship to help minoritized students navigate the daily sense of exclusion.

4. Students don’t understand programming languages

The core pedagogy in CS is 1) lecture on syntax, 2) give problem sets, 3) grade problem sets.

This pedagogy does not work because it is missing two essential things: 1) direct instruction on language semantics, 2) feedback on misconceptions about language semantics.

Some students learn despite this because they find peers and TAs who provide this direct instruction in lab sections, but marginalized students often don’t feel safe seeking this support from arrogant, privileged, overachieving peers.

5. Classes move too fast

Many introductory programming courses now include a 1) professional-grade programming language, 2) a professional-grade IDE, 3) a professional-grade version control system, and 4) a professional-grade test framework.

Evidence shows that this is far too much, but that 1) faculty believe that “real” programmers will be able to learn quickly, and 2) students believe that authentic tools are more valuable to their career prospects. This overwhelms students’ capacity to learn.

How is CS education changing in the pandemic?

Some emerging trends.

Direct effects of the pandemic

- Pivot to online learning
- Hiring freezes
- Lost internships

These direct effects are rippling through CS education in a few specific ways.

Trend #1: amplification of demand for CS education

Students already viewed CS as one of the only paths to a financially stable future for themselves and their family.

Now they view it as the *only* path, as all other tenuous industries are collapsing, increasing enrollments and pressure to scale teaching.

This increases pressure to succeed. For example, I oversee admissions for our *Informatics* degree (data science, databases, software engineering, UX design). I rejected 600 students in April; one died by suicide and one attempted.

Trend #2: amplification of inequity

Students who relied on in-person college for:

- Campus housing as a refuge from unstable homes
- Computer labs
- Informal, serendipitous peer learning and mentoring
- Physical and mental health services

… no longer have these supports.

This is eroding students' wellness, hope, and motivation, all of which make learning and teaching harder than ever before.

Trend #3: amplification of poor teaching skills

Unlike K-12 teachers, most CS faculty have 1) no professional education in teaching or learning and 2) few opportunities to learn from each other.

Instead, their teaching skills are developed slowly through experience, which means many faculty are ineffective teachers.

The pivot to online learning has unmasked this lack of skills, further reducing the quality of teaching and learning, exacerbated by the lack of higher education professional development of teaching skills.

This poor teaching further erodes students' motivation and learning.

Trend #4: scale erodes quality of feedback

Assessments are how teachers learn what their students know. In CS, they have typically come in the form of exam or homework-based problem sets.

To support teaching at increased scale, and to avoid open-book online exams, much assessment is now automated tests on problem sets.

More immediate, automated, impersonal feedback lowers quality of learning and reduces willingness to reach out for help from TAs and instructors.

Trend #5: CS realizes that it is broken

Across academia, some CS faculty are realizing:

- CS is a sexist, racist, ableist, elitist community
- CS has accrued substantial power during the erosion of public funding and liberal arts, but it has not wielded it responsibly
- CS needs liberal arts more than ever, at a time when public support for and student interest in liberal arts is at an all time low.

This view is still a minority one, but it has taken hold, and is starting conversations amongst leaders about how to self-reform.

How Microsoft can support CS educators?

Microsoft has a history of CS education support

It's centered diversity and inclusion in its hiring and product mission (e.g., recent funding of CREATE center on accessible technology)

It has funded TEALS, which has grown the capacity of high schools across the U.S. to teach AP CS classes

It has partnered with hundreds of not-for-profits to develop interest in CS amongst youth across America

What's next?

Idea #1: Equitable tools for teaching at scale

Faculty need:

- More learnable, reliable, configurable programming languages and tools to effectively pace learning.
- Autograding that not only provides more personalized, effective feedback, but also classwide insight on misconceptions
- Help identifying inequities in their materials and problem sets (e.g., accessibility issues, unfamiliar domain contexts).

Some research on the topics above is mature enough for products.

Idea #2: More authentically social learning

Too many developer tools assume a single user, with collaboration mediated by version control, where there's a presumed relationship between collaborators.

But when students work together, they increasingly *don't* know each other. How can tools not only coordinate work, but develop trusting social relationships between students, TAs, and instructors to facilitate peer learning?

Idea #3: Faculty visibility into inequity

Most faculty have little insight into who is struggling most into their class, because the ones that struggle are most silent.

Tools can make struggle visible at scale, and help faculty open supportive dialog with students who are struggling, providing them additional support, guidance, and teaching.

Idea #4: Support teacher professional development

Faculty need to know more than CS to teach CS; they need robust knowledge about how to teach CS (what education researchers call *pedagogical content knowledge*). Faculty currently have nowhere to gain this knowledge, and no time to gain it.

Microsoft, in partnership with state governments, should be supporting the hiring of CS education faculty and the creation of CS education classes to prepare effective CS teachers for K-12 and higher education classrooms.

Idea #5: Advocate for saving higher education

CS needs all of academia to thrive, but most of academia is in decay.

CS needs to represent everyone, but participation is narrowing.

Use your individual and corporate influence to advocate for stable funding for public K-12 and higher education, as we briefly had 1950-1980. The quality of teaching should be more than adequate, it should be *excellent*, as it is in countries that prioritize education funding.

Without this, there soon won't be a CS to support.

Questions?

For more resources, and for pointers to the vast research literature on these topics that support the claims in this talk, see my Computing Education Research FAQ for a (growing) reading list: http://faculty.washington.edu/ajko/cer. If you want stories supporting the claims, read #Blackinthelvory on Twitter.

Key points:

- Higher education is collapsing
- Higher education CS is broken, but growing faster than ever
- Students mostly fail to learn because of oppressive cultures and structures, but also because CS faculty are low-skill teachers and lack the tools and time for teaching better.
- The pandemic is revealing these failures and amplifying their harm.
- Microsoft can help by creating more configurable, social tools, and advocating more strongly for public support of higher education.