Requirements of Oppression

Amy J. Ko, Ph.D.
Professor
The Information School
University of Washington, Seattle, USA
1992-1994

I discover code

I first discovered programming in my 7th grade pre-algebra class. My teacher required us to purchase TI-82 graphing calculators and taught us how to write simple formulas. My classmate shared a version of Tetris, after a summer of trying to make it fast enough to be playable and I was hooked.

At the time, requirements were tacit to me: software was supposed to support my creative expression, whatever that happened to be. I knew what I wanted and didn’t need to write it down.
I discover requirements

After a few years of making software for myself, my friend and I decided to make a game. He was illustrator and sound engineer, I was developer. But we couldn’t afford graphic design tools, so I set out to make a bitmap editor for tiled graphics. I didn’t talk to him at all about what he needed—I just made something, and found out later he never used it because it didn’t meet his needs, but he didn’t want to my feelings.

By ignoring my friend’s needs, I had discovered requirements, and the need to define them based on stakeholder needs.
I discover CS research

Throughout college at my alma mater Oregon State University, I learned about programming languages, operating systems, software engineering, and more. But I used none of this in what I built by myself or with others, since requirements were always given to me. Disillusioned by the constraints I perceived in industry, I pursued research, where I could continue my creative expression.

I came to view requirements as something imposed by teachers, organizations, and markets: they were a constraint on my creativity.
While earning my Ph.D. at Carnegie Mellon, I learned that there’s more to software engineering than I was taught: it was full of fascinating technical and social problems. I began to contribute technical solutions and social insights, and continued to do so throughout my first several years of faculty life.

Throughout this decade of research, I came to see HCl+design methods as key approaches to eliciting draft requirements, but requirements engineering as a necessary process of specifying them to ensure conceptual integrity.
I discover industry

Just prior to earning tenure, I took leave for 3 years to do a software startup as CTO. I designed and engineered our stack, made strategic business decisions, set requirements, managed a team of 9 engineers and designers, and directly engaged in sales. While I was always pretty sure that the hard parts of engineering were people, now I was convinced: everything hard was in sales, marketing, management, and requirements elicitation.

I began to view requirements not as purely technical but sociotechnical constraints.
I discover myself

Post-tenure, post-startup, I finally had some time to deal with my gender dysphoria. But as I accepted myself as transgender, and then came out, it became immediately clear how often software was not designed for me, and was even designed against me, leading people to deadname me, misgender me, bully me, and even physically harm me through medical errors.

I came to realize that software requirements were more than sociotechnical—they were value judgements about how the world should be, and those judgements were often oppressive.
As I learned to live oppressed by sexism and transphobia, I watched as those far more marginalized than me were even more disregarded and oppressed by software. People without internet access were excluded entirely from COVID tests and vaccines. Black people in the U.S. were surveilled, arrested, beaten, and killed, fueled by racially biased software and data.

I began to see requirements as not just value judgements, but as instruments of dehumanization, modeling people as form-fillers, threats, and perpetrators instead of people.
Requirements are not contracts, constraints, or user needs. They are social infrastructure that reflect, reinforce, and amplify the matrix of oppression.
Our roadmap

- Three cases on requirements and oppression from history
  - Gender
  - Ability
  - Race

- Imagining anti-oppressive requirements engineering
Background

I changed my name in 2019. I have what many trans people call a **deadname**. For many trans people—including me—when we read or hear our deadname, we feel pain, grief, regret, disrespect, and erasure, much like if someone called a cis person by a name that wasn’t theirs. Seeing my deadname usually ruins my day.
The requirements failure

IEEE *refuses* to fix my name in my publications, causing me emotional harm every time someone deadnames me in a citation, but also separating me from my professional history, robbing me of credit for my prior work, and putting me at risk of harassment.

For example, 300+/500+ papers cite this paper by my deadname.
Let’s play a root cause analysis game of “5 whys”...

Why won’t IEEE fix the articles that deadname and misgender me?
“Digital libraries were designed to store immutable information for printing.”

Why immutable?
“Research libraries represent a history of knowledge and history cannot change.”

Why include names + pronouns in this immutable history?
“Names and pronouns are also part of history”

Why would persisting obsolete names that refer to no one be valuable?
“Changing a name would violate a principle of historical accuracy”

Why is accuracy more important than respecting people’s names?
“Because the only people whose names are worth respecting are those whose don’t change.”
“Author names can’t change”

Underlying this central requirement was an oppressive idea: that names shouldn’t change, that those who change their names shouldn’t have their names respected, and that any harm authors experience from having their name ignored—including violent transphobic threats—is less important than “historical accuracy”.

Requirements of Oppression — Dr. Amy J. Ko, Ph.D. — IEEE RE 2021
Was it a missing requirement?

No. It was a **value judgment**. Consider the 1990’s, when IEEE started building its digital library: people changed their names for all kinds of reasons beyond gender, including marriage, divorce, religion, safety, adoption, immigration.

The IEEE requirements engineers ignored that obvious reality, and chose to prioritize a view of names as “history” over a view of names as mutable pointers. It was plainly exclusionary.
How do we know?

I’m part of the Name Change Policy Working Group, which includes trans scholars who have been advocating for name change policies across academia.

Nearly every publisher resisted, arguing that it wasn’t a missed requirement; they argued that respecting names was less important than preserving “history”.

Requirements of Oppression — Dr. Amy J. Ko, Ph.D. — IEEE RE 2021
Change required activism

We eventually won the war: ACM, IEEE, and two dozen other publishers eventually announced policies, though most are implementing them slowly and poorly, and refusing to change citations. This oppressive requirement, then, was a plain rejection of anyone who changes their name to be credited for their work. This is the matrix of oppression, manifested as a software requirement.

Requirements of Oppression — Dr. Amy J. Ko, Ph.D. — IEEE RE 2021
TRANS RIGHTS ARE HUMAN RIGHTS
The HTML alt attribute
Background

Every day, ~30 million blind people globally use the internet to live their lives. They rely on an access technology called a **screen reader** to translate the textual and visual content of the web into synthesized speech.
The requirements failure

Most developers do not write HTML in a way that is compatible with screen readers. For example, one egregious example is not writing alt text on images, meaning that most of the <img> elements on the web are invisible to people who are blind, excluding them from most news, art, and culture.
Let’s play 5 whys again...

Why don’t developers write alt attribute text?
“Because they don’t have to; pages render regardless.”

Why don’t developers have to?
“Because the standard doesn’t require alt attributes.”

Why aren’t they required?
“Per HTML 2.0, *alt* was for ‘processing constraints or user preference.’”

Why “processing” and “preference”? 
“The 90’s internet was slow, and browsers needed to show something while images downloaded.”

Why not require text?
According to W3C, the internet was getting faster, so alt attributes solved a temporary performance problem.
“Descriptions are optional”

In the 1990’s, Tim Berners Lee, browser vendors, and web developers viewed image descriptions as “nice to have” for people on slow connections, but ultimately a short term issue, because eventually everyone would be able to get images quickly.

Image: IMG

The IMG element refers to an image or icon via a hyperlink.

HTML user agents may process the value of the ALT attribute.

Attributes of the IMG element:

ALIGN
alignment of the image with respect to the text baseline.

- `TOP` specifies that the top of the image aligns with the text baseline.
- `MIDDLE` specifies that the center of the image aligns with the text baseline.
- `BOTTOM` specifies that the bottom of the image aligns with the text baseline.

ALT
text to use in place of the referenced image resource.

ISMAP
indicates an image map (see section Image Maps).

SRC
specifies the URI of the image resource. (23)

Examples of use:

<IMG SRC="triangle.xbm" ALT="Warning:" > Be sure to read these instructions.

<a href="http://machine/htbin/imagemap/sample">
<IMG SRC="sample.xbm" ISMAP>
</a>
Was it a missing requirement?

No, it was a **value judgement**. Blind people existed in the 1990’s when the standards were being developed. HTML and the web could have required **alt** attributes, but it didn’t. Berners Lee, browser vendors, and developers all simply disregarded the needs of the blind community, creating an internet ecosystem that made the visual web inaccessible.
How do we know?

Because after a decade of advocacy by blind and low vision developers, the `alt` attribute became required in HTML 4.01 after blind developers demanded it! Image tags without one will not pass a validator. And yet still, browsers render images without them, IDEs warn about them at best, developers largely ignore these warnings, and <1% of images on the web contain a description.
Change required activism

This change in HTML’s requirements required blind users of the web to organize and advocate, forcing W3C to include them. And yet, general disregard for people with disabilities—persist in every other layer of implementation of the HTML 4.01 specification, especially browsers.

Once again, the matrix of oppression was manifested as a software requirement.
"INJUSTICE ANYWHERE IS A THREAT TO JUSTICE EVERYWHERE."

Martin Luther King, Jr.
U.S. police use of facial recognition
Background

For the past 25+ years, police in the U.S. state of Florida have submitted images to a federally-funded state database that links faces to crime data. This software is used to identify people who have potentially violated laws.
The requirements failure

However, the data and algorithms used, while relatively accurate for the faces of White people, are highly inaccurate for Black people. The result is that every 3 days, someone in Florida is falsely identified, arrested, and jailed, are never told how they were identified. Many cannot afford bail or lawyers, and so stay in jail. Most are Black.
Let’s play 5 whys again...

Why is Florida’s recognition algorithm less accurate for Black faces?
“The training data lacked a sufficient number of Black faces.”

Why is the training data lacking?
“The private company who makes it didn’t gather data on Black faces.”

Why not?
“The police and state bought the software independent of its accuracy on Black faces.”

Why wasn’t this a factor?
“The state was convinced by aggregate accuracy measures, which hid any systematic bias.”

Why wasn’t bias a factor?
“Policing in Florida isn’t about justice, it’s about arrests, politics, and white fear of Black people.”
“Accuracy must exceed [n]%”

Accuracy, especially in the 1990’s, and even today throughout machine learning, is seen as an aggregate measure, not a disaggregate measure. The company sold on that metric, and Florida bought on that metric. This requirement assumed that accuracy would be comparable across all groups, independent of their race, gender, or ability. This wasn’t true at all.
Was it a missing requirement?

No, it was a **value judgement**. When Florida built the system, the goal was not equal treatment, but police funding, which was tied to closing cases by making arrests.

Uses of facial recognition could be less biased, but it’s not, because police departments and white majorities in the US do not see fairness to Black people as a requirement in criminal justice.
How do we know this?

For years, Black communities across the United States, as well as the UK, have been fighting legal battles over the use of facial recognition by police. And in a few cities, it has been banned. But in most, there is fierce resistance from technologists and white majorities, who want to prioritize a sense of safety over racial justice.

The Algorithmic Justice League (ajl.org) helps communities organize against oppressive algorithms and data.
Change requires activism

This underlying culture of racism therefore embeds itself in the requirements of facial recognition software, reinforcing and amplifying racist criminal justice outcomes, leading thousands of innocent people to be trapped in broken criminal justice systems, all because of a flawed application of machine vision.
BLACK LIVES MATTER.
Requirements are not contracts, constraints, or user needs. They are social infrastructure that reflect, reinforce, and amplify the matrix of oppression.
Requirements reflect and reinforce oppression

What these cases show is that software oppression derives from the broader systems of oppression in which we live: sexism, transphobia, ableism, racism, nationalism, xenophobia, and capitalism all find their ways into the concepts and visions that shape software requirements, resulting in software that serves majority groups at the expense of the marginalized.

Oppression is intersectional.

Credit: Rupert Millard.
Requirements engineering by and for majorities

If it’s largely cis heterosexual non-disabled White and Asian men developing, designing, negotiating, selling, and marketing software— and they are largely eliciting requirements and data from that same demographic following capitalist goals—it is inevitable that software is going to reflect their values.

Google U.S. hiring demographics
The result is software that works for those who hold power, and often harms, excludes, and oppresses everyone else.
Principles for anti-oppressive requirements engineering
1. Center the margins

Requirements engineering should focus its research, activities, and outputs on people marginalized in society, and not the majority.

This means understanding the full diversity of experiences, needs, and contexts in which people live, not just average cases.
2. Center resistance

Software requirements engineering has to be a site of resistance. It’s the responsibility of requirements analysts and developers to reject oppressive requirements, and accept responsibility for harm if they don’t.

This means engaging in social conflict with majorities, demanding change, and deprioritizing profit.
3. Center humanity

Requirements engineering has to be a discipline about people first.

This means that the software being made is secondary. In fact, anti-oppressive requirements engineering may mean refusing to engineer software that poses harm to marginalized groups, or actively advocating for the dismantling of oppressive software.
Does this sound scary?

I can say from experience, it is. My transition from bystander to activist continues to be frightening. It has thrust me into social conflict, which requires resilience, patience, and community. This talk is designed to create conflict. I suspect some of you in the audience, (especially our closing keynote speaker), is itching for conflict.
Silence is a privilege

But as a trans person, I unfortunately have no choice in this matter. And neither do the disabled people in our community, or the Black people, or the women, or the other groups that CS often disregards. If we aren’t anti-oppressive in our work and life, we aren’t free to be. We don’t have the privilege of silence and inaction, as majorities do.
What will you choose?

Will you continue to study and teach requirements engineering as a neutral practice, at the expense of people’s inclusion, safety, and freedom?

Or will you center the margins, resistance, and humanity in how you imagine software requirements, demanding software that works for everyone and hurts no one?
Let’s talk.

- Requirements are **social infrastructure** that reflect, reinforce, and amplify the matrix of oppression.

- To dismantle requirements of oppression, we must to 1) center the **margins**, 2) center **resistance**, and 3) center **humanity**.

Thank you to the students in the Code & Cognition lab for their feedback on this talk.

This material is based upon work supported by Google, Microsoft, Adobe, and the National Science Foundation under Grant No. 2031265, 1703304, 1836813, 1539179. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.