MODULARITY, BUG //
An Anthropological Examination of Two Core Computer Science Ideas
David Gray Widder, Carnegie Mellon University
2023 Meeting of the Society for Applied Anthropology
A bit about me, and what I’ll talk about today…

I’m a Doctoral Candidate in the School of Computer Science at a top tech school named after noted union crusher and philanthro-capitalist Andrew Carnegie and his banker.

I’ve used ethnographic methods (eg, participant observation, interviews, workshops, etc) before at NASA and Microsoft Research. But at Intel Labs, I was mentored by anthropologists Dawn Nafus and John Sherry, where I learned to theorize my findings more deeply.

Today, I’ll problematize two core constructs in computer science from an anthropological perspective:

What does Software Modularity do to ethics?

What is a Bug?

Discussing this work with my Software Engineering colleagues can be awkward but productive!
Modularity is a technical and social practice that makes it easier to disavow harm.

The ethos of Software Modularity is:

A technical practice: users of your module need only understand its external interface but not internal workings, minimizes friction in reuse of “general purpose” code bits.

A social practice: allows “bracketing off” relations outside the module, allows division of labor and supports an imaginary of how organizations ought to be organized.

Software systems are composed of existing modules, but developers:

Rely on upstream datasets and “fundamental” models, but disavow and rarely scrutinize their flaws.

Release what they build openly, for anyone to use for anything downstream, while disavowing these uses.

More basic capabilities

Dataset of Faces

Facial Recognition Model

Facial Recognition Doorbell

More specific uses
Participants accept responsibility for their module, but not how it is used.

**More basic capabilities**

**Technique to regularize model accuracy**

- Model “benchmarks”, “showcases”, “demos”

**VR Training Software for Department of Defense**

**More specific uses**

- "a procedure […] a new way to optimize your machine learning model and depending on the data set you use, the application domain you pick can be potentially endless"

- "nothing that would concern me [except] general ways in which you can abuse machine learning:"

- "there is a very little interest in the […] the meaning of translation, but rather [more interest in] the performance numbers"

- "an engineer working [in the] machine translation area, he or she is aware of […] the bias"

- "It’s a concern to me because there could be flaws in the code, security risks, quality risks, and effectively, if anything goes wrong, it looks bad on us."

- "We’re not going to have a random [person] buy our products and begin using it. There’s always going to be some level of […] customer qualification"

- "I get to turn a blind eye to certain social aspects, because we have program managers that tend to be the buffer [between us and the user]"
Lucy Suchman helps us Locate Accountability.

Responsibly developing tech must be “a boundary-crossing activity, taking place through the deliberate creation of situations that allow for the meeting of different partial knowledges”

Requires a shift “from a view of design as the creation of discrete devices, or even networks of devices, to a view of systems development as entry into the networks of working relations”

What holds ethics together is outside of the modularized supply chain: personal and company reputation concerns, delivering value to end users, seeing them as people.

What if we thought of a chain of modules as something that enables a view from somewhere, to see where action can take place?

This situates even relatively “general purpose” AI libraries or frameworks in the context of the downstream harms they potentiate or constrain.
Three Ways Forward…

Work within the modules?

AI Ethics Interventions (model cards, datasheets, toolkits) must delineate labor, support appending partial knowledges

“supply chain” metaphor

Strengthen module interfaces?

Bidirectional communication, thicker social ties between module creators and users.

Accept and leverage opportunities for partial control, even if not complete, such as ethical licences.

“Value Chain” metaphor

Reject modularity?

Radically reimagine software development. Build relations first, technology secondarily, scalability last if at all.

Distinctions between software producer and user soften or dissolve.

Indigenous Data Sovereignty, and “critical technical practice” (Agre 1997)
Responses from Software Engineering Professors

“You may have well told me Jesus isn’t real”

“Can you modularize ethics?”

Modularity as dominant ideology, subsuming and organizing other concerns like ethics

“Where does it end?” Can you not use compilers? Existing hardware?

“Modularity ‘manages complexity’. How else would we build large systems?”

Push towards scale. Assumption that software must be built.

Are there structures for the production of software that satisfy needs of modularity and relational approaches address?

“Working misunderstandings” borrowing from legal anthropologist Paul Bohannan

Reject modularity?
What is a “Bug”? 🐜

Ongoing work with Claire Le Goues, a computer science professor who does “Automatic Program Repair” research (can we make computers fix their own bugs?)

Work often relies on testing new approaches against standard benchmarks containing known bugs in a codebase.

**Objectivity:** P values, hypothesis testing, “science”!, quantify this

Bug is objectively definable, so that detection and repair can be automated

But, researchers sometimes **don’t agree** on benchmarks.

Some think that benchmarks can be satisfied, but the bugs/fixes are **not useful**, and therefore push for **human subject studies**
David is moving to @davidthewid@hci.social
@davidthewid

@clegoues and I spent the last 45 mins in lively conversation, primarily arguing what a software "bug" is.

Q: #SoftwareEngineering practitioners and researchers: How would you define "bug"?

3:02 PM · Sep 19, 2022

5 Retweets 5 Quotes 13 Likes
“In my PhD dissertation, I defined a software bug as "any error or flaw in the implementation of a software system that causes it to produce incorrect results, exhibit undesirable behavior, or cause unintended consequences".”

“a deviation from intent”

"A friend worked on project creating a catalog of vulnerabilities for the federal gov; they started w/ engineers but ended up having to bring in epistemologists because no one could decide what constituted one and where it started or ended. Bugs are social not technical things…”

"Seeing some good definitions being thrown around here, but as a seasoned engineer, I cannot stress enough that bugs don't become bugs because they do something unintended, they become bugs because they do something unintended that gets caught or noticed.

All software is a complex tangle of executed intentions, riddled with problems. But the 'bugs' are the ones we recognize. We notice bugs into existence."
But, what is a “Bug”? 

Formally defined, a mismatch between’s a software’s specification and behavior. 

But software is always underspecified, so lots of assumptions. 

In practice: no one writes comprehensive specifications, so it’s a “you know it when you see it” type deal. 

Framing problem of Charles Frake, complexity literature. You can’t specify the state of the world and you can’t be exhaustive. 

“Job security” 

Often, people ended up talking about intent. -> **Subjectivity**!
The Epistemic Power to declare “Bug”

When something is declared a “bug”, it is a statement that something is obviously wrong.

Whose subjectivity matters in defining bugs?

Managers, Developers, Customers

Not: other stakeholders subjected to the system, or “users”.

Declaring a bug is an epistemic power move.

**Epistemic Power:** A person has epistemic power to the extent she is able to influence what people think, believe, and know, and to the extent she is able to enable and disable others from exerting epistemic influence. (Archer et.al. 2019)
Gender Bias Bug

Professor Margaret Burnett’s work recasts the definition of bug to include nuances in the design of software that might make it less gender inclusive.

This is an epistemic power move to expand common notion of bug to include things important to those less likely to be software engineers.

Uses personas to help engineers adopt the subjectivity of people different than themselves.
In my own work, I surveyed ~130 software engineers about their ethical concerns. Some described how they raised concerns about bugs that can cause ethical issues.

Eg: numerical error in crane simulation software might kill someone: ethics bug

But some bigger things: like, “I work at a military contractor and don’t like military uses of my tech”

Too big for framing of bug.

Definition of bug is situated! Depends on the power you have to affect outcomes.

“you’re actually asking to shut down the business. [...] It’s not really a concern you can raise.”

“And so when I brought that issue up [...] they did a big investigation”
Takeaway:

I believe subjectivity can make software better via an improved understanding of bugs, relationships to other software components, and ethics and representation in software.

I hope I have shown software development to be an interesting as a site of anthropological enquiry, where the kinds of subjectivities, disputed meanings, disagreements and power relations exist within CS, despite its highly rationalized, technical mode of production.


I’d love to talk, connect, or give this talk again!
dwidder@cmu.edu • @davidthewid • @davidthewid@hci.social • www.davidwidder.me

I’ll be at Cornell Tech in NYC in the fall, studying norms and privacy in AI.

Feedback and critique please! I don’t often get to present to anthropologists!