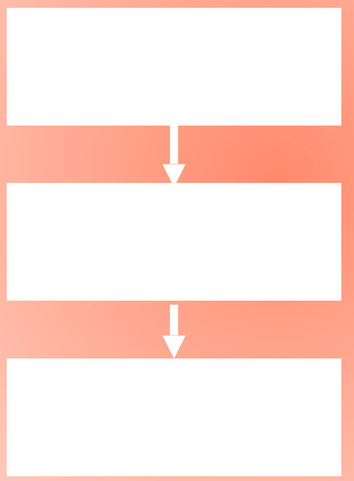
# // MODULARITY, BUG //
# An Anthropological Examination of Two Core Computer Science Ideas

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:

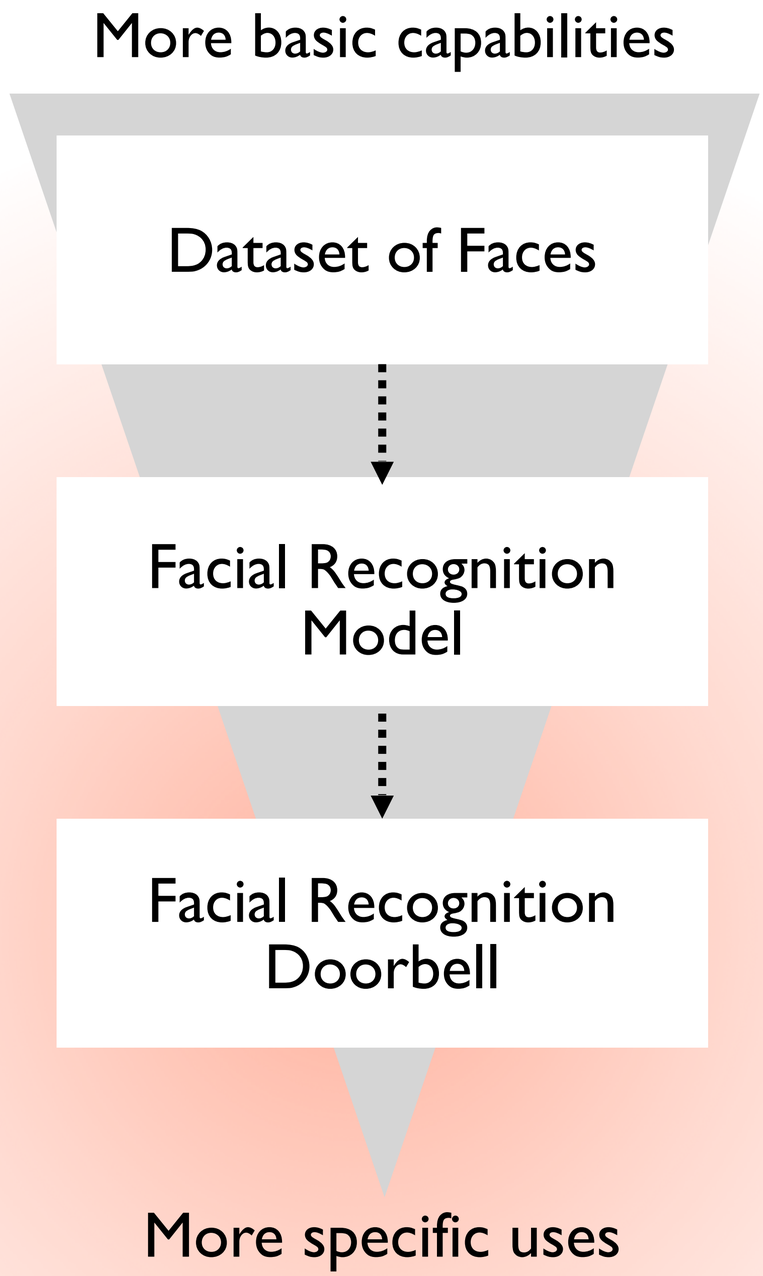
A **technical** practice: users of your module need only understand its external interface but not internal workings, minimizes friction in reuse of "general purpose" code bits

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



# Participants accept responsibility for their module, but not how it is used.

More basic capabilities

Technique to regularize model accuracy

"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**."

Model "benchmarks", "showcases", "demos"

"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**"

VR Training Software for Department of Defense

"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]"

More specific uses

# Lucy Suchman helps us *Locate Accountability*.

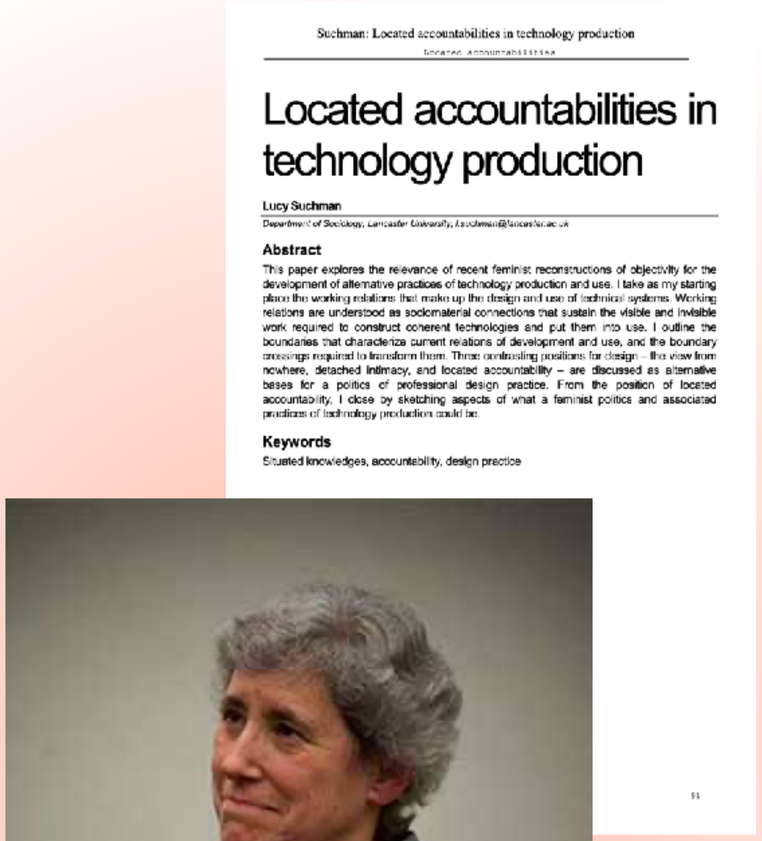
Responsibly developing tech must be “a **boundary-crossing activity**, taking place through the deliberate creation of situations that allow for the meeting of different partial knowledges”

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 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.


Suchman: Located accountabilities in technology production

Located accountabilities in technology production

Lucy Suchman

Department of Sociology, Lancaster University, l.suchman@lancaster.ac.uk

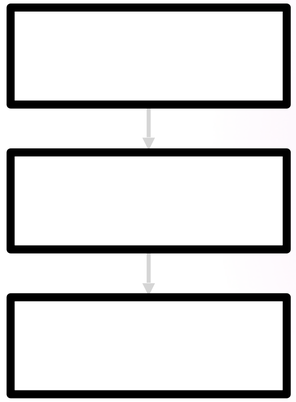
**Abstract**

This paper explores the relevance of recent feminist reconstructions of objectivity for the development of alternative practices of technology production and use. I take as my starting place the working relations that make up the design and use of technical systems. Working relations are understood as sociomaterial connections that sustain the visible and invisible work required to construct coherent technologies and put them into use. I outline the boundaries that characterize current relations of development and use, and the boundary crossings required to transform them. Three contrasting positions for design – the view from nowhere, detached intimacy, and located accountability – are discussed as alternative bases for a politics of professional design practice. From the position of located accountability, I close by sketching aspects of what a feminist politics and associated practices of technology production could be.

**Keywords**

Situated knowledges, accountability, design practice

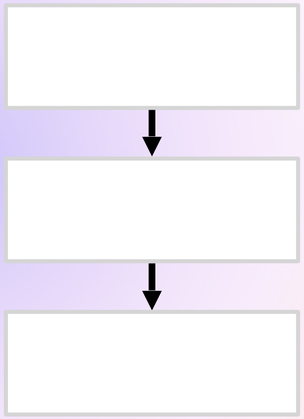
# Three Ways Forward…



## Work within the modules?

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

“supply chain” metaphor

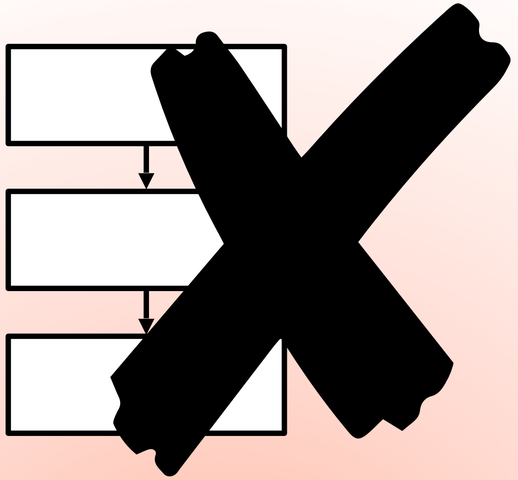


## Strengthen module interfaces?

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

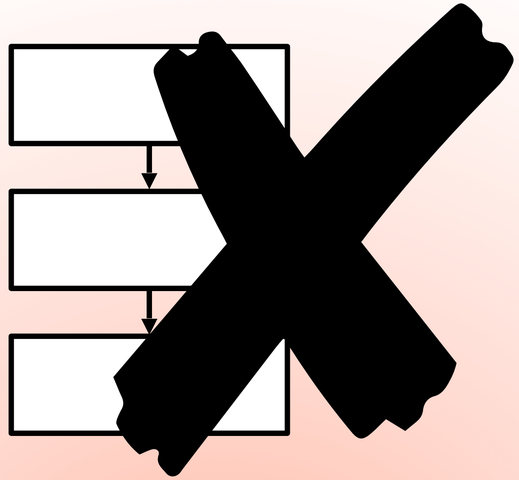
"Where does it end?" Can you not use compilers? Existing hardware?

"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

View Tweet analytics

**5** Retweets **5** Quotes **13** Likes

what is bug tweets

File Edit View Insert Format Data Tools Extensions Help

50% $ % .0 .00 123 Defaul... 10

D2 https://twitter.com/moarbugs/status/1572030327396208643?s=20&t=JBQW25

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | https://twitter.com | Name | Twitter Handle | Link | Reply or Quote T | Comment |
| 2 | 2 | Jessica Colnago | @jessica_colnag | https://twitter.com | Reply | "Q: are privacy issues ever talked about as "bugs"?"<br>idk what do you mean, most of the time they are the features being developed. <is sigh> |
| 3 | 3 | Es Braziel | @sbraziel | https://twitter.com | Reply | Q: does word "bug" have salience in design/ design research?"<br>that's a good question, I don't hear about "bugs" in the design research context at all. not sure if it gets thrown around in design more broadly |
| 4 | 4 | Rohan Padhye | @moarbugs | https://twitter.com | Reply | **In my PhD dissertation (https://rohan.padhye.org/files/phd-dissertation.pdf...), 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".** |
| 5 | 5 | Claire Le Goues | @clegoues | https://twitter.com | Reply | "colloquialism referring to a mistake in a program's source code that leads to undesired behavior when the program is executed, whether the undesired behavior is a deviation from the functional specification, an explicit security vulnerability that may be maliciously exploited...<br>or a service failure of any kind." ^^ aka a mouthful that doesn't fit into a single tweet. |
| 6 | 6 | Michael Hilton | @michaelhilton | https://twitter.com | Reply | So, like using Celsius? 😉😉😉 |
| 7 | 7 | K. Alexandria Bs | @__zeletic__ | https://twitter.com | Reply | a deviation from intent |
| 8 | 8 | Frank Elavsky | @FrankElavsky | https://twitter.com | Reply | 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.<br>All software is a complex tangle of executed intentions, riddled with problems. Bu the "bugs" are the ones we recognize. We notice bugs into existence |
| 9 | 9 | Meredith Whittak | @mer__edith | https://twitter.com | Reply | It doesn't do what we meant it to do. |
| 10 | 10 | Dawn Nafus | @dawnnafus | https://twitter.com | Reply | I've done UX studies where "is it a bug or is it a feature" was totally real— not a joke. I reported bugs that turned out to be someone's belief about what it should do. |
| 11 | 11 | Michael Coblenz | @mcoblenz | https://twitter.com | Reply | Why is this a useful discussion? Every org has its own process, and what matters is not whether something is called a bug but whether it is prioritized for fixing.<br>Designer: "The button is red. It should be green." Engineer: "The spec doesn't say what color it should be. Not a bug."<br>Designer: "It looks bad. Fix it."<br>Manager to engineer: "Just get it done."<br>Of course, it might have ended: Manager to designer: "Sorry, we're shipping tomorrow, and if you'd wanted it green, you should have said so earlier. File an issue in the issue tracker." I don't think it's worth arguing whether that was a bug. Software dev. is a human process. |
| 12 | 12 | Sukrit Venkatagir | @SukritVt | https://twitter.com | Reply | [replying to Coblenz] But only when we deem something to be a "bug" do we foreground its existence and the fact that we can/should act on it: fix it, ignore it, note it down, "it's actually a feature," etc. |
| 13 | 13 | Ian Sweet | @completelysou | https://twitter.com | Reply | http://www.pl-enthusiast.net/2015/09/08/what-is-a-bug/ |
| 14 | 14 | Robbie | @bravenewprinc | https://twitter.com | Reply | A software "bug" is any time the program does not meet the specification. |
| 15 | 15 | Jacques Carette | @jjcarett2 | | Reply | [replying to Robbie] This is it, this is THE definition.<br>Predentically, code with no spec has no bugs.<br>Realistically, all software has an implicit, undocumented spec. That spec changes over time as people's expectations changes. |
| 16 | 16 | Robbie | @bravenewprinc | https://twitter.com | Reply | [repying to Carette] <3<br>This is the book I like to reference any time I talk about software quality assurance, but I'm biased because Bill taught my college class on software quality assurance.<br>https://dl.acm.org/doi/book/10.5555/3019418 |
| 17 | 17 | Jurgen Vinju | @jurgenvinju | https://twitter.com | Reply | [replying to Jacques] Much code grows by neither rhyme nor reason and so its behavior is emergent from the incremental changes that the code has suffered. There is no spec. Not even platonically, because from non-understanding nothing rational can grow. Only accidental "correctness" is to be expected.....<br>...That's why I like @AndreasZeller's empirical definition better; first there's a (subjectively) observed failure, then there is a search for the defect in code that causes it, and then the fix confirms that it was indeed a "bug", if the failure is now avoided. No need for specs. |
| 18 | 18 | Jacques Carette | @jjcarett2 | https://twitter.com | Reply | [replying to Jurgen] If there is a "failure", then calling it a bug is easy. It's much harder when the software succeeds but returns an answer that is found to be subjectively unexpected. Is that a bug? This happens in research software all the time. |
| 19 | 19 | Jurgen Vinju | @jurgenvinju | https://twitter.com | Reply | [replying to Jacques] I really don't know a general answer to that. Good question. Especially for research software I do expect people to voice their expectations (specify) such that we can either change the spec or the code when they disagree, but not just "because"....<br>....Unfortunately that is not a realistic expectation for software in the wild. |
| 20 | 20 | Jacques Carette | @jjcarett2 | https://twitter.com | Reply | [replying to Jurgen] Worked at one of those "software in the wild" companies for a long time. Its implicit spec was simple: it does 'symbolic math'. But often users would report bugs, and the reaction was "won't fix" with a long convoluted explanation based on 20-30 year old implementation details....<br>.... Much worse were the horrors committed in the NAME of 'usability' but that were completely unusable (and not useful) shiny gizmos that made for pretty demos. |
| 21 | 21 | Michael Coblenz | @mcoblenz | https://twitter.com | Reply | [replying to Jacques, 15] Also, some aspects of the "spec" will not have been considered. "What happens when the user does X?" "Oh, I didn't think of that. Well, let's say the software should do 'Y.' "Well, it currently does Z." Is that a bug? Who cares? The question is whether they decide to change it. |
| 22 | 22 | Alcides Fonseca | @alcides | https://twitter.com | Reply | [replying to Michael] What if there are multiple stakehlders with opposing points of view? A bug for one user is not a bug for the other.... |
| 23 | 23 | Jacques Carette | @jjcarett2 | https://twitter.com | Reply | [replying to Alicides] I've seen situations where multiple developers on the same team is enough, never mind different stakeholders! Though indeed when it's contradictions between stakeholders, those bugs are thorniest. |
| 24 | 24 | Robbie | @bravenewprinc | https://twitter.com | Reply | [reolying to Michael. 21] Behavior outside of the spec is undefined. When a stakeholder's expectations don't align with the behavior, then you refine the spec. |
| 25 | 25 | Kel Cecil | @praisechaos | https://twitter.com | Reply | I generally refer to a "bug" as any time the code doesn't match the requirement or spec. I also tend to prefer "defect" over "bug". It was easier to introduce effect to describe as actua engineering misstep than to exclude requirement oversights from the term "bug". |
| 26 | 26 | ShriramKrishnan | @ShriramKMurth | https://twitter.com | Reply | Pretty sure @AndreasZeller wrote a book about it. |
| 27 | 27 | Joe Gibbs Politz | @joepolitz | https://twitter.com | Reply | When a user complains and one of the developers agees the state shouldn't have been reachable by the code that way. |
| 28 | 28 | Art Yerkes | @prozacchiwawa | https://twitter.com | Reply | any time expectations are defied in a bad way, it's a bug. |
| 29 | 29 | Alex Rothuis | @ARothuis | https://twitter.com | Reply | IMO, a bug is an instance of when the software does not fulfill the stated or implied needs of its stakeholders, often with regards to functional suitability. |
| 30 | 30 | Shrikanth, N.C. | @shrikanth_nc | https://twitter.com | Reply | Researcher: 1 or more LOC/Non-LOC that does not adhere to the intended requirement.<br>Practitioner: The additional effort required to close my pull request.<br>Where effort may be:<br>1.+ or - LOC or Non-LOC<br>2. Time (s) // to claim not my problem<br>3. Do nothing //deprecates sometime |
| 31 | 31 | danny "disco" mc | @hipsterelectron | https://twitter.com | Reply | when the code makes me mad...<br>...(i think i agree most with the answers about deviation from expectation/specification but wanted to offer a possible alternative) |
| 32 | 32 | Professional Mar | @allspiritseve | https://twitter.com | Reply | Job security |
| 33 | 33 | Elizabeth Dinella | @eadinella | https://twitter.com | Reply | Mismatch between specification (sometimes not explicit) and implementation |
| | | | | | | Definitely the go-to book on debugging and the requested definition is @AndreasZeller 's book |

“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 its 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.

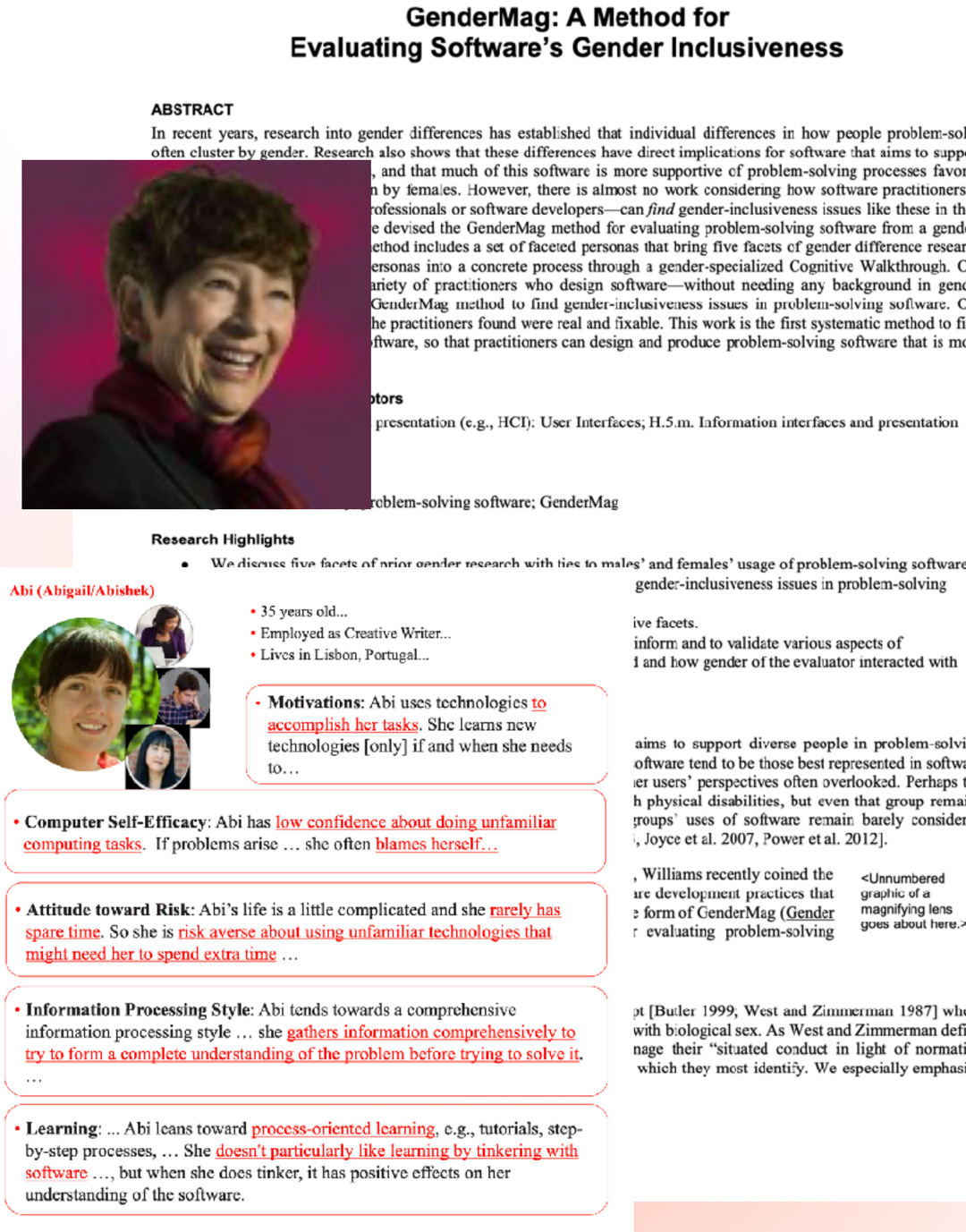
***Epistemic Power:*** *A person has epistemic power to the extent she is able to influence what people think, believe, and know, and to the extent she is able to enable and disable others from exerting epistemic influence. (Archer et.al. 2019)*

# 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 epistimic 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.



GenderMag: A Method for Evaluating Software's Gender Inclusiveness

ABSTRACT

In recent years, research into gender differences has established that individual differences in how people problem-sol
often cluster by gender. Research also shows that these differences have direct implications for software that aims to suppo
, and that much of this software is more supportive of problem-solving processes favor
n by females. However, there is almost no work considering how software practitioners
rofessionals or software developers—can *find* gender-inclusiveness issues like these in th
e devised the GenderMag method for evaluating problem-solving software from a gende
ethod includes a set of faceted personas that bring five facets of gender difference resear
ersonas into a concrete process through a gender-specialized Cognitive Walkthrough. C
ariety of practitioners who design software—without needing any background in gend
GenderMag method to find gender-inclusiveness issues in problem-solving software. C
he practitioners found were real and fixable. This work is the first systematic method to fi
ftware, so that practitioners can design and produce problem-solving software that is mo

tors

presentation (e.g., HCI): User Interfaces; H.5.m. Information interfaces and presentation

roblem-solving software; GenderMag

Research Highlights

• We discuss five facets of prior gender research with ties to males' and females' usage of problem-solving software
gender-inclusiveness issues in problem-solving

ive facets.
inform and to validate various aspects of
l and how gender of the evaluator interacted with

Abi (Abigail/Abishek)

• 35 years old...
• Employed as Creative Writer...
• Lives in Lisbon, Portugal...

• **Motivations:** Abi uses technologies to accomplish her tasks. She learns new technologies [only] if and when she needs to…

• **Computer Self-Efficacy**: Abi has low confidence about doing unfamiliar computing tasks. If problems arise … she often blames herself…

• **Attitude toward Risk**: Abi's life is a little complicated and she rarely has spare time. So she is risk averse about using unfamiliar technologies that might need her to spend extra time …

• **Information Processing Style**: Abi tends towards a comprehensive information processing style … she gathers information comprehensively to try to form a complete understanding of the problem before trying to solve it. …

• **Learning**: ... Abi leans toward process-oriented learning, e.g., tutorials, step-by-step processes, … She doesn't particularly like learning by tinkering with software …, but when she does tinker, it has positive effects on her understanding of the software.

aims to support diverse people in problem-solvi
oftware tend to be those best represented in softwa
er users' perspectives often overlooked. Perhaps t
h physical disabilities, but even that group remai
roups' uses of software remain barely consider
, Joyce et al. 2007, Power et al. 2012].

, Williams recently coined the
re development practices that
e form of GenderMag (Gender
r evaluating problem-solving

<Unnumbered graphic of a magnifying lens goes about here.>

pt [Butler 1999, West and Zimmerman 1987] whe
with biological sex. As West and Zimmerman defi
nage their "situated conduct in light of normati
which they most identify. We especially emphasi

# Ethics bug

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

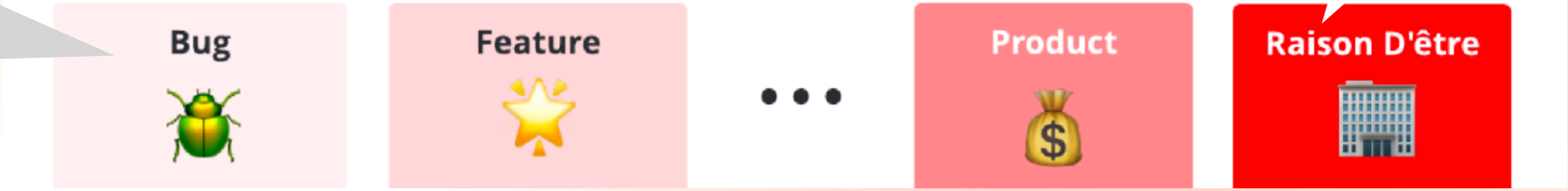
But some bigger things: like, “I work at a military contractor and don’t like military uses of my tech”

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.

davidwidder.me/supply-chain.pdf, to appear in: SAGE journal of *Big Data & Society*

**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!