

ONBOARD LIKE A PHP

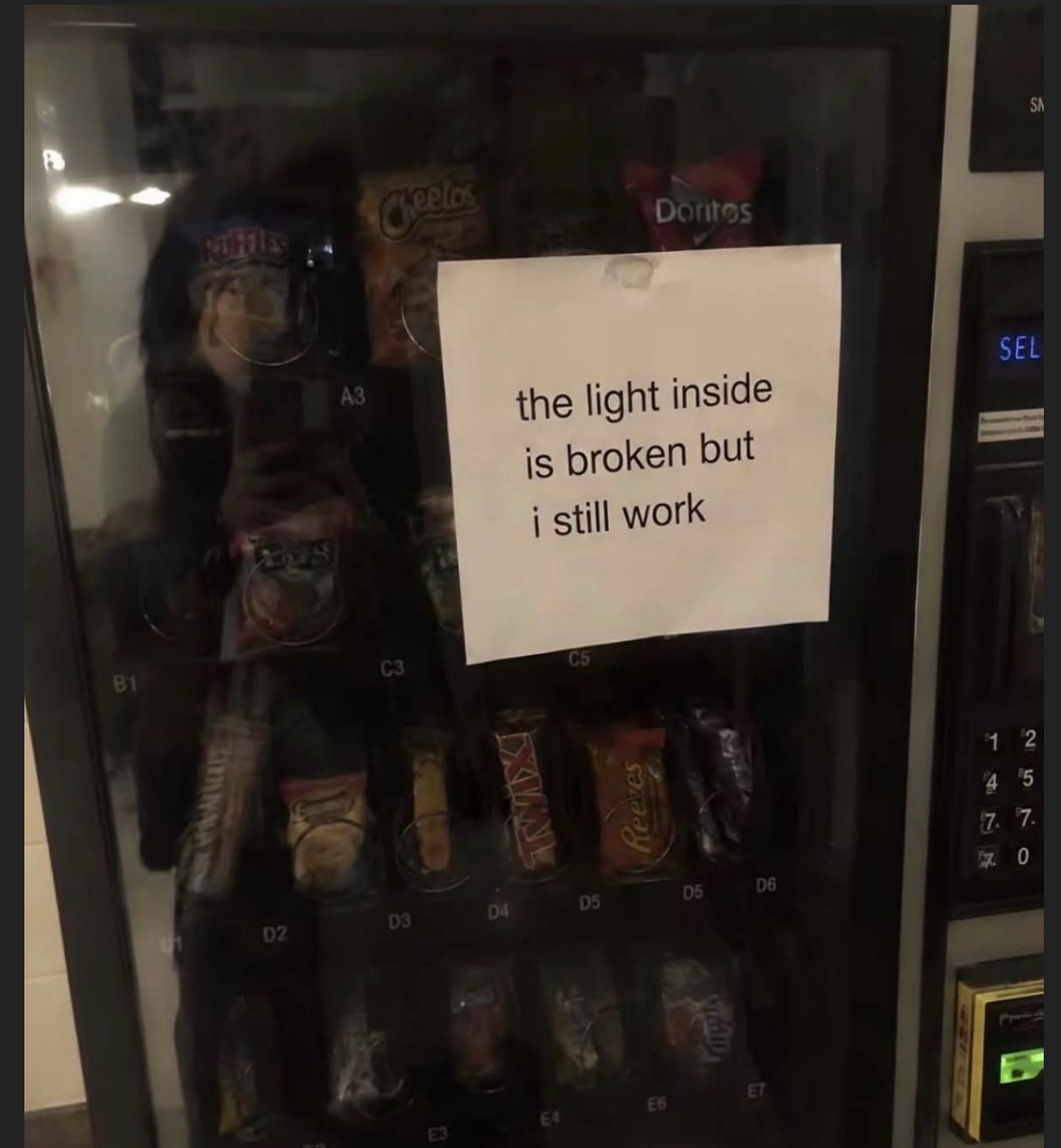
STAFF ENGINEER

AUDIENCE OVERVIEW — HI THERE!

- ▶  Career level
- ▶  Current goals
- ▶  PHP experience
- ▶  LLM attitudes
- ▶ Onboarding experiences
- ▶ Expectations for today

SPEAKER OVERVIEW — LINCOLN RUSSELL

- ▶ PHP since 2002, full-time since 2007
- ▶ Led teams since 2010, lead / exec since 2017
- ▶ Product-focused (B2B SaaS) since 2012
- ▶ Participated in >50 hiring teams
- ▶ Recipient of a 2026 NLnet grant
- ▶ PHPDetroit co-organizer once-upon-a-time



TALK OVERVIEW — ONBOARD LIKE A PHP STAFF ENGINEER

- ▶ Align on Values
- ▶ Define the Role
- ▶ Define the Process
- ▶ Discover & Assess *
- ▶ Make a Plan
- ▶ Evaluate Success



The Big Lebowski (1998)

YOU CANNOT BE HELPFUL WITHOUT CONSENT



TRUST CANNOT BE SOLVED WITH A COMPUTER



PROGRAMMING IS BUILDING A MENTAL MODEL IT IS NOT MECHANICAL CODE PRODUCTION.

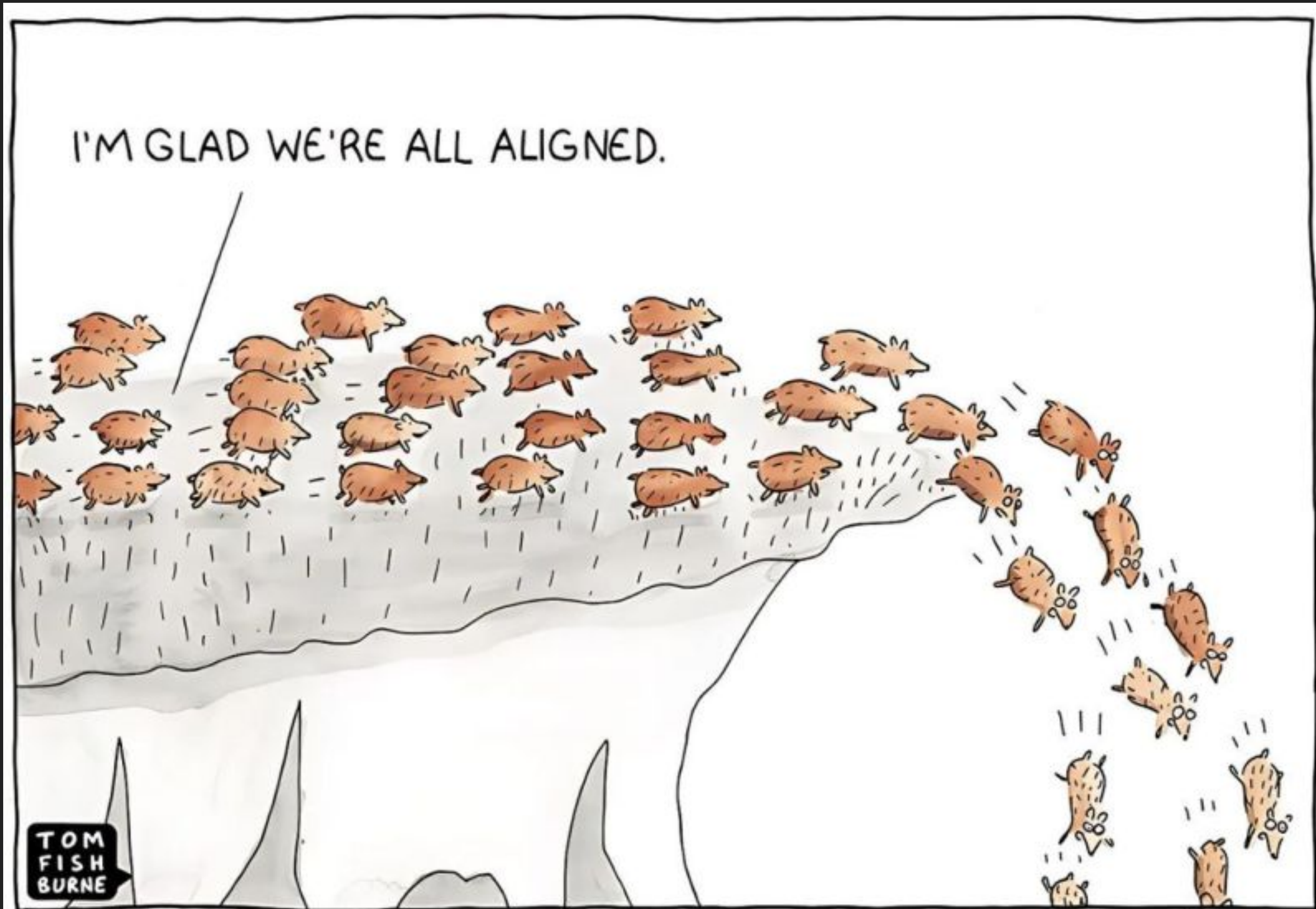
The shared mental model in developers' minds is the real product.

A program is "living" only while the team understands this theory.

When programmers leave, it becomes "dead".

Without the original theory, new maintainers treat software as a black box,
creating patchwork fixes that cause side effects and architectural decay.

–Peter Naur, "Programming as Theory Building" (1985)

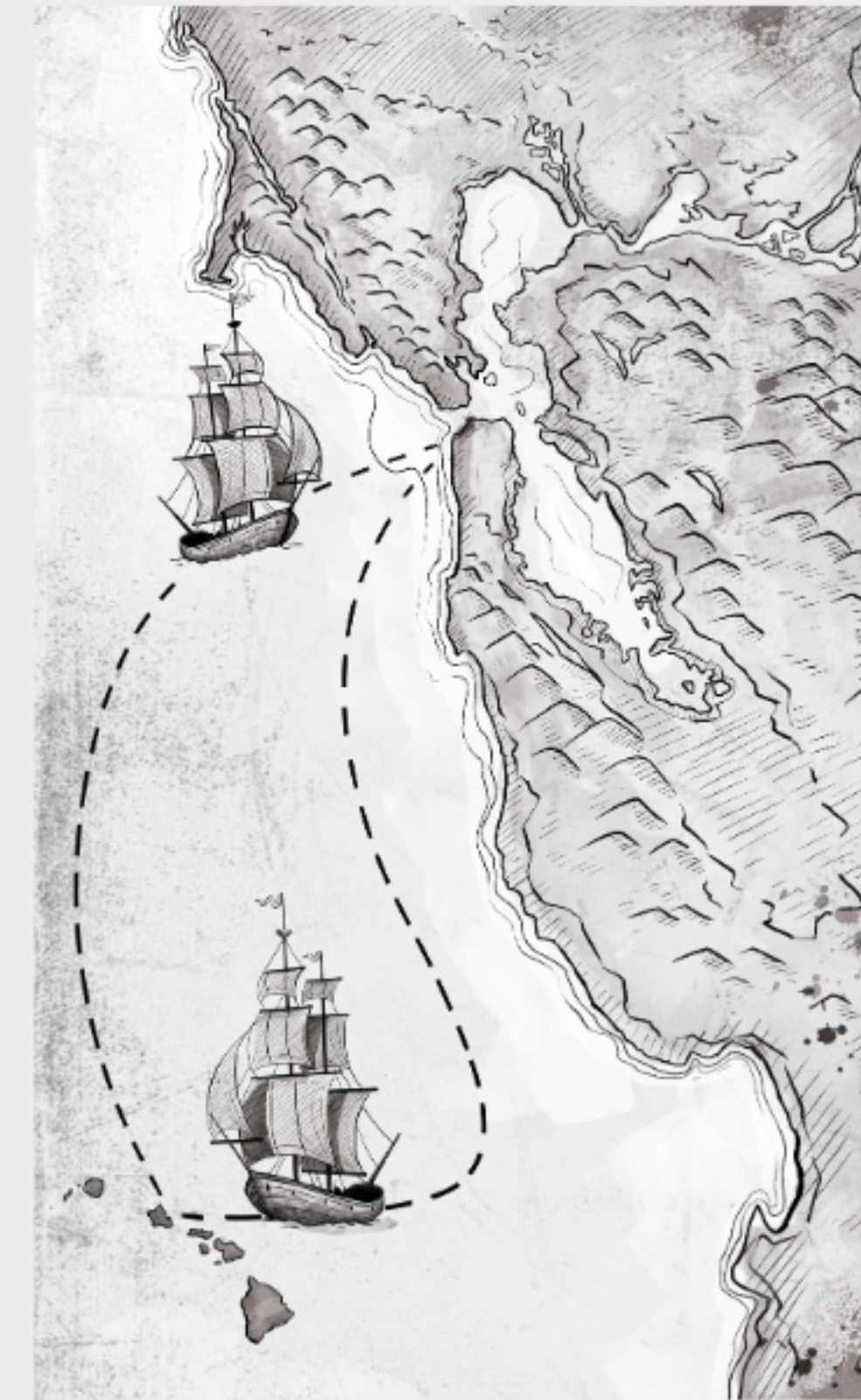


LARSON'S 4 ARCHETYPES

- ▶ Tech Lead
- ▶ Architect
- ▶ Solver

ROLE ORIGIN: MECHANICS

- ▶ Promoted vs Hired
- ▶ Vacancy vs New Role



Staff Engineer

Leadership beyond the management track

Will Larson

ROLE ORIGIN: PURPOSE

- ▶ Specific challenge requiring...
 - ▶ Experience / Training
 - ▶ Focus / Coordination
- ▶ Target STAFF : SENIOR ratio
 - ▶ Mentoring
 - ▶ Payroll balance
- ▶ "Velocity"



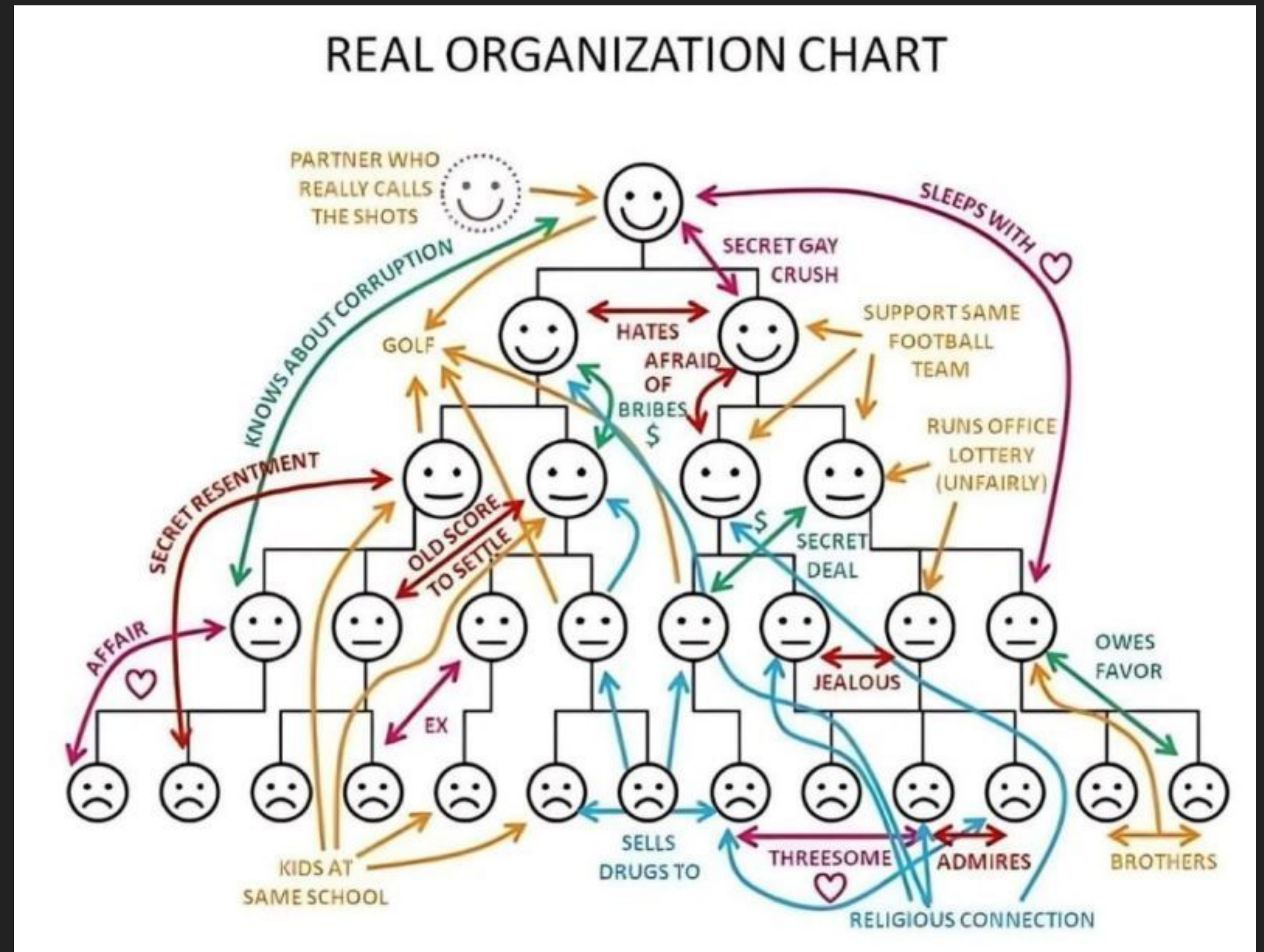
@dank@jorts.horse, 13 Apr 2026

DEFINING THE ROLE

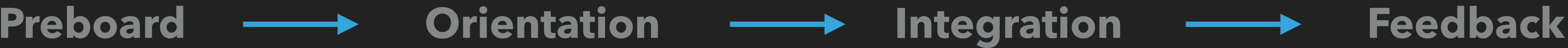
IN RELATION TO...

- ▶ Senior
- ▶ Staff++
- ▶ Cross-dept
- ▶ Board

Software developer: the only career path to learn about the business last



FORMAL STEPS



Can you start here?
Contact info?
Leading questions?

Formal process?
Duration?
Investment?

Week 1 or 4...?
Tools needed?
Access missing?

Who evaluates?
Based on what?
Who sees/cares?

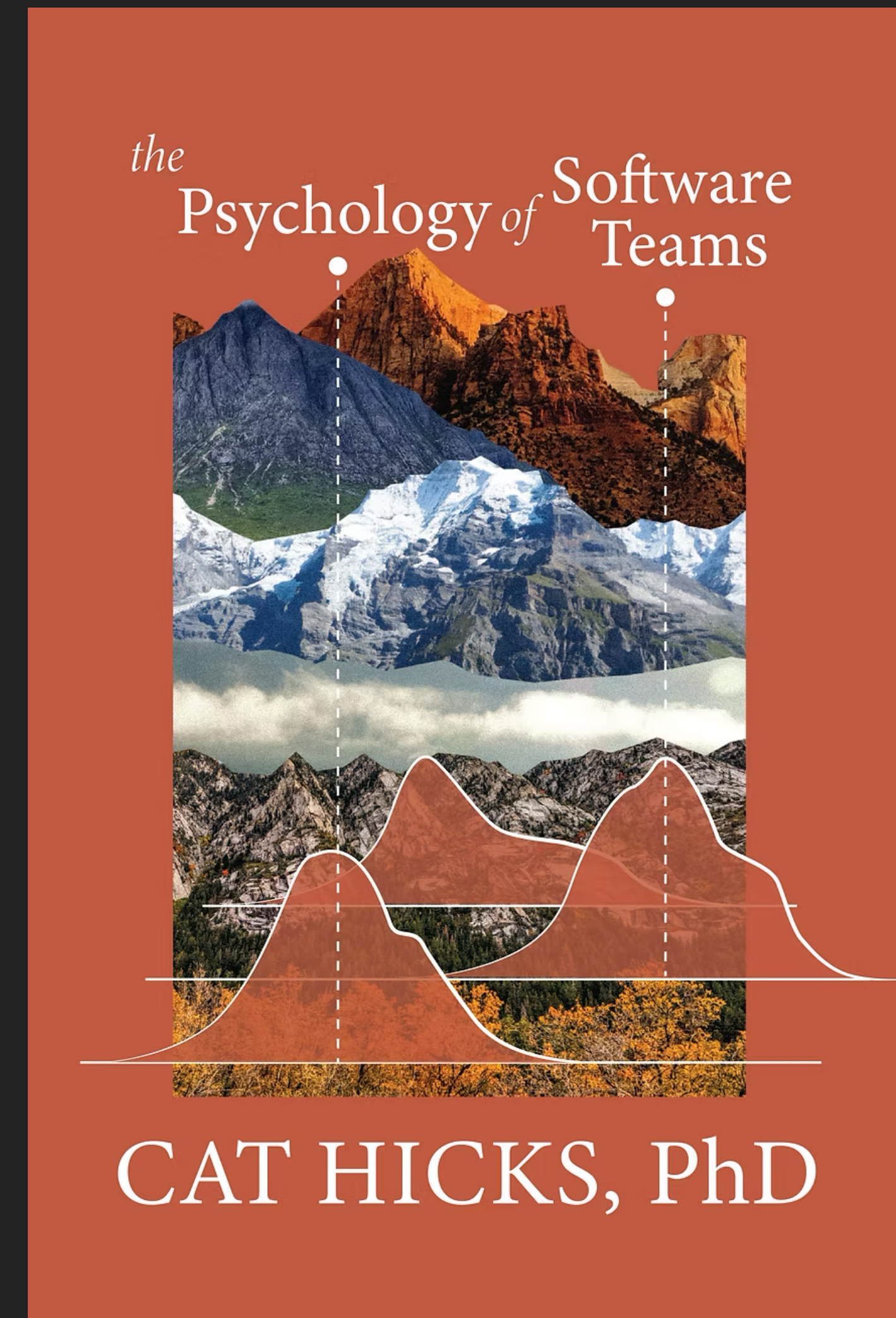
FACTORS

- ▶ Formal Curriculum
- ▶ Your Manager
- ▶ Your Team
- ▶ Which Industry
- ▶ Access Issues



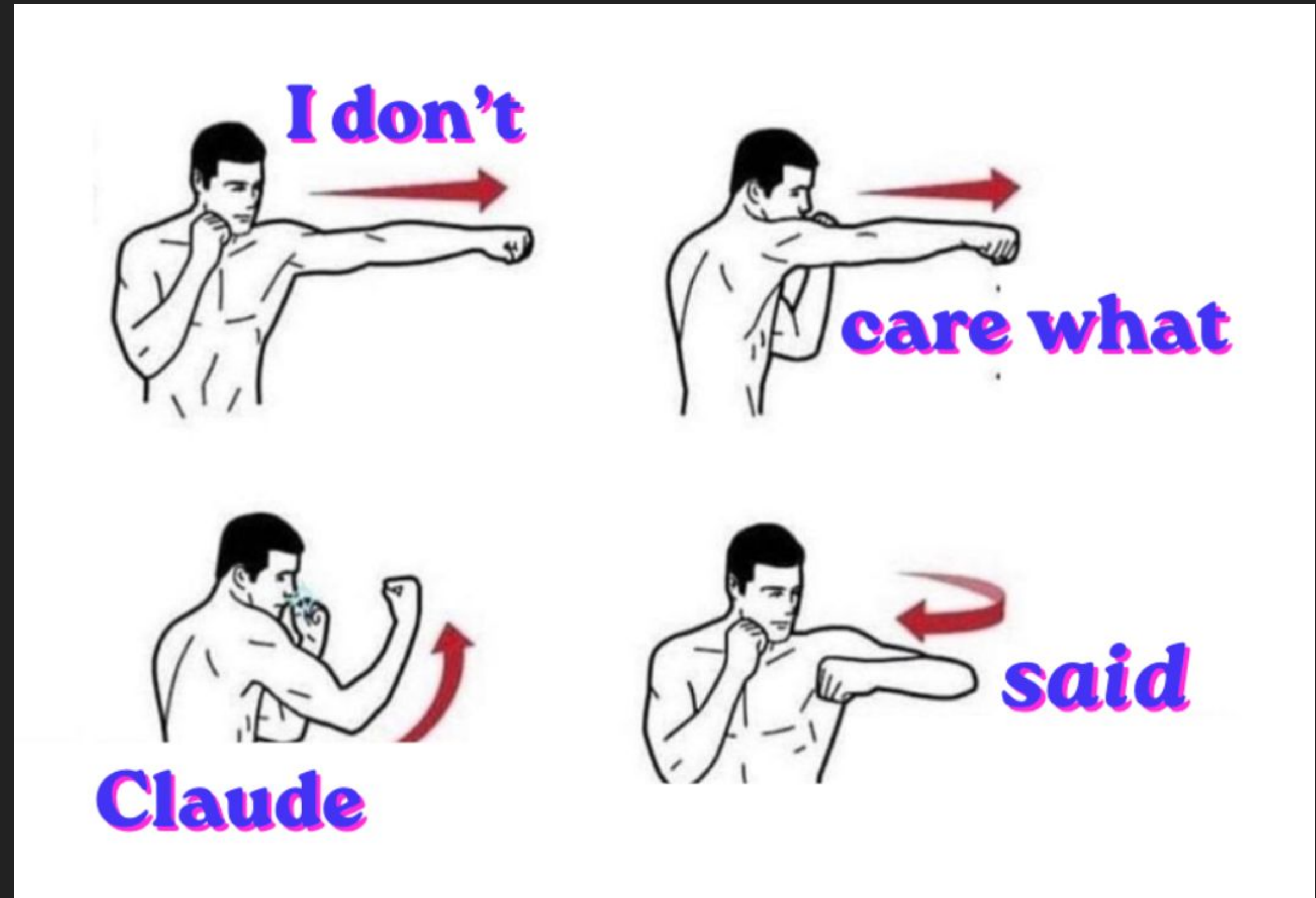
DEVS ARE NOT "BRAINS IN JARS"

- ▶ Ideas: Assumptions about what is good, powerful, effective – and why
- ▶ Institutions: Practices & policies that shape how culture is expressed
- ▶ Interactions: Informal dynamics between people, groups, & artifacts
- ▶ Individuals: Thoughts, feelings, identities, mindsets, & behavior



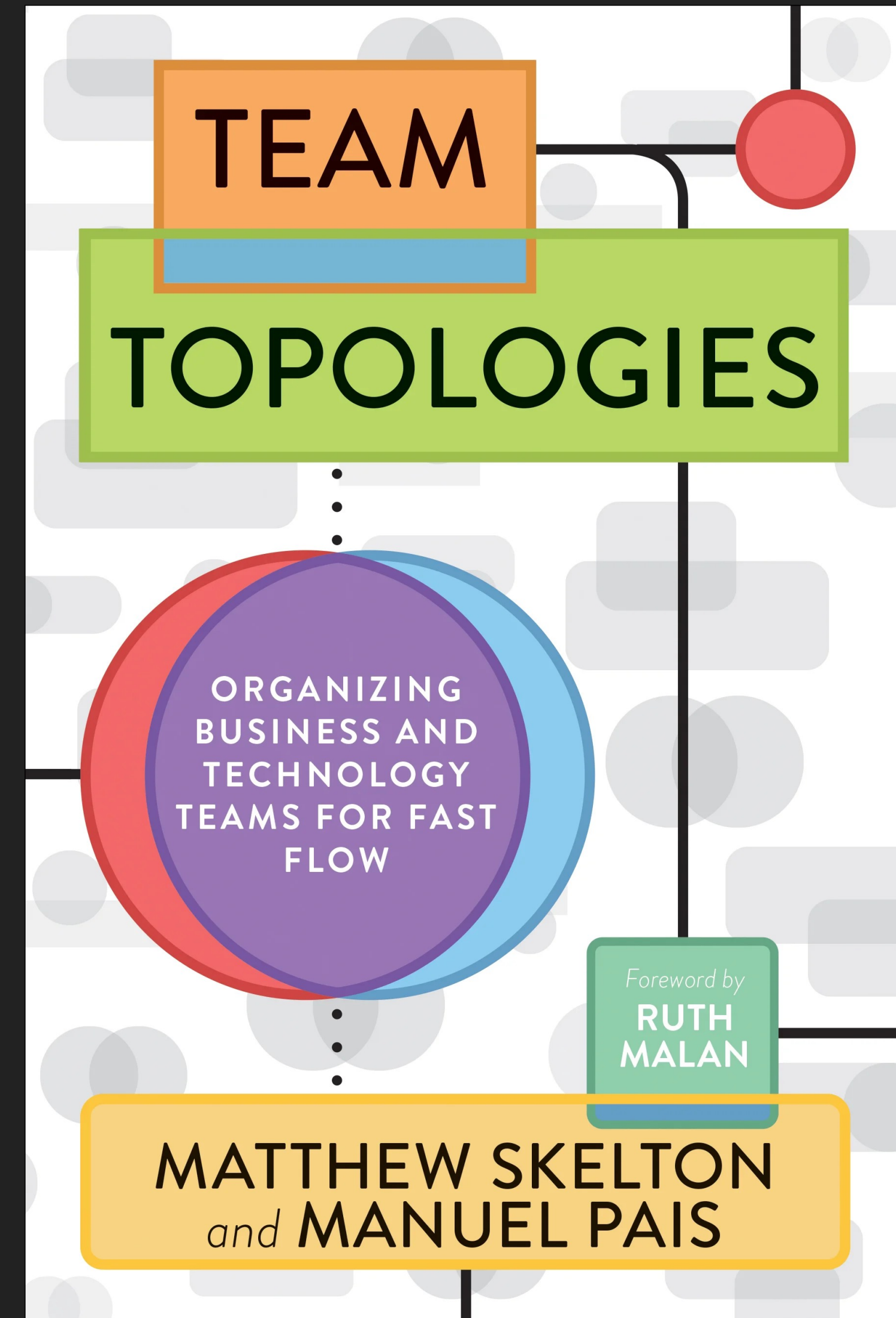
DISCOVERY OF...

- ▶ Culture
- ▶ Narratives
- ▶ Risks & Opportunities
- ▶ Liabilities & Assets
- ▶ Friction
- ▶ Knowns vs Conjecture
- ▶ Annual Rhythm



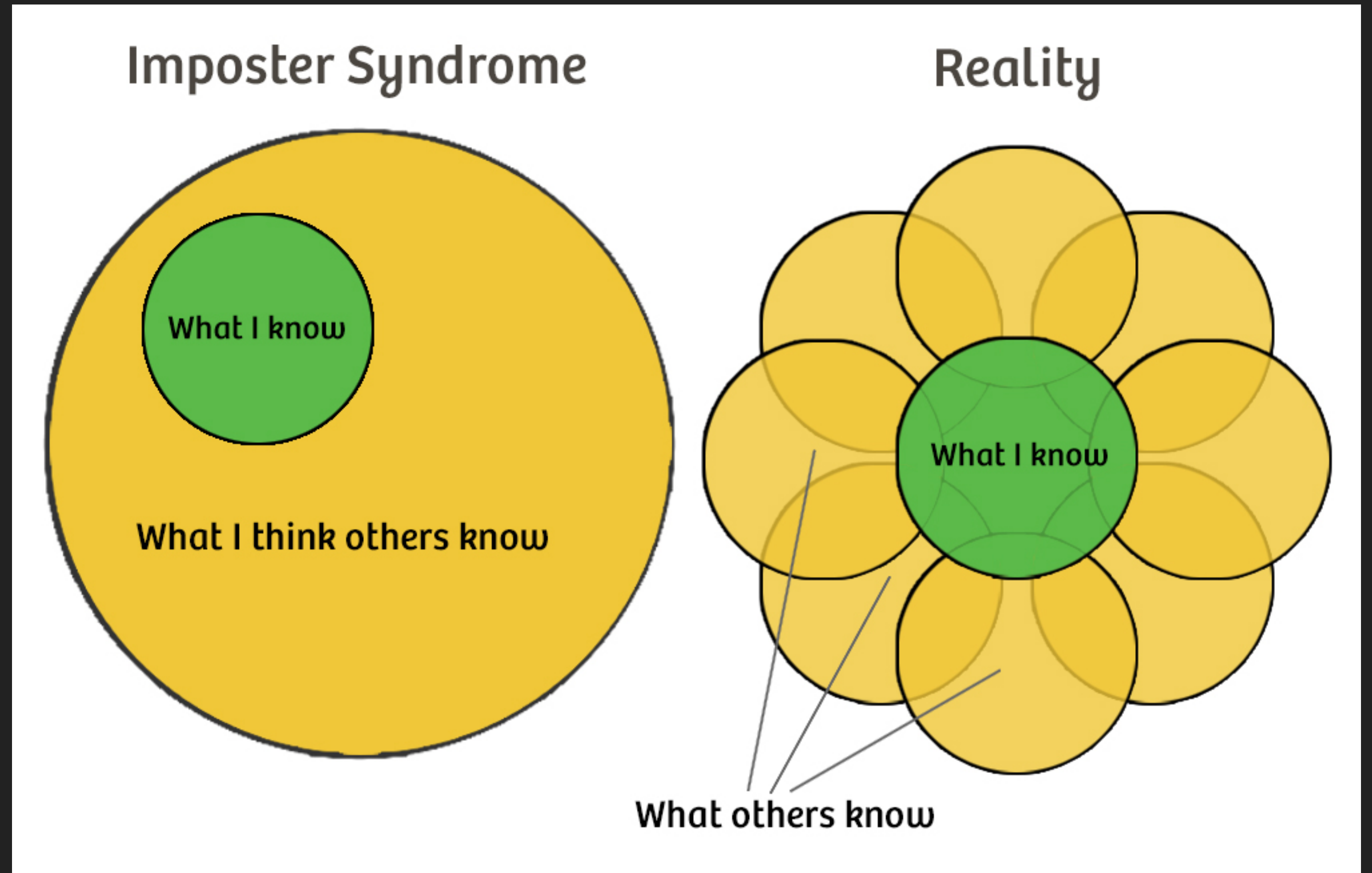
HOW DOES THIS ORG WORK?

- ▶ Team boundaries
- ▶ Recent org history
- ▶ Hidden couplings
- ▶ Controlling comms
- ▶ Awkward interactions



WHO OWNS...

- ▶ Functionality
 - ▶ Compliance
 - ▶ Effort (Priorities x Workflow)
-
- ▶ Security
 - ▶ Accessibility
 - ▶ Quality
 - ▶ Performance
 - ▶ Architecture



ARTIFACTS

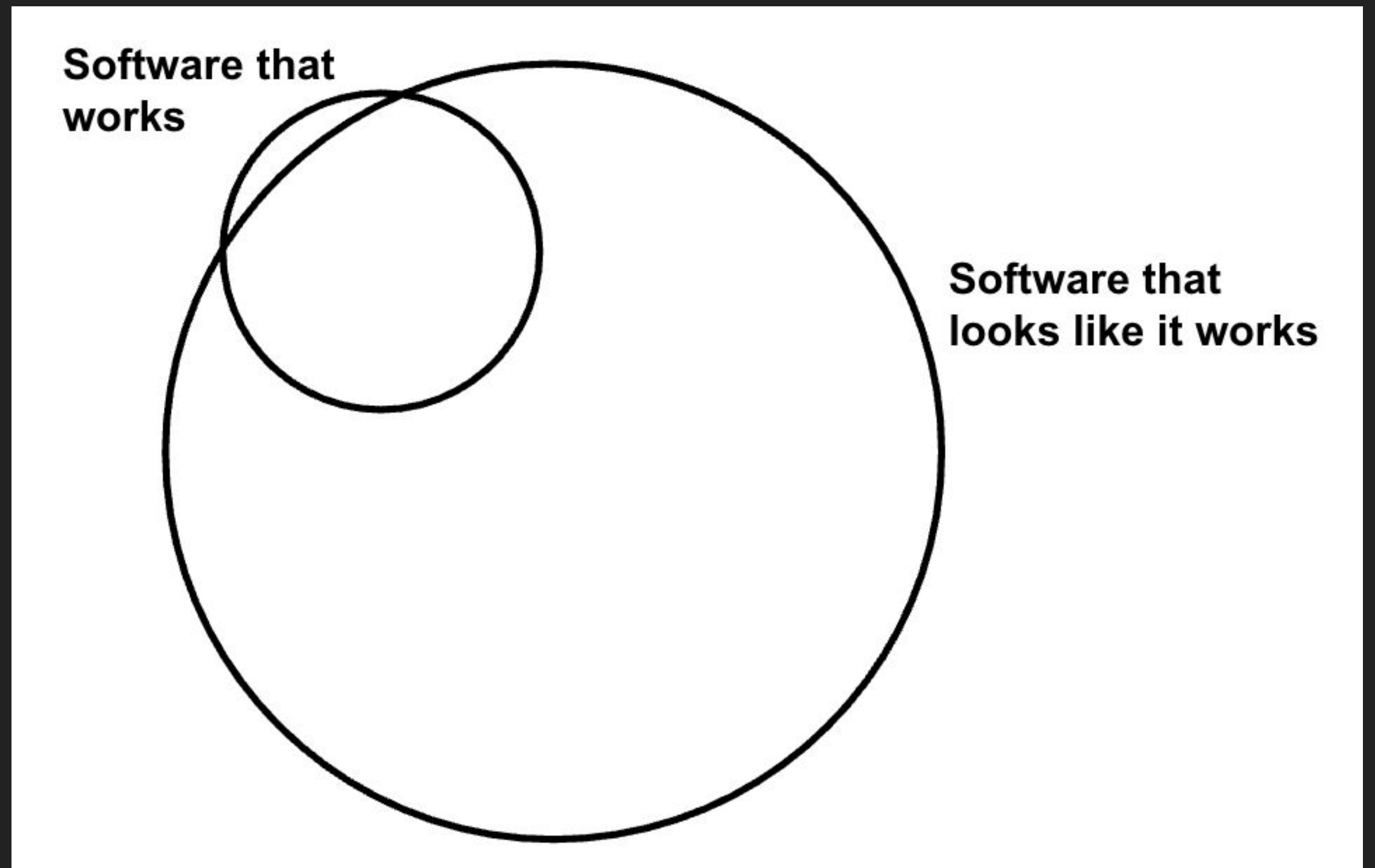
- ▶ RCAs – how is crisis managed?
- ▶ RFPs – how is change managed?
- ▶ CI pipeline reports – what's "normal"?
- ▶ Issues – who, why, and how?
- ▶ PRs – what's the code review vibe?
- ▶ Retros – how the process working?
- ▶ Docs – inline, adjacent, internal, external



WHY NOT DEPLOY NOW?

PHP STACK

- ▶ CI automation
- ▶ Dependencies
- ▶ Environments
- ▶ Coding standards
- ▶ Static analysis
- ▶ Test coverage & types
- ▶ Debugging



PHP MATURITY

- ▶ Is there a 1- or 2-year upgrade cycle? [php.net / supported-versions.php](https://php.net/supported-versions.php)
- ▶ Language level + feature adoption rate
- ▶ Where are they on journey to stronger typing?

7.4 Typed properties

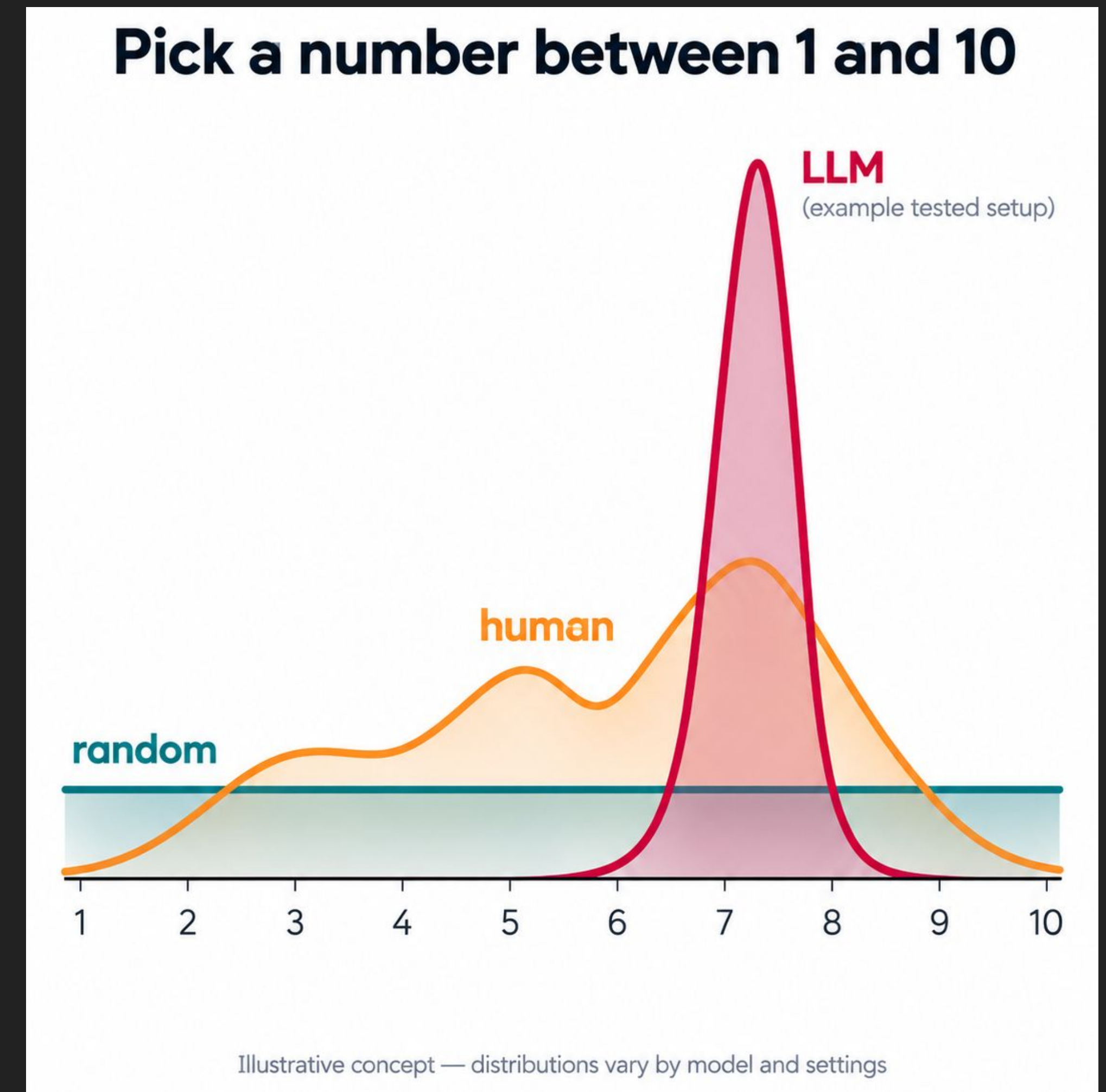
8.0 / 8.1 Union / Intersection types

8.2 Types null | true | false

8.3 Class const types

HOW DO LLMS FIT IN?

- ▶ Speed up (some) technical discovery tasks
- ▶ May help timebox onboard struggles*
- ▶ Bridge language knowledge gap
- ▶ "GenAI" also destroys mental models, reinforces bias, & induces dependency



* it can also make valuable info re: struggles invisible to folks who could fix it

TECHNICAL DEBT

- ▶ What code changes will this business need soon?
- ▶ What code changes may it need in 1-3 years?
- ▶ What code changes is it unlikely to ever need?

INTERVIEW SCENARIO

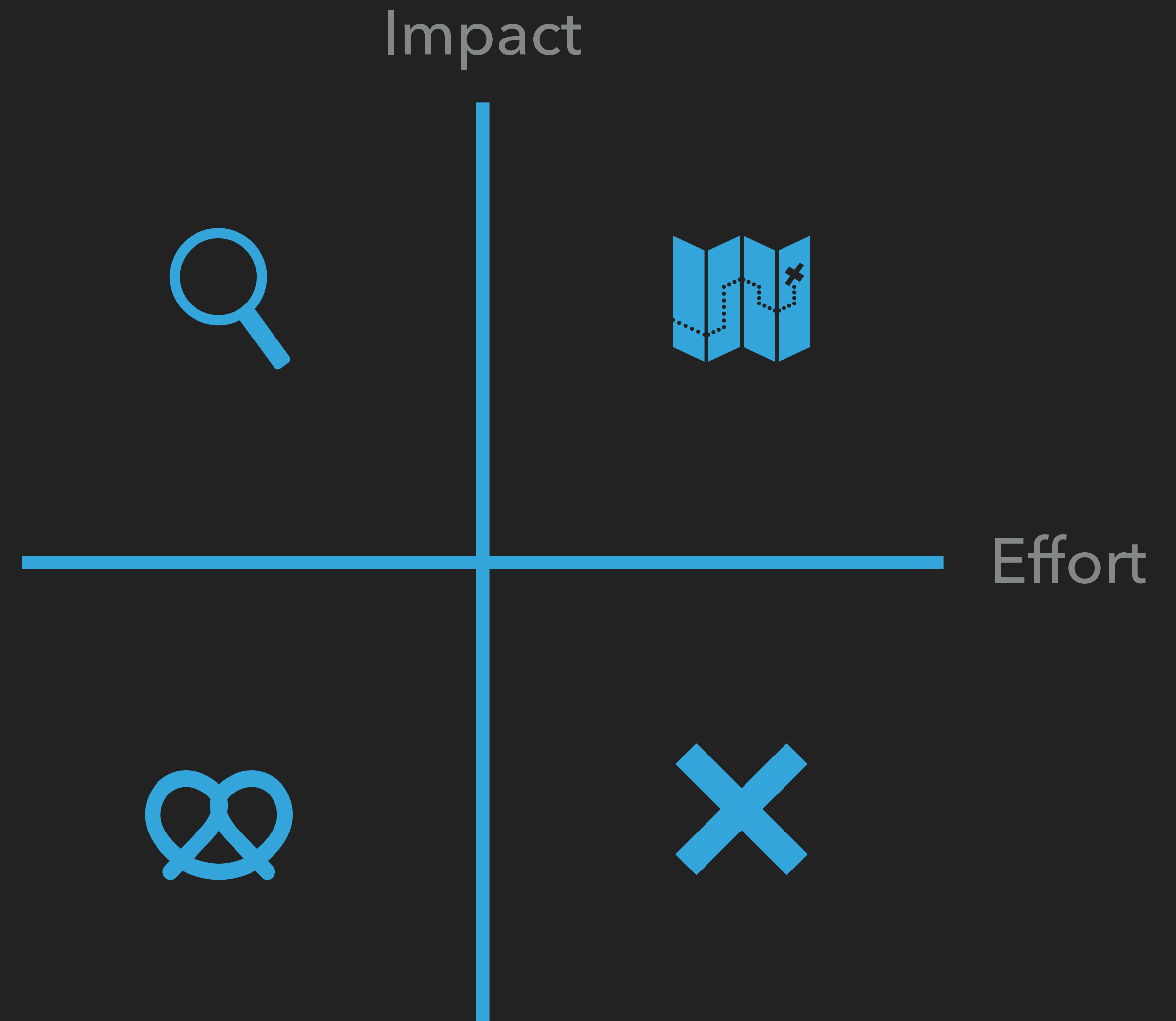
- ▶ Running a monolith with:
 - ▶ outdated PHP
 - ▶ outdated framework
 - ▶ EOL localhost config
- ▶ Goal: Resolution in 3-6 months
- ▶ What order should you address them in?

DISCOVER & ASSESS

IT DEPENDS.TM

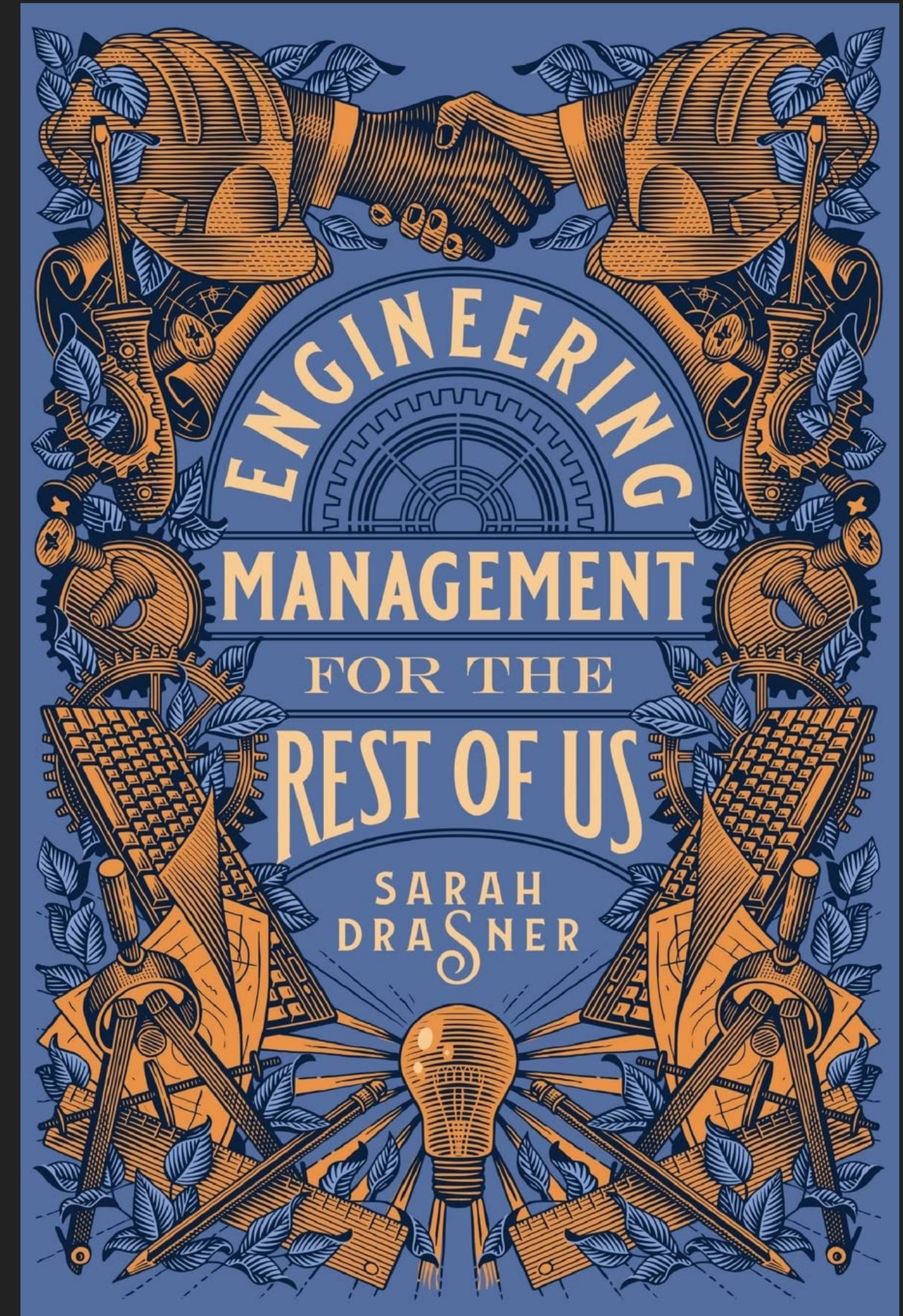
FOCUS ON IMPACT

- ▶ Impact vs Effort
- ▶ "Correct" vs Successful
- ▶ Stay Curious
- ▶ Get Guidance



ENGINEERING MANAGEMENT TOOLS

- ▶ 30 / 60 / 90 plans
 - ▶ Identify what is quantifiable
 - ▶ Goals vs Needs
- ▶ Force multiplier vs solo missions



WHAT CAN WE EVALUATE?

- ▶ Quantifiable
- ▶ Metrics
- ▶ Unquantifiable



PERFORMANCE REVIEWS

- ▶ Technical
- ▶ Teamwork
- ▶ Alignment



Kevlin Henney
@KevlinHenney

The act of describing a program in unambiguous detail and the act of programming are one and the same.

2:50 PM · Aug 17, 2009

EX CRITERIA: TECHNICAL > PROFICIENCY > TOOL MANAGEMENT

Upgrade existing dependencies and navigate complex challenges found in intertwined dependency trees.

Build tools used by the rest of the team to automate repetitive or sensitive tasks.

Update best practices based on closely following the wider community's framework use and socialize them internally.

Control code deployments thru a well-maintained CI/CD pipeline that is iterated on to address challenges and flaws.

Deeply understand frameworks used internally and consistently demonstrate best practices for its use.

Researches new technologies and vendors as required.

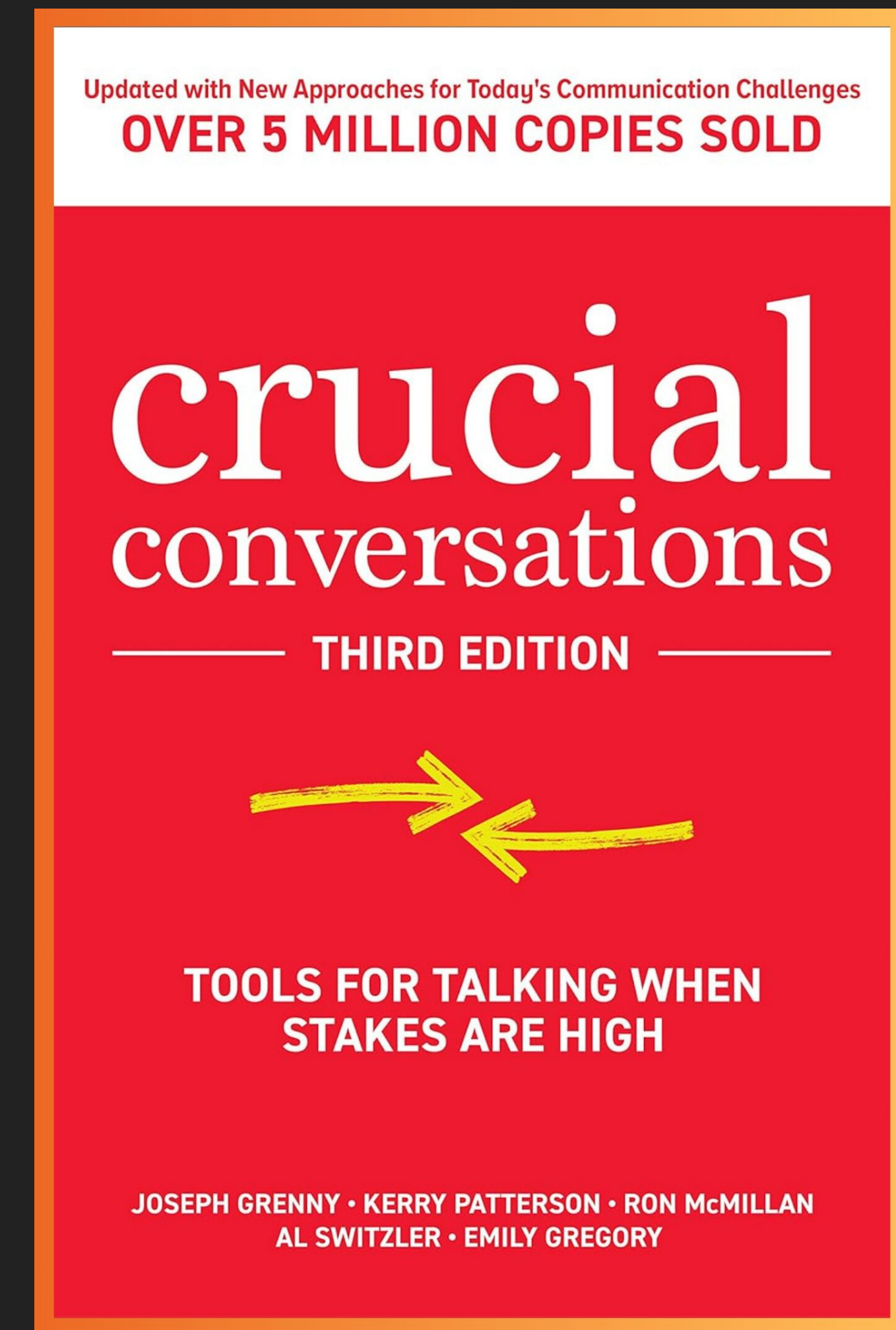
Full evaluation criteria: [lincolnwebs.com / eval](https://lincolnwebs.com/eval)

INTERNAL WORK

- ▶ Evaluate yourself
- ▶ Accept slower feedback
- ▶ Empower decisions without you
- ▶ Build a network of peers

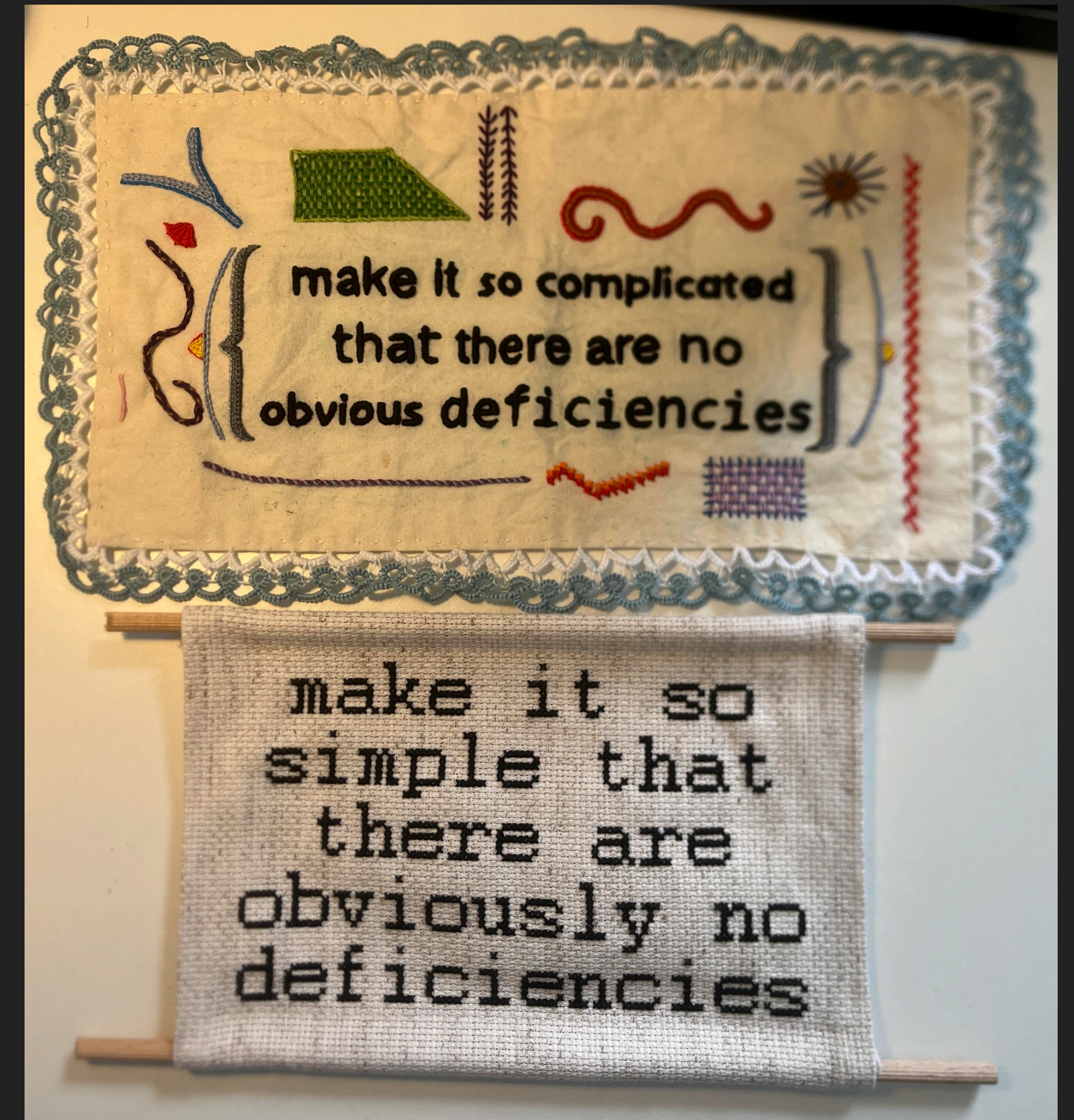
WHEN THINGS ARE... LESS SUCCESSFUL

- ▶ Retrace your steps
- ▶ Contrast to fix misunderstanding
- ▶ Create mutual purpose
- ▶ Assess your emotions
- ▶ Avoid "clever stories"



FINAL THOUGHTS

- ▶ Find artifacts to align faster and stay curious
- ▶ Assess state of PHP tools & environment / org
- ▶ Keep notes as minutes, journaling, docs++
- ▶ Make a plan then iterate as you learn
- ▶ Communicate intentionally in all directions
- ▶ Evaluate yourself to frame performance



FEEDBACK, CONTACT, & QUESTIONS

- ▶ [https:// joind.in / talk / a7170](https://joind.in/talk/a7170) or →
- ▶ php @ **lincolnwebs.com**
- ▶ phpc.social / @ **linc**
- ▶ linkedin.com/in/ **lincolnwebs**
- ▶ 🙋 *Did you agree / disagree on anything?*
- ▶ 🙋 *What surprised you or was unexpected?*
- ▶ 🙋 *What do you wish I'd covered (more)?*



FURTHER READING, BY RELEVANCE

- ▶ **Staff Engineer:** Leadership beyond the management track – Will Larson, 2021
- ▶ **The Psychology of Software Teams** – Cat Hicks, 2026
- ▶ **The Engineering Leader:** Strategies for Scaling Teams & Yourself – Cate Huston, 2024
- ▶ **Engineering Management** for the Rest of Us – Sarah Drasner, 2022 (also: career-ladders.dev)
- ▶ **Peopleware:** Productive Projects & Teams (3rd Ed) – DeMarco & Lister, 2013
- ▶ **The Art of Leadership:** Small Things, Done Well – Michael Lopp, 2020
- ▶ **Team Topologies:** Organizing Business & Tech Teams for Fast Flow – Skelton & Pais, 2019
- ▶ **The Manager's Path** – Camille Fournier, 2017
- ▶ **Crucial Conversations:** Tools for Talking When Stakes are High – Patterson, Covery, & Grenny – 2002
- ▶ **The Making & Maintenance of Open Source Software** – Nadia Eghbal, 2020
- ▶ **Twenty Bits** I Learned About Design, Business, & Community – Dan Cederholm, 2020

ABOUT THIS DECK

- ▶ Created in Keynote, "Modern Type" default theme
- ▶ Memes & photos are attributed to the extent possible
- ▶ First presented Oct 2026 at PHP Longhorn in Austin, TX USA
- ▶ Updated versions available at lincolnwebs.com