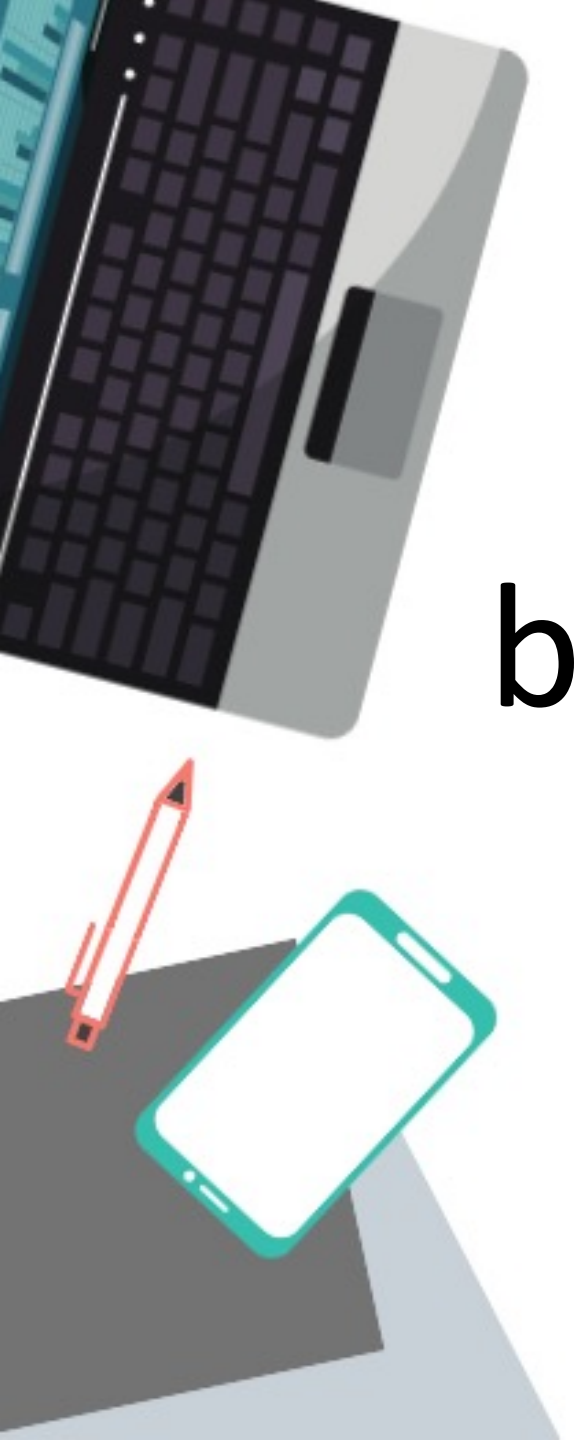


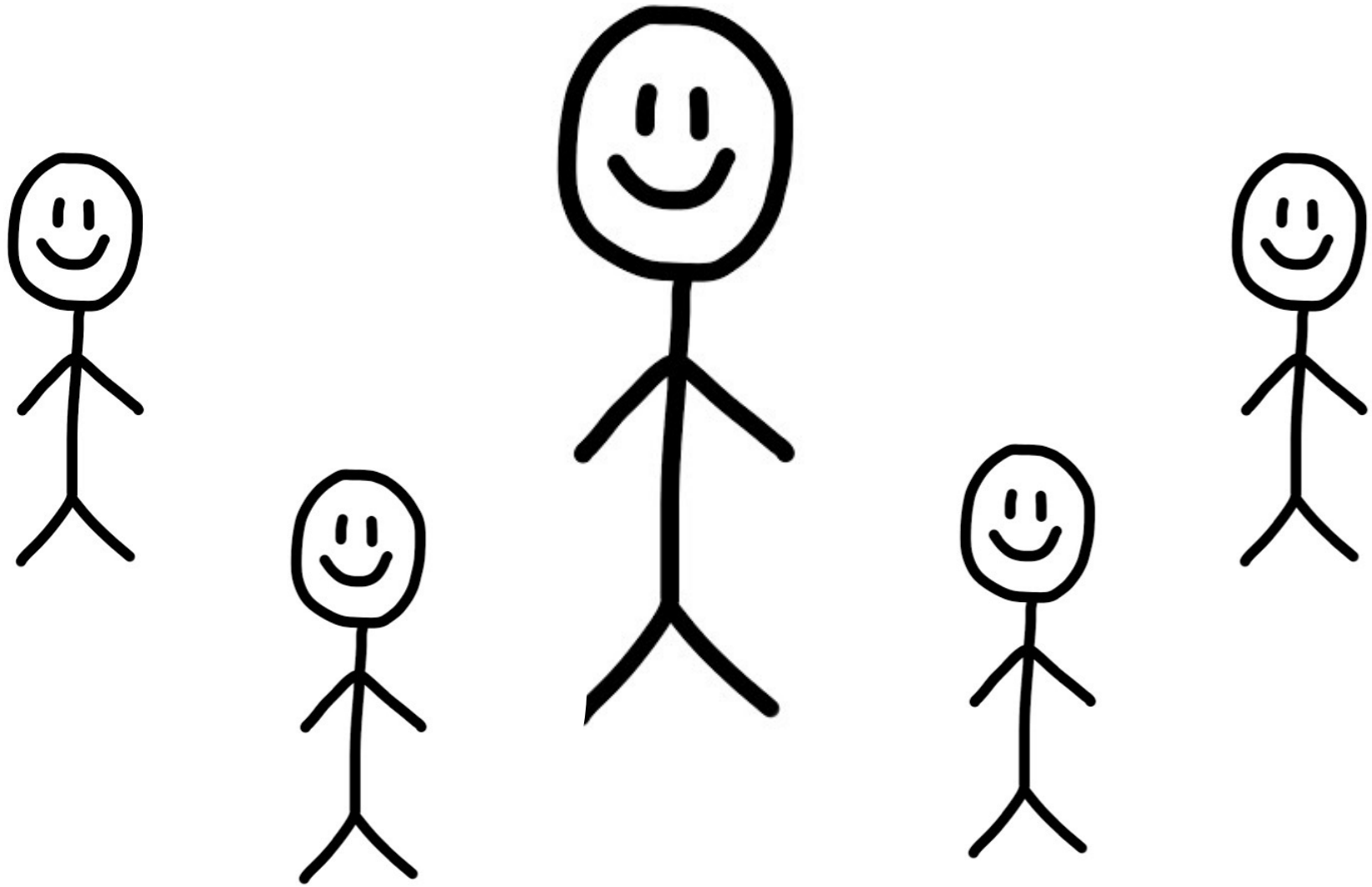
# build the engineering culture

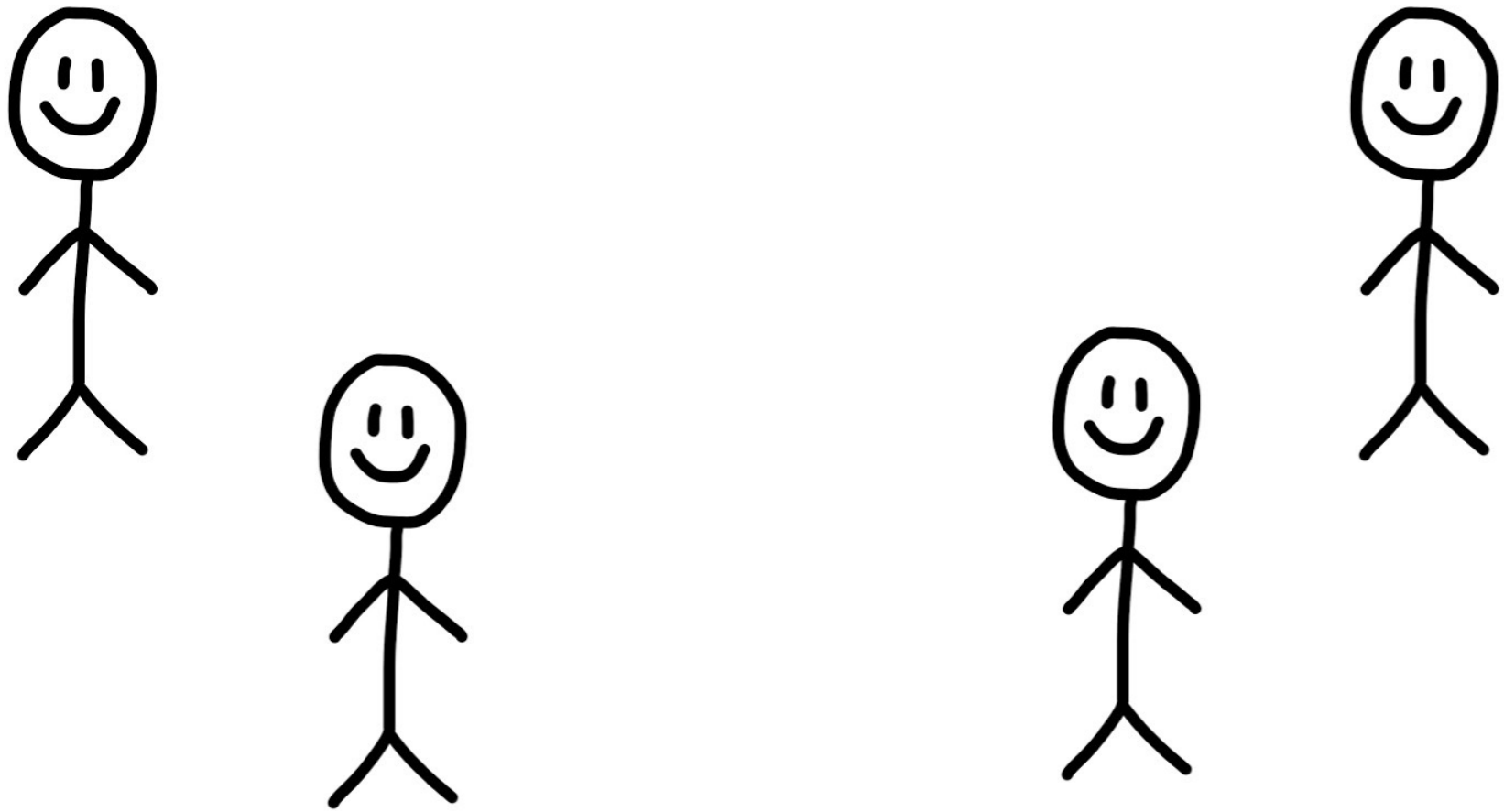
## you want to work for!

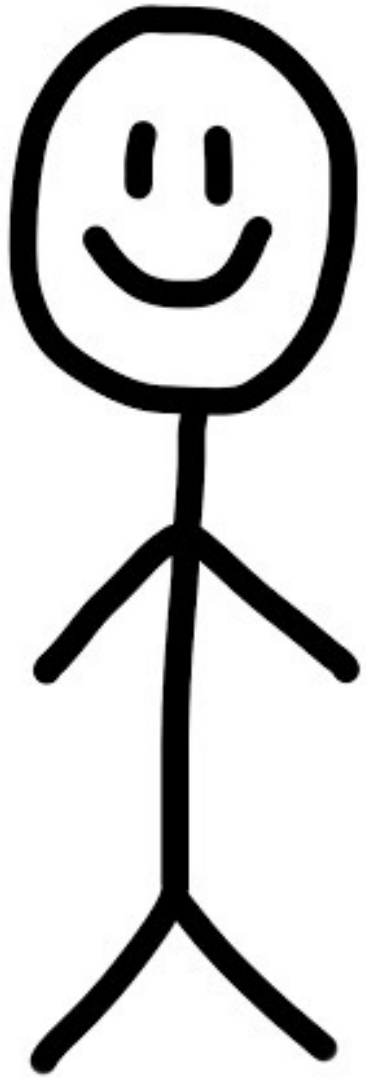
# why am I here?

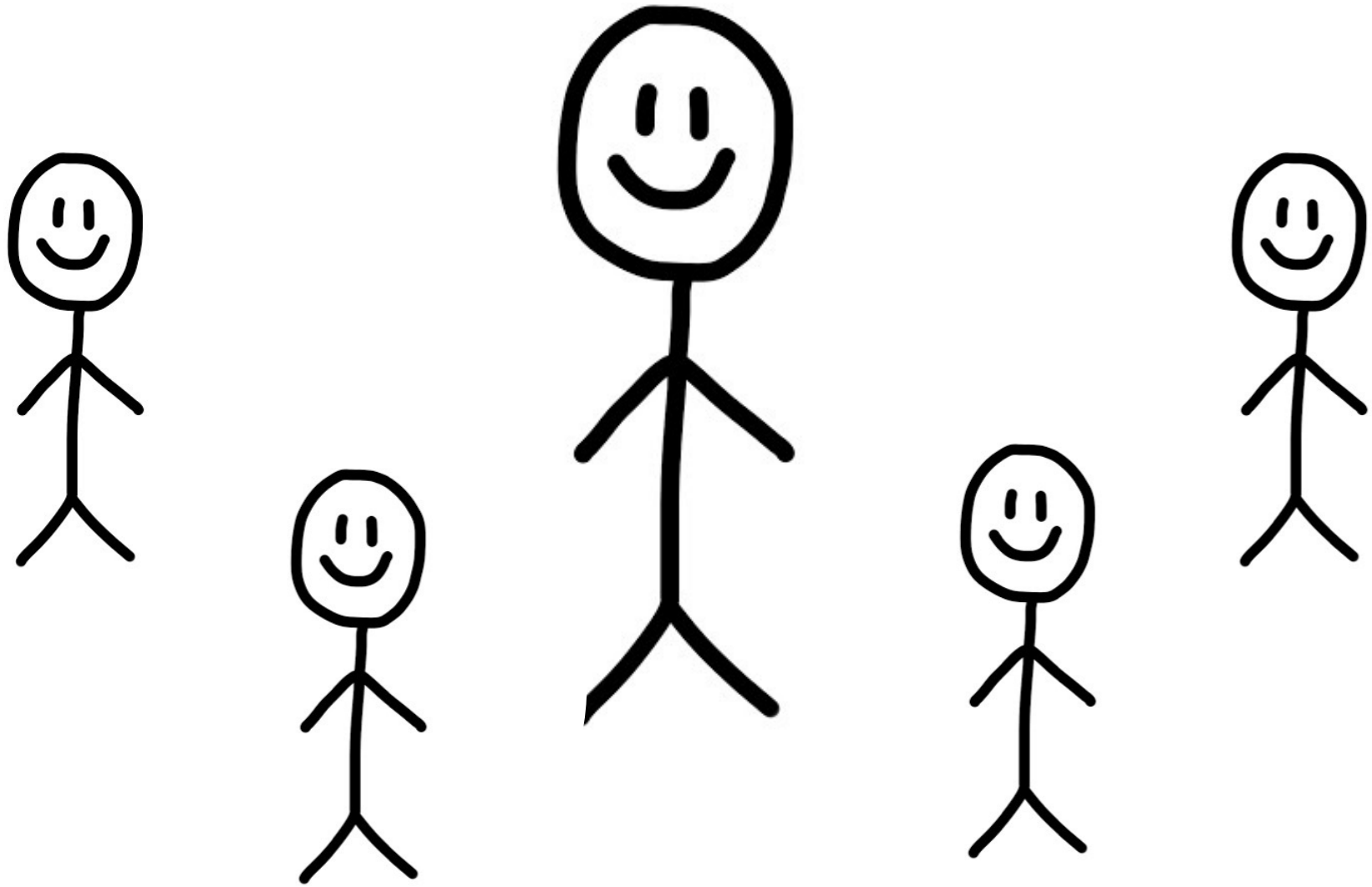

Milk Chocolate Dream
Dark Chocolate Dream
Rocky Road
Cookies & Cream
Choc Caramel
Raspberry Jam
White Chocolate Dream
Strawberry
Raspberry White Choc
Milk Choc & Custard
Cheesecake Dream
Double Choc Mousse
Coffee Cream Tiramisu
Choc Nut

what made this team's *culture* so special?









you don’t need
to be a manager
to be a *leader*

leadership should come from everywhere

# how do we do this?


to

# build an *awesome* engineering culture

identify and remove blockers that prevent the team from being productive

# onboarding

# onboarding tips

discuss team **ways of working**

provide an **overview of the team/project** and where it sits in the organization

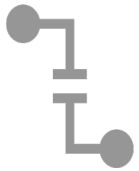
match the new hire with a **peer buddy AND mentor**

help the new **hire build a social network** – provide list of recommended people to meet with

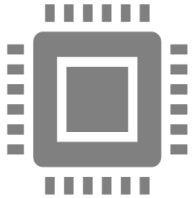
**ask new hire** what he/she prefers for **welcome celebration** (happy hour, lunch, donuts, etc)

# comprehensive dev onboarding is a MUST

access requests completed



set up **development environment** (step by step with links/screenshots)



application network architecture tech stack **diagram and visuals**

**release process** documentation (technical steps, who and how to communicate changes)

**misc developer tools** – CI/CD, testing, project management tool, chatops, source control / preferred branching/forking/etc technique, license for IDE

## find "quick wins" the new hire can complete their first week:

- bug fixes
- small enhancements

### these allow new hire to:

- build confidence
- familiarize with codebase
- understand full dev workflow

what does the day in the life of a productive Engineering team look like?

# a day in the life

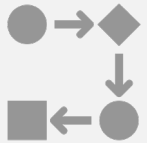
- ✓ checks team project management tool
- ✓ attends team standup and shares an update (+blockers she is facing) on her current work
- ✓ checks to confirm dev environment has been automatically updated with libraries matching dev/stage/prod
- ✓ pulls down latest code from dev environment
- ✓ makes incremental changes that are quickly validated by deploying to her local environment + running unit tests
- ✓ while working, accesses various technical documentation that is easy to find, gets help from another team member by posting question in the team's chatOps tool
- ✓ focuses on task for a few hours without interruption
- ✓ commits code change which runs through various automated checks before being deployed to production. code change includes a thorough description of what the change achieves
- ✓ releases the changes in production while monitoring health metrics in a one-stop-shop dashboard tool

# an unproductive day in the life

- ✓ wakes up to several escalation alerts for unresolved issues in production (the *assigned* on call team member didn't wake up to calls)
- ✓ checks various dashboards and digs into app logs to attempt to troubleshoot the issue
- ✓ identifies the issue
- ✓ reaches out to users to apologize for the issue while responding to several direct messages from leadership asking for updates on the fixes
- ✓ searches for documentation for another team's API that is required to fix the issue
- ✓ writes fix, waits for infrastructure team to approve emergency change request to deploy to the access-controlled server
- ✓ misses standup meeting while troubleshooting but joins several other status meetings throughout the day where she shares a recap of the issue and her progress

# the unproductive organization

everything takes longer than it should



many blockers, small inefficiencies, and process friction that add up

unhappy developers

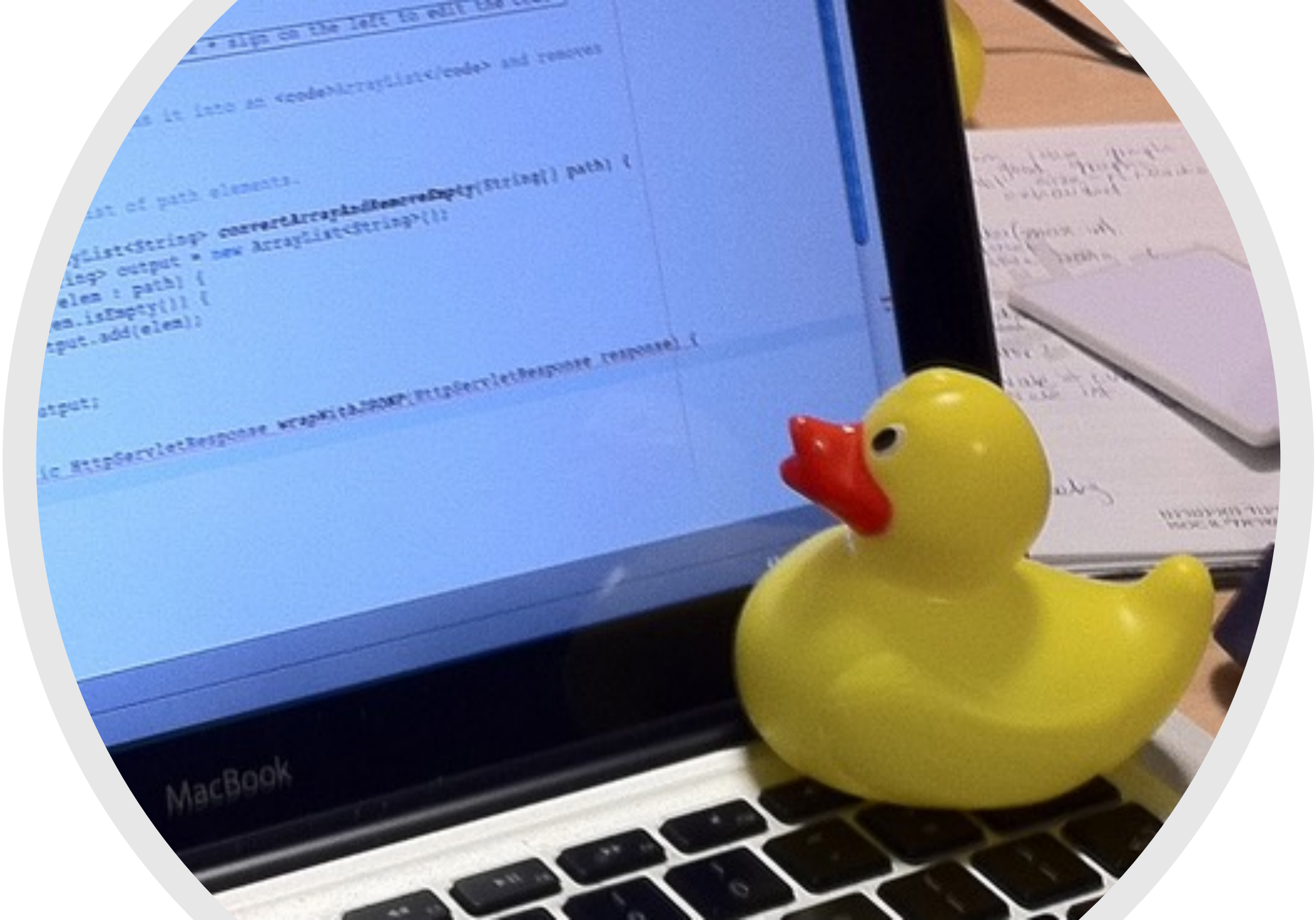
# what processes are broken?

what is harder than it should be?

what do people on the team complain about?

# reflection questions

- what are some tactics that you/your team uses to speed up development cycles?
- what are some processes that you have worked (or should work) to automate?
- what are some best practices that you've developed to facilitate better communication on your team?
- how does your team measure productivity?
- what are the time wasters that you are still working to improve?


MacBook

1. explain to the duck what your code is supposed to do at a high level

2. line-by-line, explain your code in detail

3. (ideally) at some point you will tell the duck what you are doing next and realize your problem.

be the rubber duck.

ask questions that enable people to reach a new perspective on the problem and reach a solution on their own

empower others.

## bad leaders

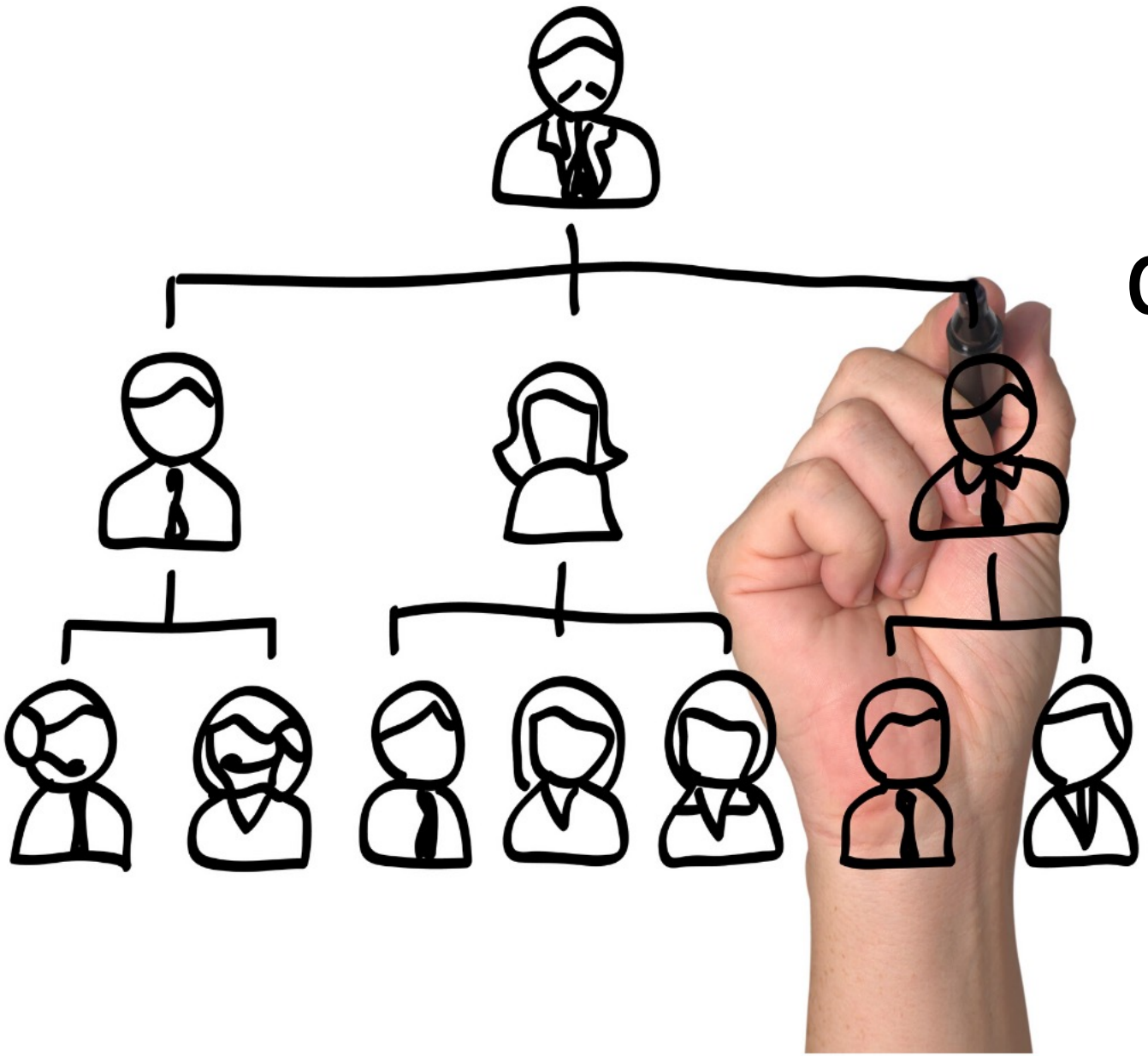
TELL PEOPLE WHAT TO DO

## good leaders

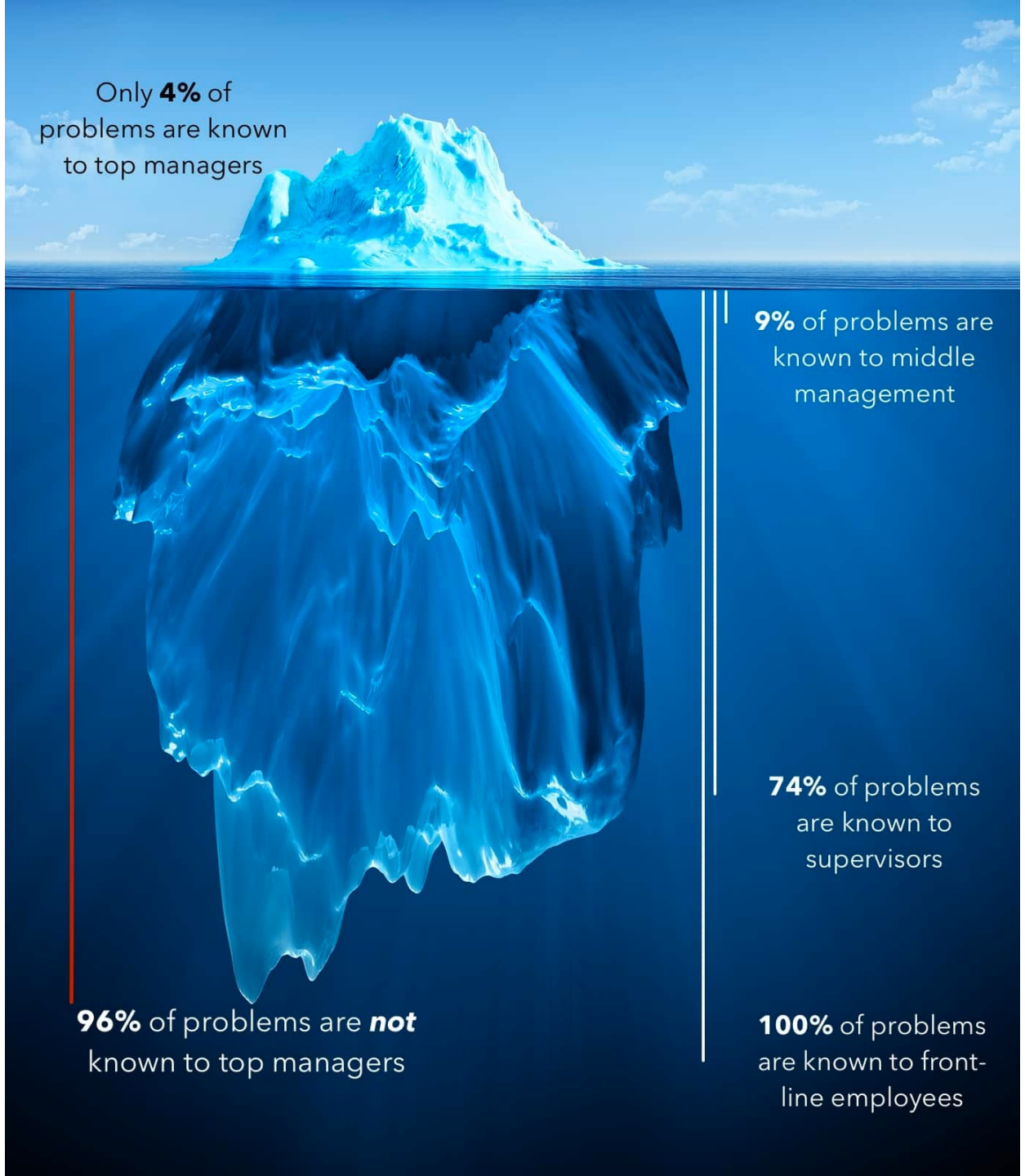
EXPLAIN WHY THEY NEED TO DO IT

## great leaders

INVOLVE PEOPLE IN DECISION MAKING



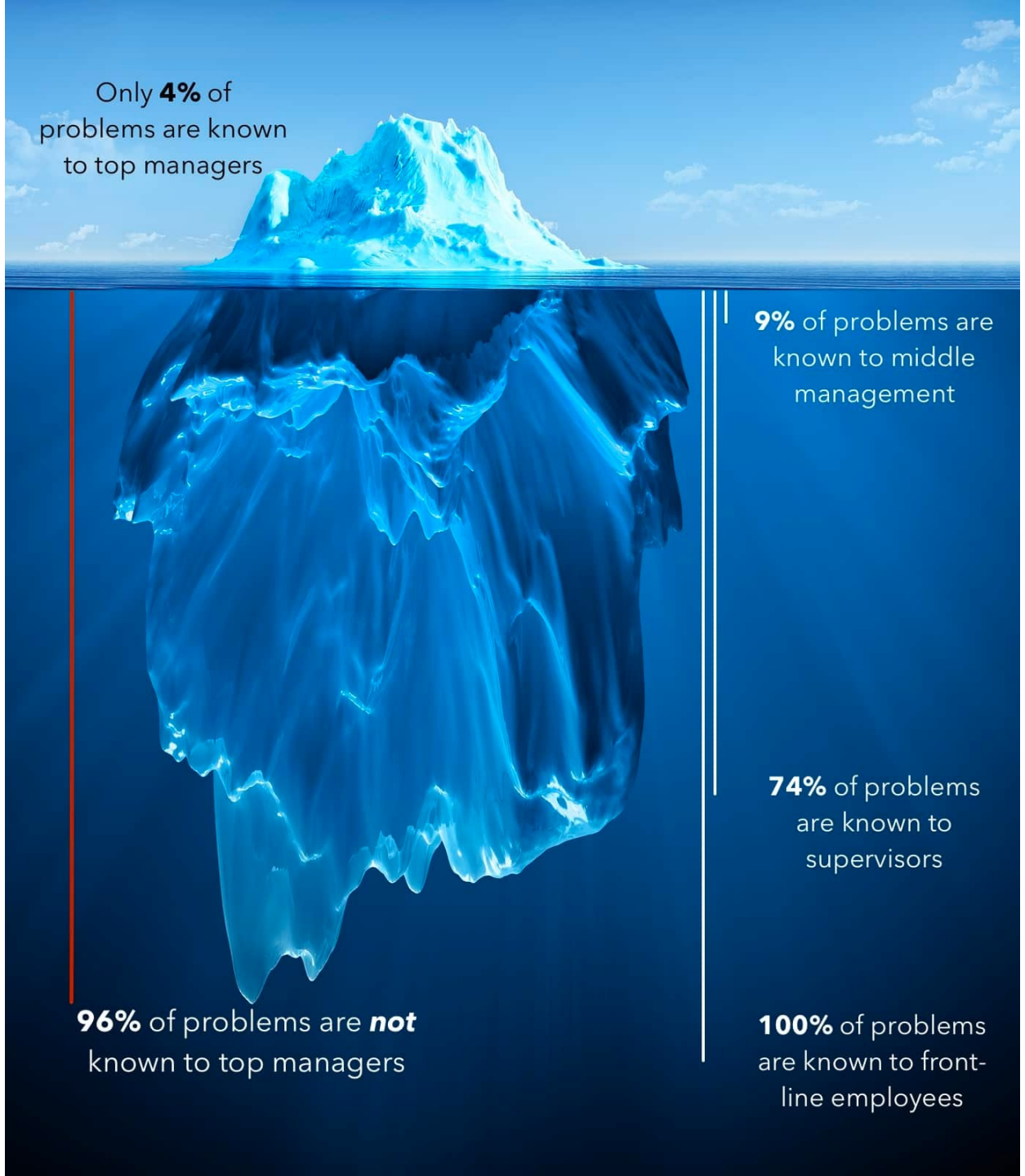
ensure that decisions are made at the level where the best information is available




**4% problems** known to **top managers**

**9% problems** known to **middle management**

**100% of problems** known to **front-line employees**




**how can we** melt the iceberg?

# be humble.

- paired programming on complex features
- collaborative code reviews
- seek to understand and expose costly pain points
- amplify your team's voice to influence organizational changes
- advocate for anonymous pulse surveys
- do stay interviews

- who should participate in making this decision?
- who will have to carry it out?
- who will be impacted by the decision?
- does everyone on the team understand the reasoning for the decision?

knowledge should be spread
across the team


wait for conversations to play out

**share credit, take blame.**

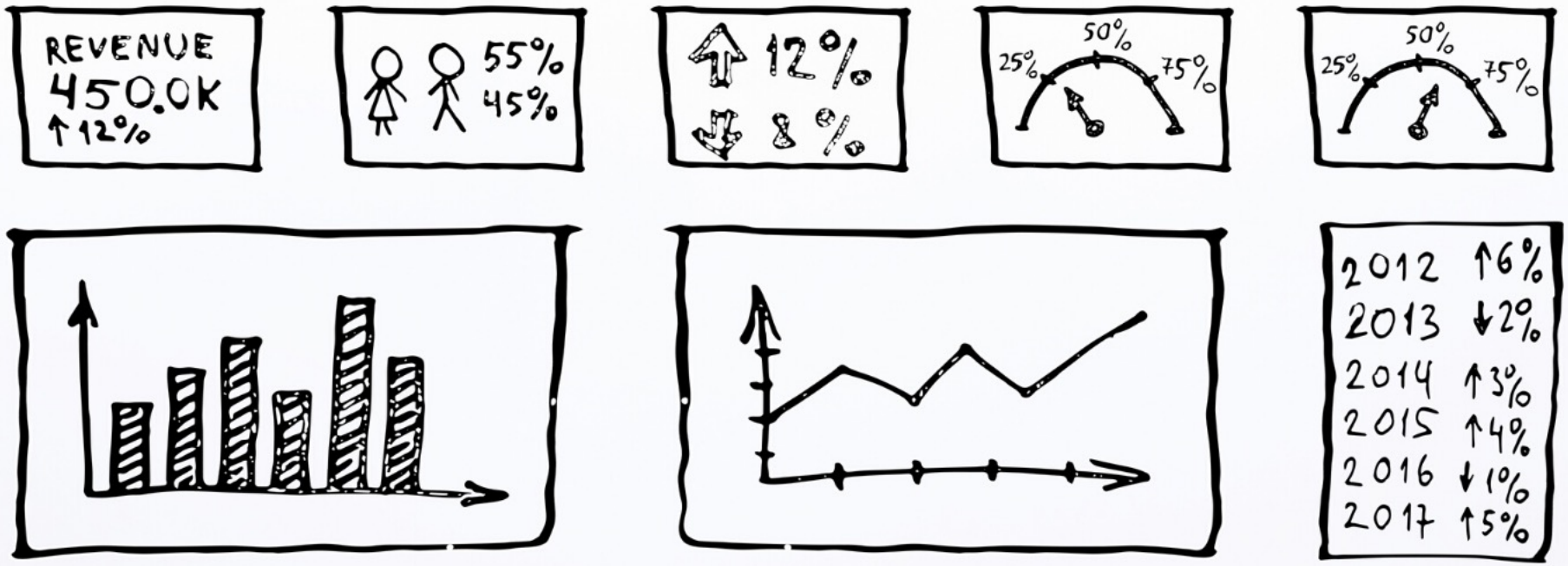
associate *names* with accomplishments


be loyal when recovering
from an ugly failure

**Brian Wang**
@brianmwang

The best measure of a leader is how much responsibility they take for their team's failures.

never devalue people
in the process of
delivering a solution.




when it comes to humans

abstraction= bad!

“when we divorce ourselves from humanity through reports and numbers, we are capable of inhumane behavior

– Simon Sinek

“tell me *how you measure me*, and I will tell you how I behave.

**- Eliyahu M. Goldratt**

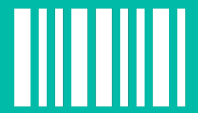
we continue making the same two mistakes

NOT MEASURING ANYTHING AT ALL

MEASURING THE WRONG THINGS

what are your *expectations* of your team members?

what are they *incentivized* by?



is it the number of commits?

test coverage?

stories completed?

establishing and tracking
valuable team metrics

# why is it important to measure things?

- helps teams estimate and plan
- allows teams to calculate “cost” which leads to better decision making
- improves processes and product outcomes
- helps to communicate individual and team expectations
- motivating & inspiring
- helps identify where change is needed

caution:

do not focus on tracking individual metrics – it should be about the team

“[if you can] reduce a job to a set of metrics, that job can be automated away. metrics are for easy problems — discrete, self-contained, well-understood problems. the more challenging and novel a problem, the less reliable these metrics will be.”

- Charity Majors

# what does a good metric look like?

- designed to answer real questions
- curated frequently
- auditable
- consistent
- visible
- diverse

# examples

# lead time

definition: amount of time from commitment to customer to working in production

to help improve: responsiveness to customers

using tools to measure: timestamps in [insert project management tool], Github hooks/APIs

# cycle time

definition: amount of time to make a change in system

to help improve: speed of technical processes

using tools to measure: CI/CD pipelines

example question: once code is ready and tested, how fast can it be available in production?

# velocity

definition: an amount of work a team can tackle in an amount of time

to help improve: estimation and planning

using tools to measure: story points

example question: can the team commit to delivering a new feature by the propose deadline?

# mean time between failures

definition: application's performance in current production environment

to help improve: performance of code

using tools to measure: monitoring and alerting platforms like ELK (elastic + kibana) and Grafana

# team morale

- I am proud of the work that I do on my team
- I am excited about the work that I do
- I find the work that I do of meaning and purpose

collect the responses

**calculate individual morale**: the scores per individual (sum the scores and divide by the number of questions)

**calculate team morale**: average the individual average's (sum the individual averages and divide by number of team members)

# additional metrics

- code test coverage (% code covered by a unit test)
- application health stats such as:
  - CPU/memory utilization
  - transactions per second
  - disk space
  - garbage collection
  - thread counts
  - response time

- how are team members mentoring one another?
- how effective are we at translating business needs into working software?
- how aligned are we to our product's vision?
- how strong are our relationships with other development teams and business partners?
- does everyone on the team have an equal voice?

resist the urge to
measure everything

resist the urge to set arbitrary metrics goals

“tell me *how you measure me*, and I will tell you how I behave.

**- Eliyahu M. Goldratt**

celebrate (and reap the benefits of) the unique characteristics of each individual

67% of job seekers consider workplace diversity an important factor when considering employment opportunities, and more than 50% of current employees want their workplace to do more to increase diversity.

organizations with above-average gender diversity and levels of employee engagement outperform companies with below-average diversity and engagement by 46% to 58%.

McKinsey & Company, **Diversity Matters** report

age
gender identity
race / ethnicity
physical abilities
sexual orientation
work experiences
mental abilities
political convictions
socioeconomic status
introvert / extrovert

geographic background
work arrangements
religion
beliefs
culture
education
language
learning style
military status
communication style

# lack of diversity on engineering teams has failed us



**The Crash Test Bias: How Male-Focused Testing Puts Female Drivers at Risk**

Researchers have known for decades that women are more likely to be killed or injured in a car crash. Why haven't safety regulators done anything about it?

**Oh dear! Amazon's facial recognition is racist and sexist – and there's a JLaw deep fake that will make you want to tear out your eyes**

building products to solve a new problem is all about creativity

“creativity is making unexpected connections between things we already know

- William A. Wulf, former president of the National Academy of Engineering

the quality of our work is affected by the degree of diversity of the team.

be vulnerable.

vulnerability: the ability to own your mistakes and not try to cover them up

articulate your own

stories of struggle

ask for

feedback

~~do you have any feedback for me?~~

if you were in my position, what would you do differently?

tell me something I do that bothers you.

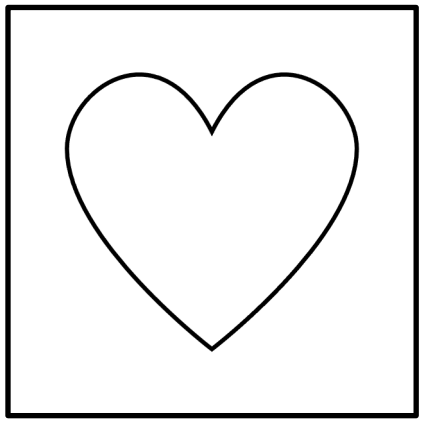
# establish trust.

6

# why is trust good overall?

- increases commitment to team goals
- communication improves
- ideas flow freely
- people are more comfortable with change and willing to work through ambiguity
- unable to inspire & influence without trust
- teams become better aligned around shared mission and common goals

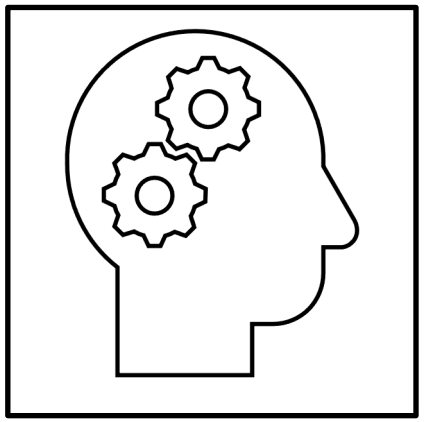
# how to build trust?



## affective trust

"trust from the heart"

sense of rapport, empathy, emotional closeness with a person based on feelings generated by interactions



## cognitive trust

"trust from the head"

confidence in the person's competence, skills, and reliability based on evidence

# affective trust examples

- trust others first
- be approachable & show empathy
- support people even when they make mistakes
- create open lines of communication
- communicate the intent behind your actions
- connect with people off-task by seeking to understand their backgrounds, interest, and aspirations

# cognitive trust examples

- acknowledge areas in which you are not an expert or when you mess up
- assume positive intent
- do what you say you will do
- be transparent, solicit feedback, and act on it

# good old fashion team building activities

- weekly reply all / icebreaker questions*
  - do you have a side project or hobby that you look forward to doing in your time off?
  - what is your ideal breakfast?
  - what is the best place you've ever traveled?
- photo of my weekend*
- monthly board games
  - Pictionary – skribbl*, drawasaurus*
- team Spotify playlist
- pets Slack channel*

*great for remote teams!

# recap

1. remove blockers
2. empower others
3. share credit, take blame
4. never devalue people in the process of delivering a solution
5. be vulnerable and authentic
6. establish and foster trust


thank you!