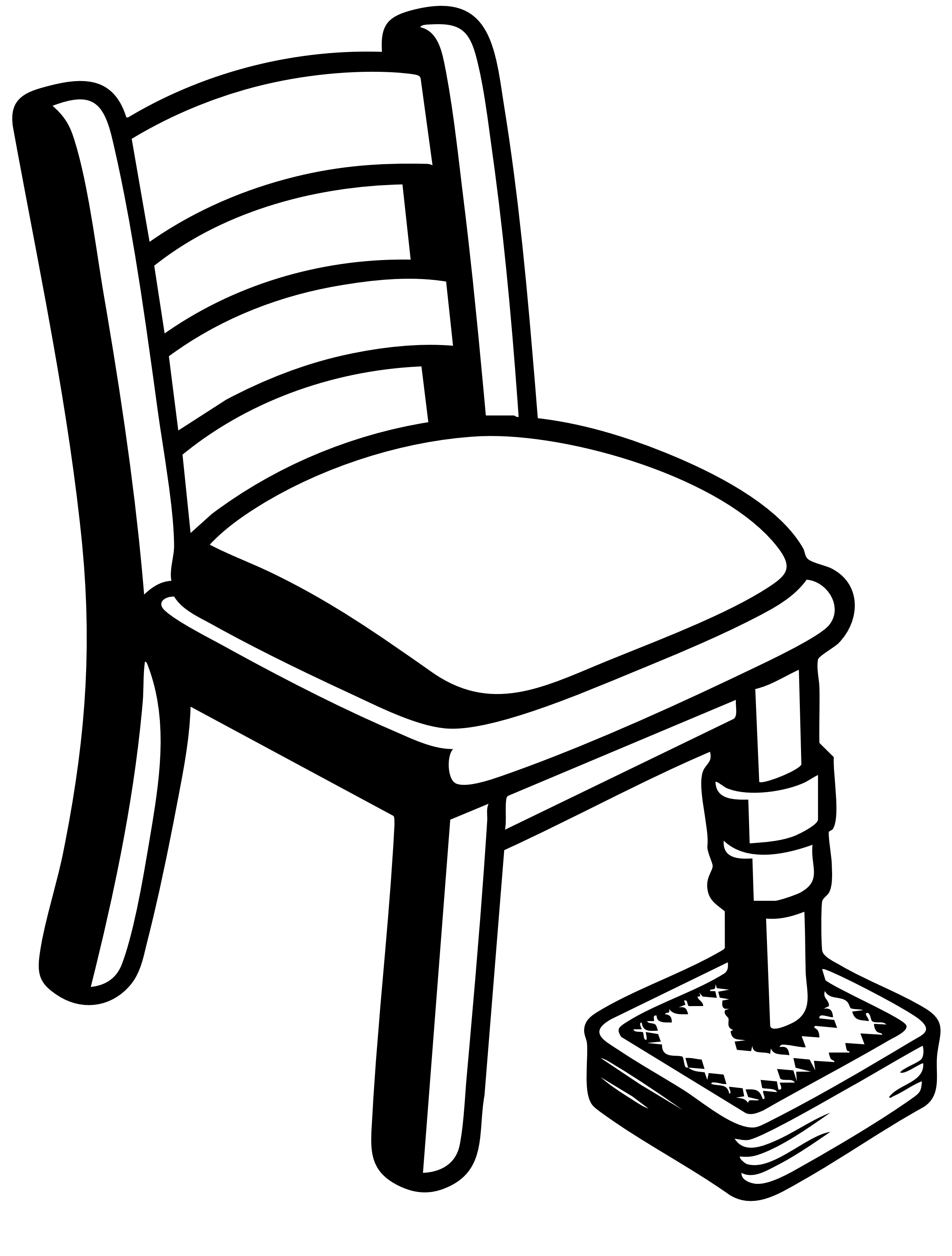
Do Dumb Things



Armin @mitsuhiko Ronacher

Who am I?

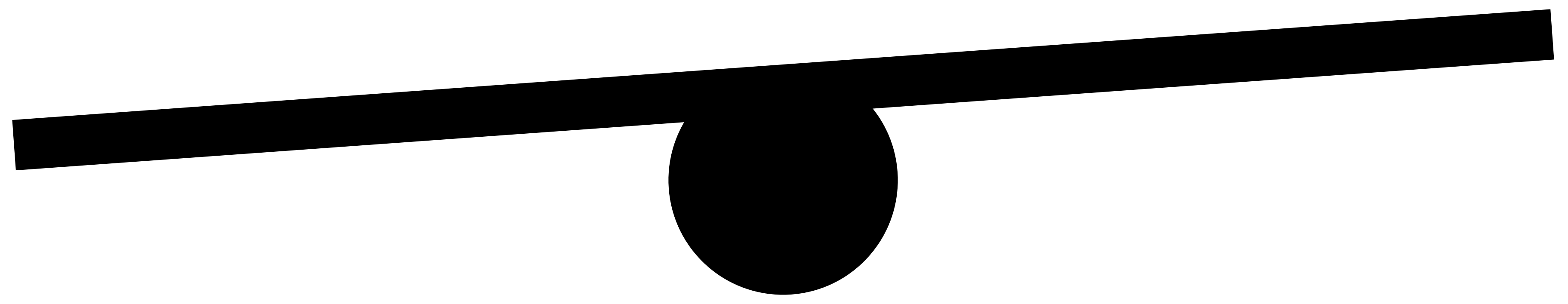
I build software — particularly Open Source software — and teams.
You might know me as creator of Flask or other Open Source projects.

- Armin @mitsuhiko Ronacher
- mitsuhiko.at
- Bluesky: bsky.app/profile/mitsuhiko.at
- Twitter: x.com/mitsuhiko
- GitHub: github.com/mitsuhiko

Two Hearts
In Conflict

- **I like building beautiful APIs and reusable libraries**
  - Build for a long time
  - Iterate carefully
  - Backwards compatible
- **I like building products that make users happy**
  - Iterate fast
  - Find product market fit

This Talk Is About The Balance



Complexity: when something is more complex than appropriate for the problem

The Dumb Things

Dumb Tools

Things you should leverage

- **Python ;-)**
- The Simplest Possible Solution That Works
- Liberal use of Copy/Paste
- Functions over Classes
- “Vibecoding” (Copilot, Cursor, ChatGPT, …)

Constraints

Constraints are Valuable

The close down paths and leave others open

- **Arbitrary Constraints**
  - Made up deadlines
  - "Must be written in Python"
- **Business Constraints**
  - Product must be cheap
  - Must support on-prem deploys

Beware of Bad Constraints

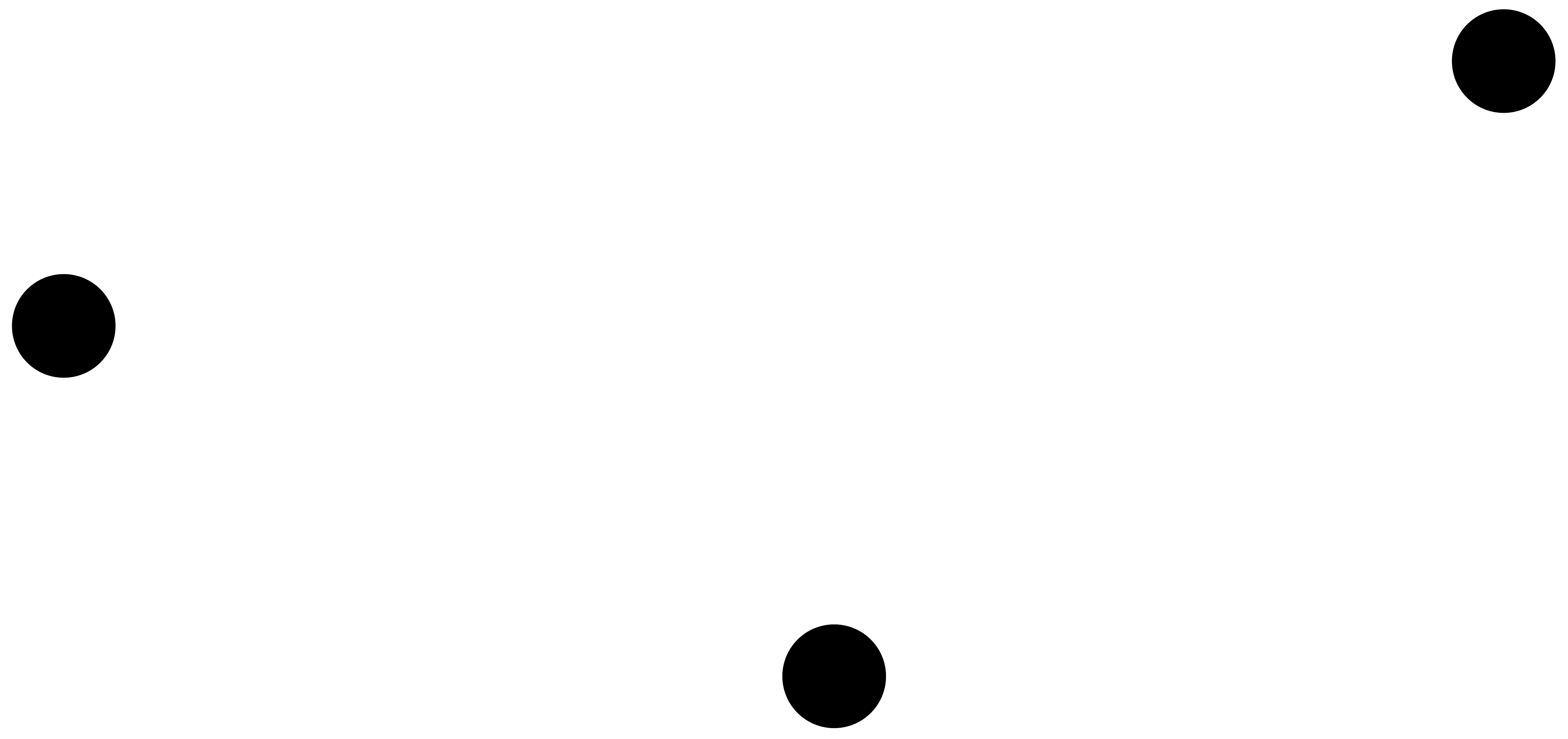
You should get something for them

- **Good constraints enable opportunities**
    - “State must fit into memory”
        - Why? Eg: Enables the use of simpler algorithms, wider choice of storage
- **Bad constraints take things that are needed**
    - “State must remain in-memory”
        - Why? *(There is no good argument in favor)* Consequence: cannot serialize, cannot use a remote in-memory database

Constraints Abstract

Emerging fix points

- Doing things a few times teaches us
- Eventually we see repetition
- We see fixed and less malleable points
- They can become the starting point of abstractions



Abstractions Kill

But they are necessary

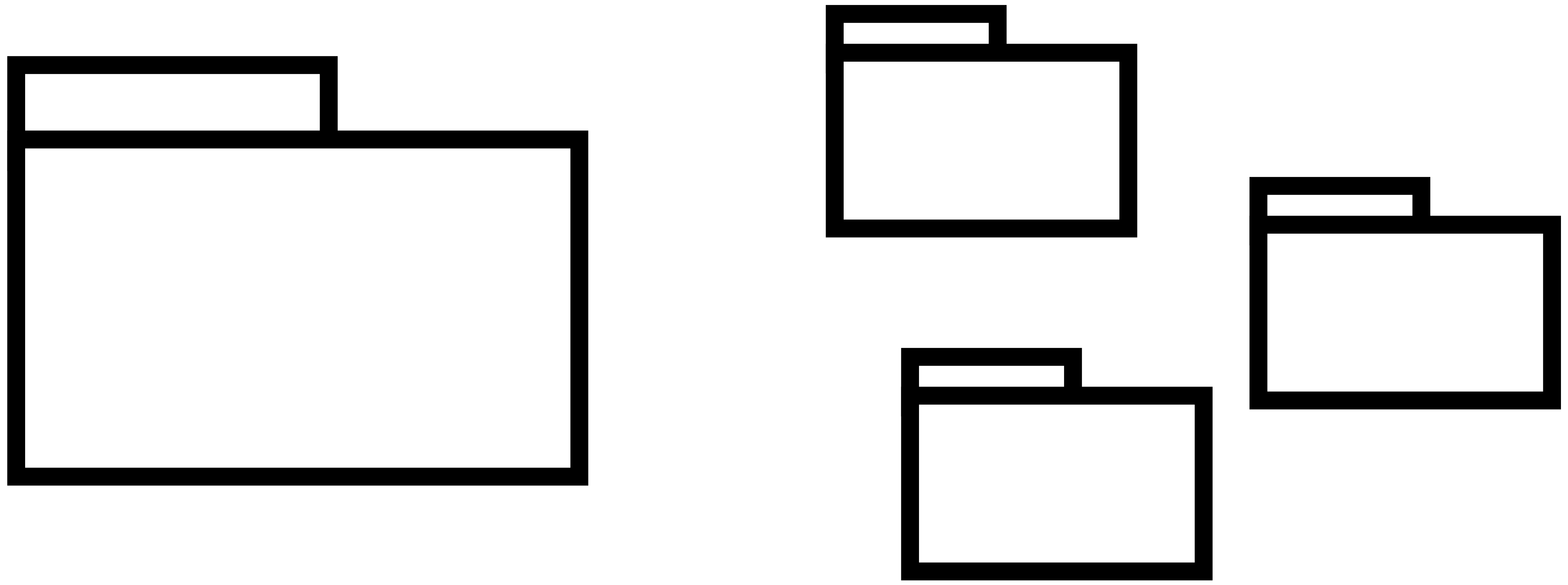
- Programmers have a tendency to over-complicate
- Most abstractions are not needed
- Only start to abstract once you have an actual need
- Refactoring is cheap if you don't have abstractions yet
- Functions beat classes for a long, long time
- Remember: **YAGNI** & **The Rule of Three**

In Closing:
Constraints are Freedom

- Where Junior Engineers see Constraints as unnecessary shackles;
- Experienced Engineers see them as a source of clarity and direction.

- Constraints narrow the problem space,

Building For Reusability



You will not reuse

Except when you do

- Most companies do not a lot of cases of re-use
- You might however have very common code paths:
  - getting a database connection / firing a database query
  - logging a debug message
  - serializing a API response
  - triggering a background task

Libraries

Are Service Boundaries

- Creating a library means having a **defined interface**
- If the interface is unstable, you create churn
- Requires lock-step movement between two code bases
- Same code base: just an internal isolation, not a library
- **Start with modularizing your code base**

Mono Repos

Cut from the Same Cloth

- Stick to a single repository for as long as you can
- Enables lock-step changes across multiple modules and components
- Simplifies tooling
- Modularize within the repository

Getting Shit Done

Chaos and Order

Why Maintenance is Important

- Codebases follow the second law of thermodynamics
  - Entropy increases in a code base over time (tends towards disorder)
  - It takes extra energy to go back to order
  - Some increases of entropy are irreversible in practice (external API contracts)
- Computers are fast, chaotic systems appear efficient
- Humans more or less stay the same, chaos results in worse iteration times

How often does it happen?

The More Common The Bigger The Impact

- How long do my tests take?
- How long does a deploy take?
- How long do I wait for a PR review?
- How long do I spend on common additions?

Engineers are Electrons

The Go The Path of Least Resistance

- Given two options they pick the one that is easier to use
- Forcing people to use "the right one" is a losing battle
- Make the right path the easy path

Cognitive Load

A Tax We Pay Everywhere

- Why does my PR review take that long?
  - Because reviewer don't show up / spend too much time
    - Because there are too many changes
    - Because it takes too long to run the PR locally
    - Because the test coverage does not give enough confidence
    - Because the changes are too hard to reason about

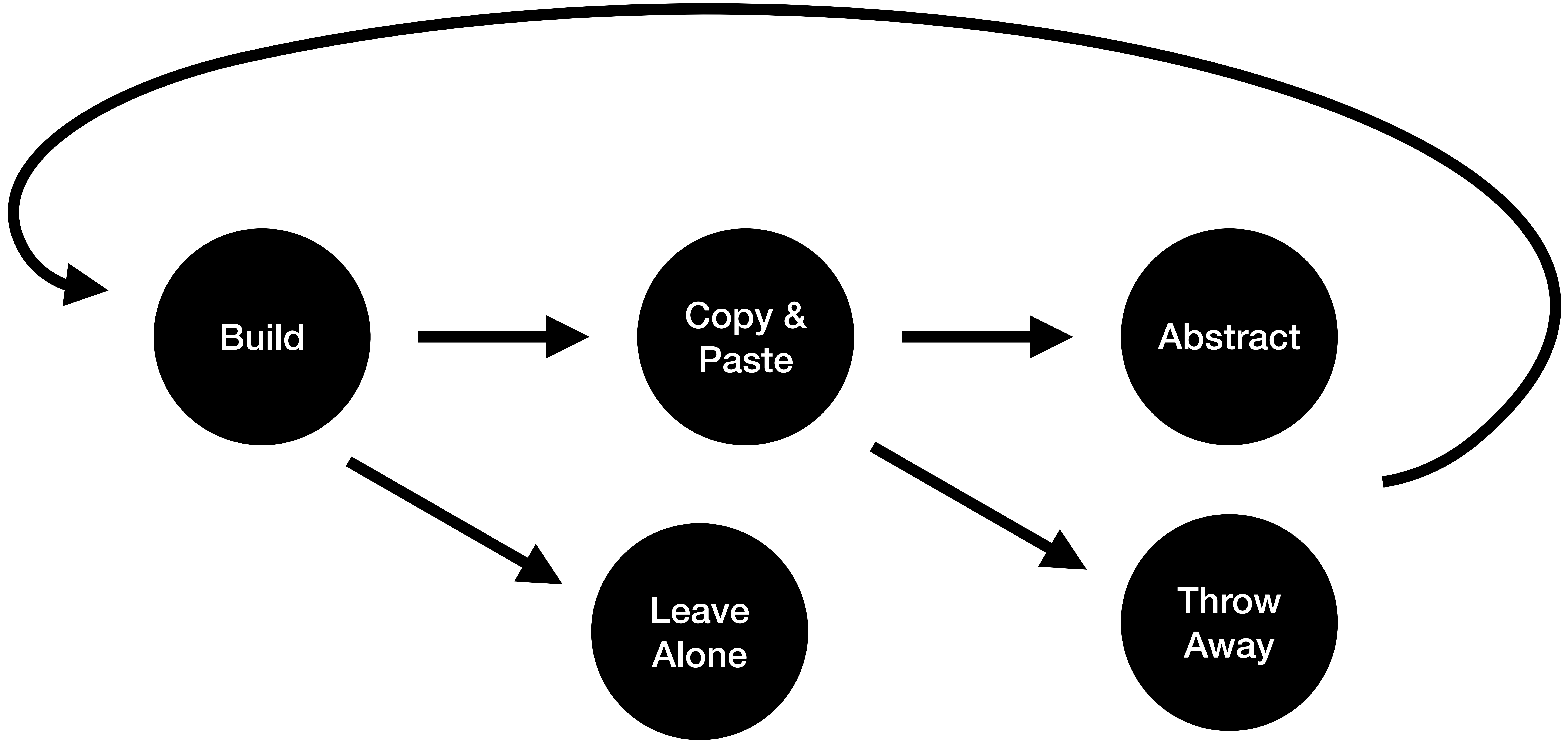
Emergent Behavior

When the machine becomes an organism

- The more complex a system, the harder it is to reason about it
- Distributed systems when not herded well, turn into complex organisms
- The earlier we lose basic understanding of behavior, the harder it becomes
- It becomes critical during incidents

The Loop

From Slop to Top



Bonus: Some Ideas

Context Managers

Instant Access to Important Data

```
from contextvars import ContextVar
from contextlib import contextmanager

APP_CONTEXT = ContextVar('APP_CONTEXT')

@contextmanager
def app_context(user=None):
    ctx = APP_CONTEXT.get({}).copy()
    if user is not None:
        ctx['user'] = user
    token = APP_CONTEXT.set(ctx)
    try:
        yield ctx
    finally:
        APP_CONTEXT.reset(token)

def get_current_user():
    return CONTEXT.get().get('user')
```

Context Managers

Example

```
def handle_request(request):
    user = get_user_from_session(request.session)
    with app_context(user=current_user):
        return dispatch_request(request)


def log(message, **data):
    data['current_user'] = get_current_user()
    logger.info(message, data)
```

Q&A