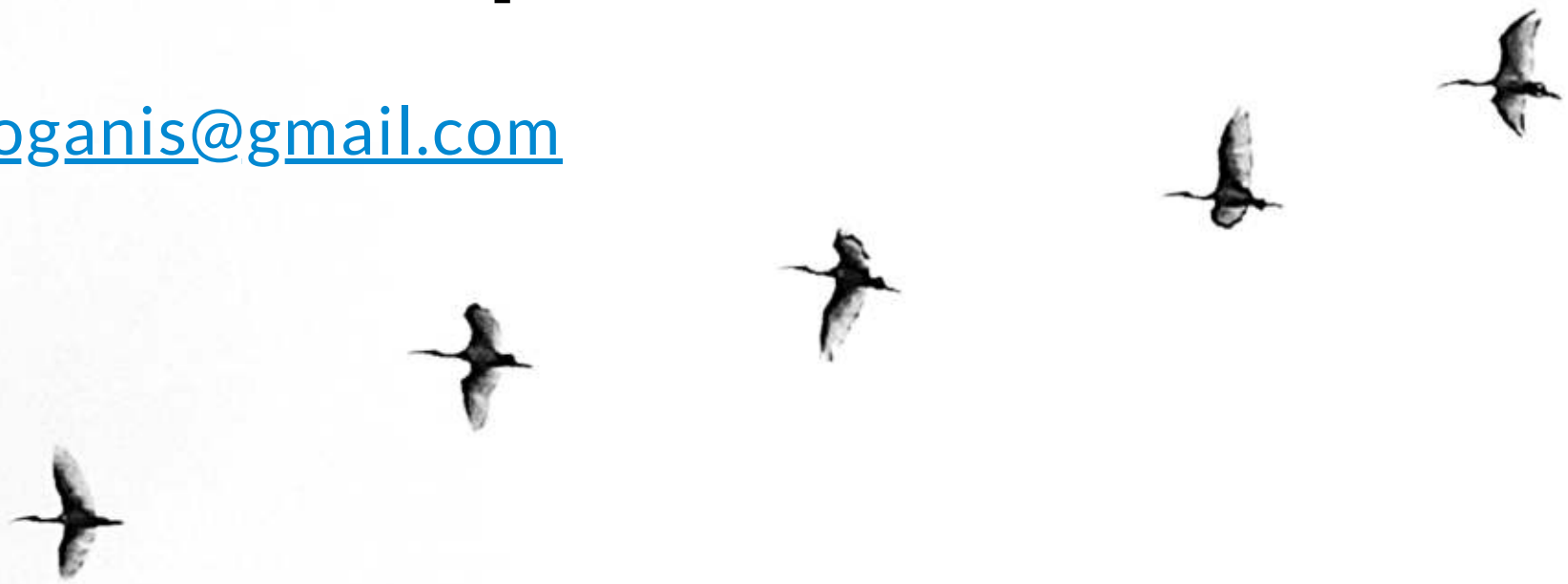
Project Development Guide

fivos.doganis@gmail.com



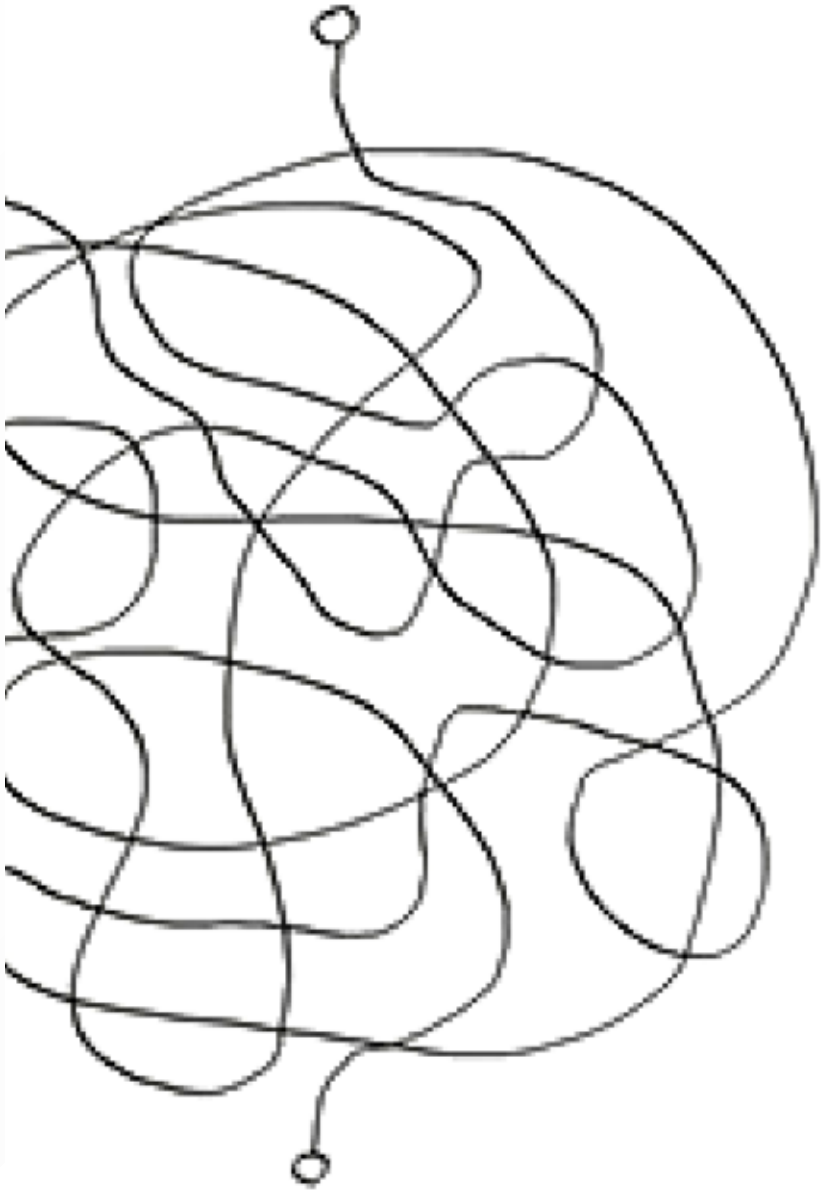
KISS

K eep

I t

S imple

S tupid



!= Easy

"~~Simplify~~, ~~Simplify~~, Simplify"

Steve Jobs

YAGNI

Y ou

A in't

G onna

N eed

I t

Code for **now**,
not for the future.
Refactor often.

ROADS?
WHERE WE'RE GOING,
WE DON'T NEED ROADS.

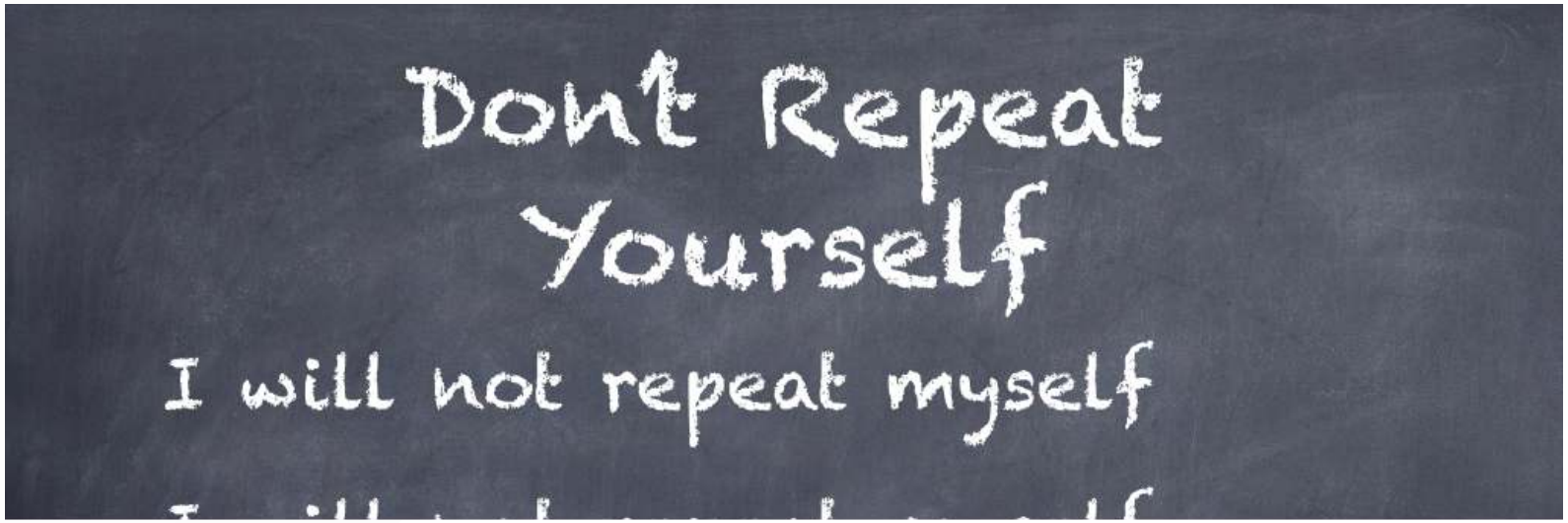
DR. EMMETT BROWN - BACK TO THE FUTURE

DRY

D on't

R epeat

Y ourself



!= WET :

~~**W**rite~~
~~**E**verything~~
~~**T**wice~~

No Copy Paste!

Copy Paste == Code Debt

Copy Paste == Refactoring Opportunity

~~Copy Paste~~

Stop!

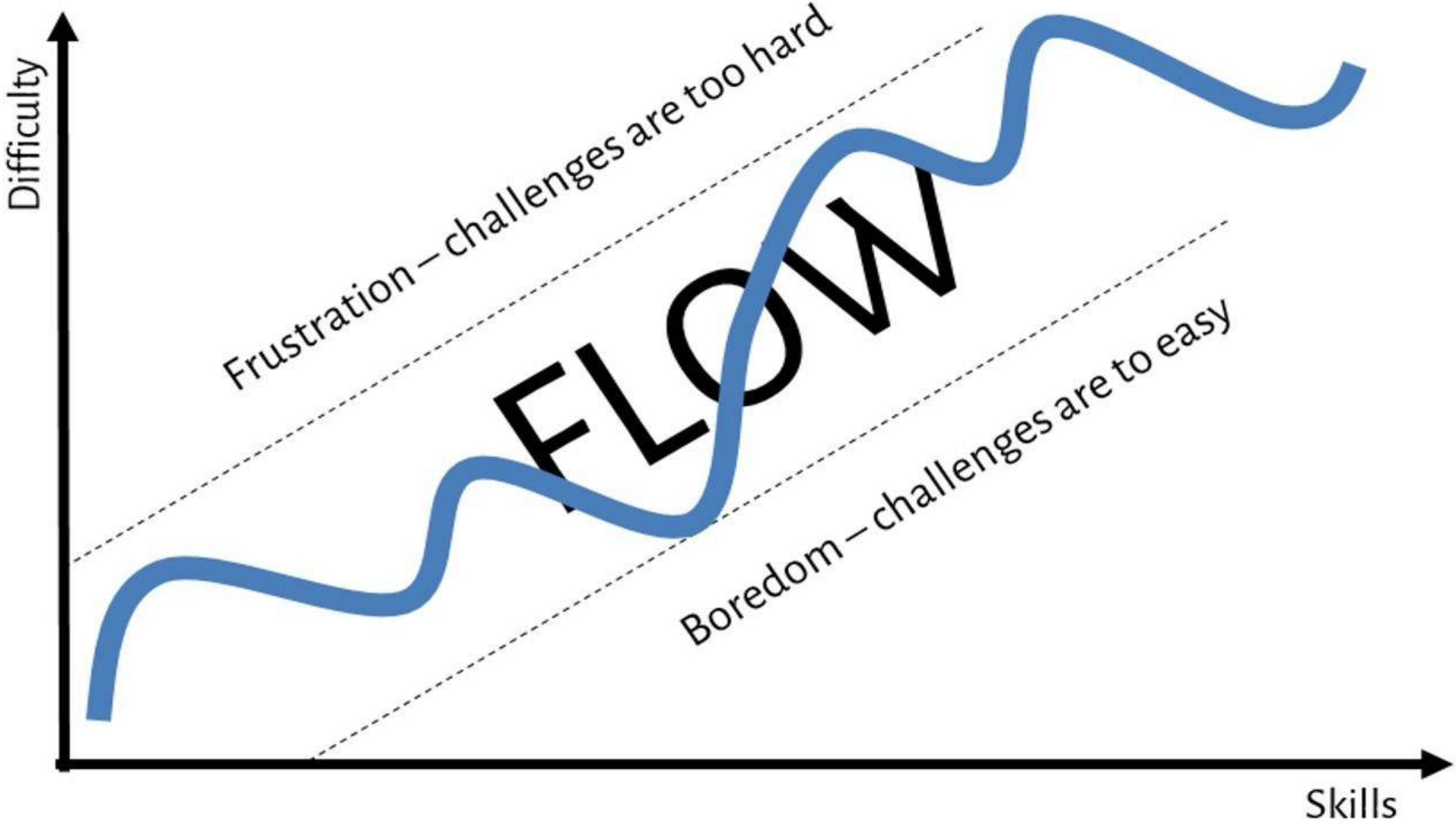
Think

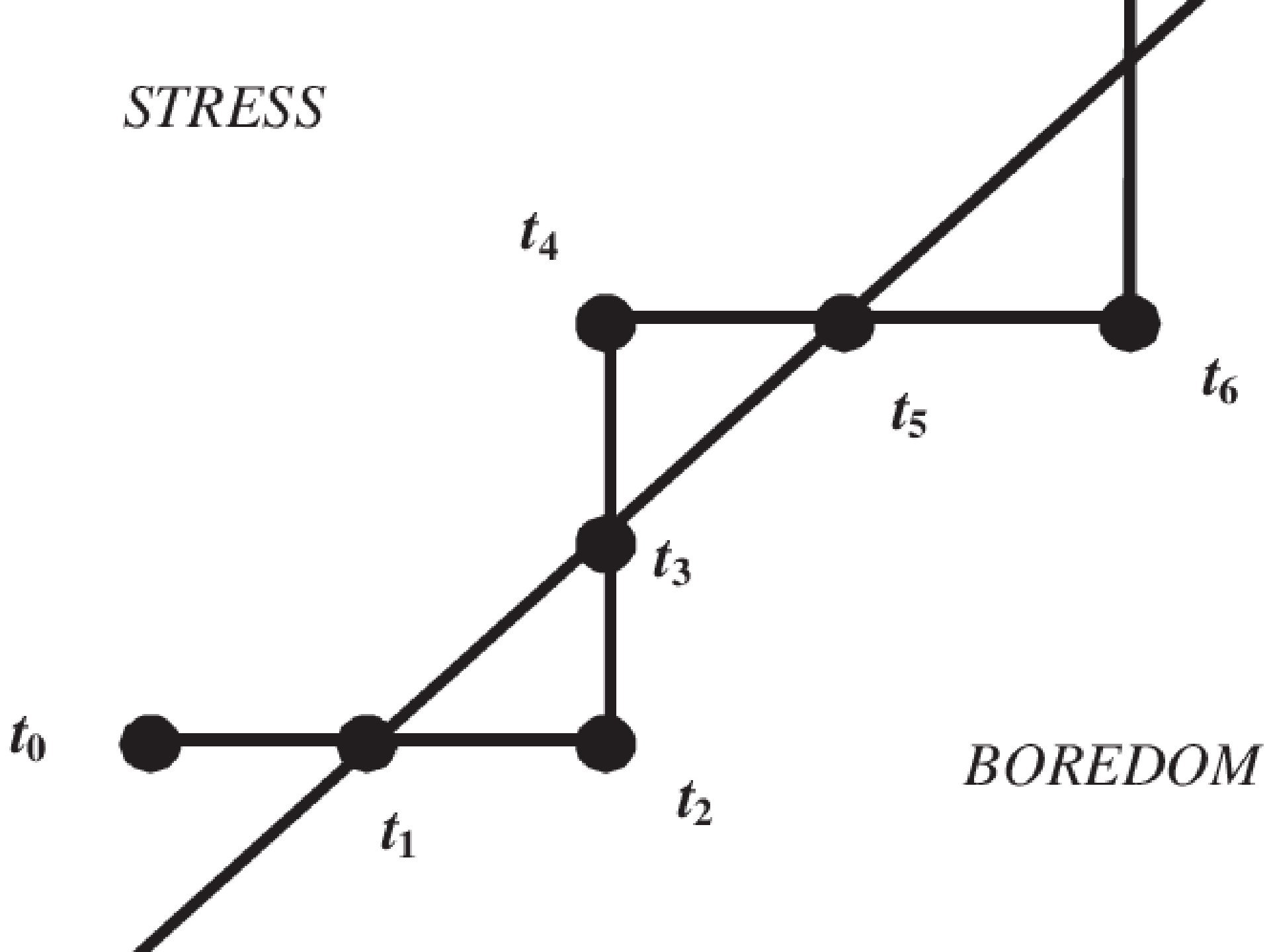
Refactor Now!

Flow Theory

The art of staying in the zone

by Mihály Csíkszentmihályi


Difficulty
Frustration – challenges are too hard
FLOW
Boredom – challenges are to easy
Skills


STRESS
t_4
t_5
t_6
t_3
t_0
t_1
t_2
BOREDOM

Stuck?

Fight procrastination

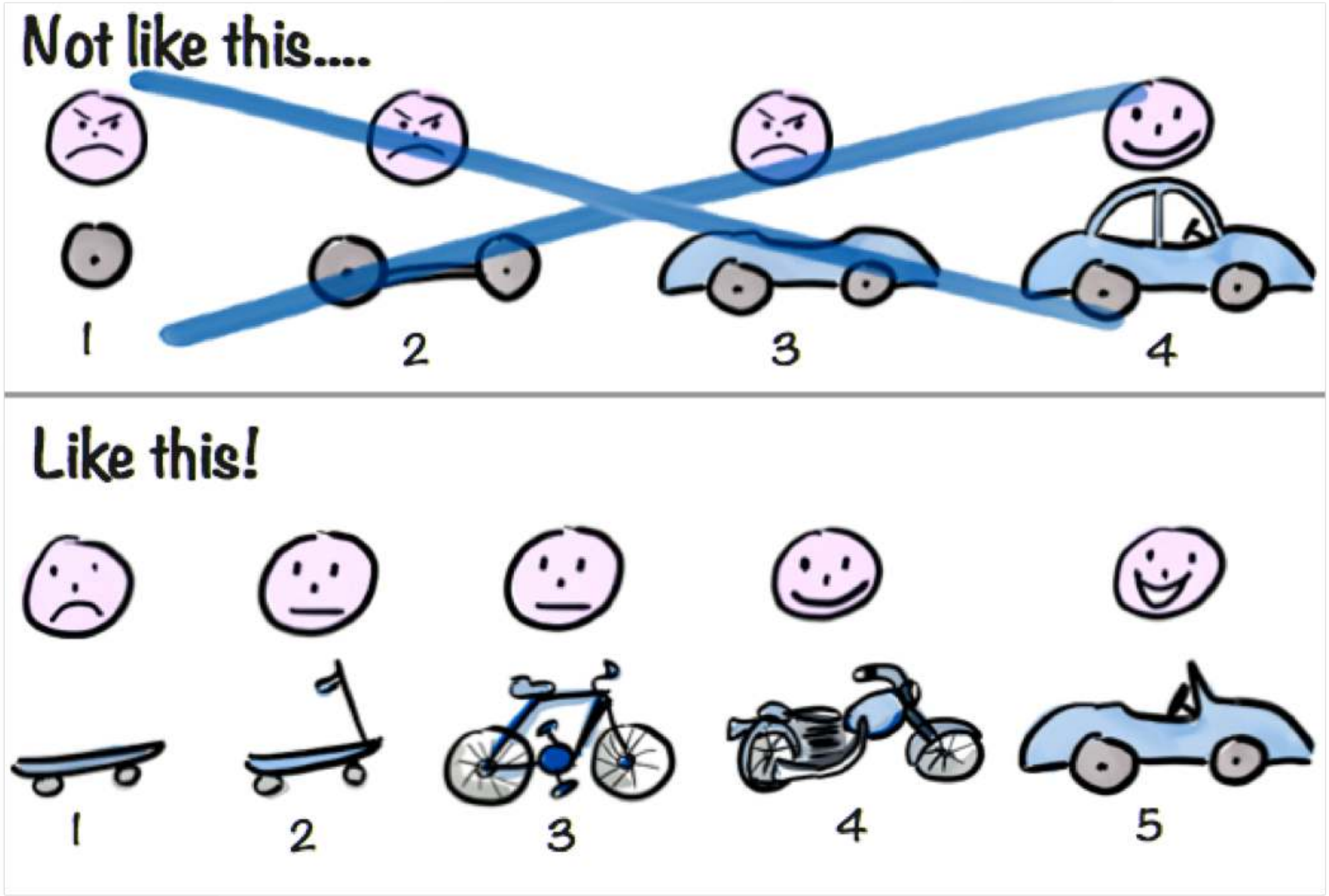
TODO ;)

Fighting procrastination

- **OK to fail**
  - ~~"Die and retry"~~ ➡️ Code and refactor
- Decompose into **micro actions**
  - Github issues + tags ("easy", "good first issue", etc.)
- Start just for **5 minutes**
  - identify easy issue
  - fix it to get the ball rolling
- Adjust skills and difficulty to stay in the zone

Iterative Design

MVP: Minimum Viable Product


Not like this....
1
2
3
4
Like this!
1
2
3
4
5

Design Thinking

A Non-Linear Process

Design Thinking

Empathize

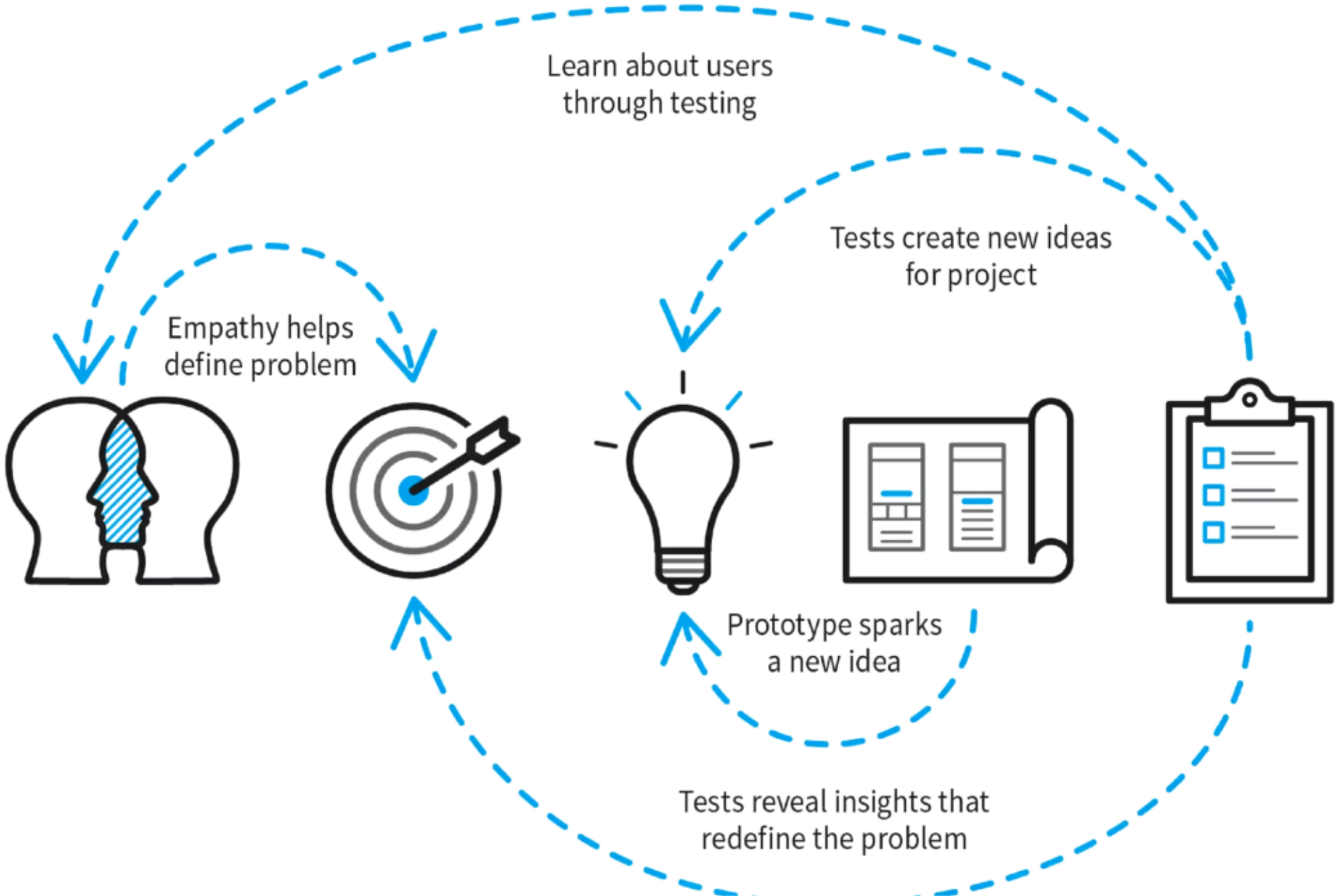
Define

Ideate

Prototype

Test

Interaction Design Foundation
interaction-design.org


Learn about users
through testing
Empathy helps
define problem
Tests create new ideas
for project
Prototype sparks
a new idea
Tests reveal insights that
redefine the problem

UX Basics

What makes a good User Experience?

Use objective criteria

SUBJECTIVE	OBJECTIVE
~~INTUITIVE~~	EFFICIENT: FAST, ERROR PROOF
~~NATURAL~~	SIMPLE
~~REALISTIC~~	PRECISE

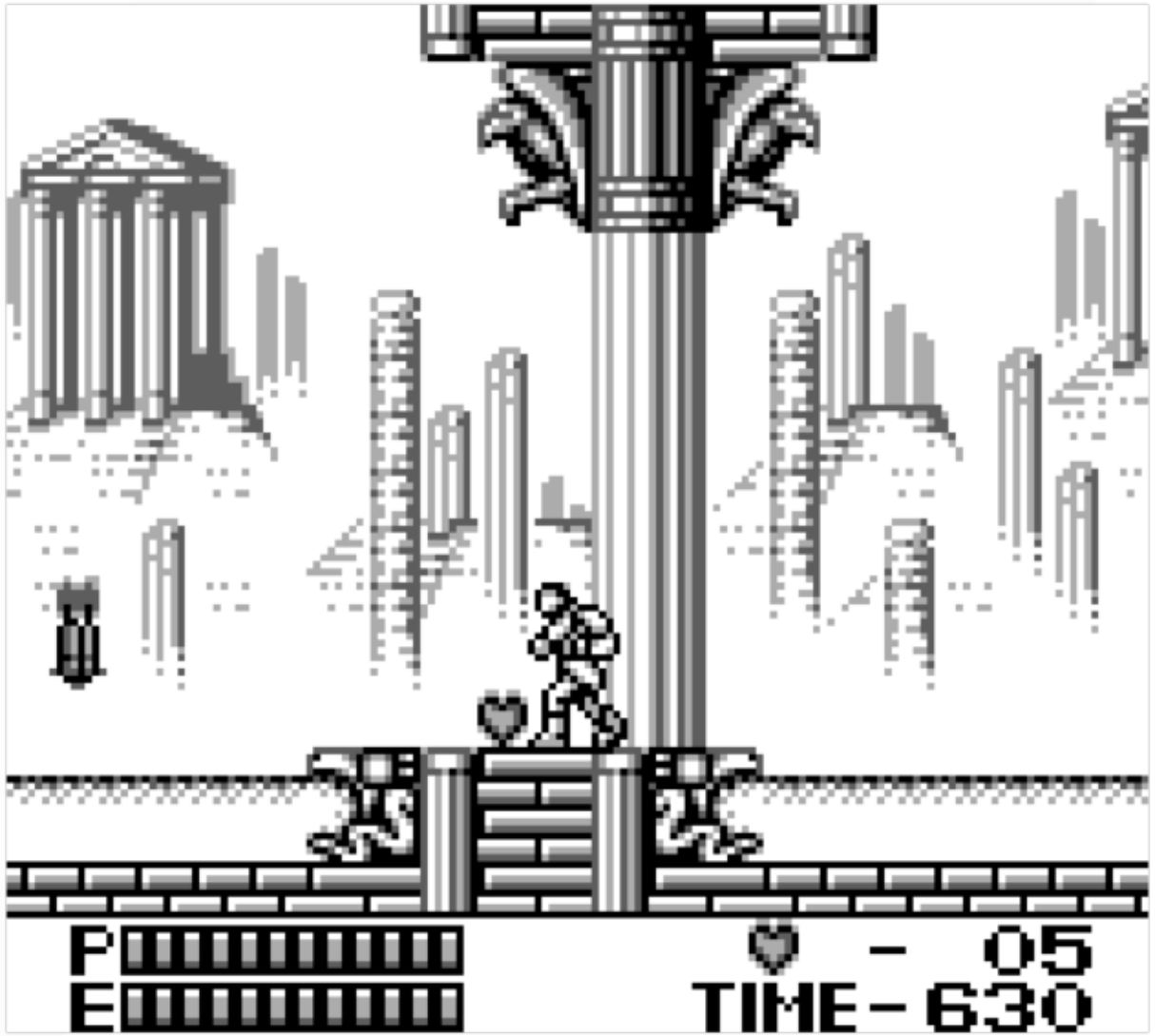
Affordance

Form ➡️ function

- button 🎹
- door knob 🚪
- hammer 🔨

Discoverability

- user guesses right

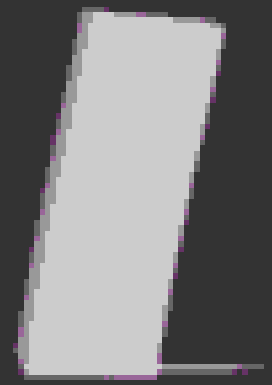


Castlevania ❤ item : **BAD** affordance!

Juice

More than eye-candy

- action continuity
- smoother experience (TWEEN / lerp)
  - Robert Penner's equations
- user satisfaction 🤩
  - reward mechanism
- ⚠️ avoid bad juice
  - distracting, unexpected



The laws of UX

Fitts's law

Fitts' Law

In plain words

Big and near objects are easy to click.

Small and far objects are hard to click.

Interaction Design Foundation
interaction-design.org

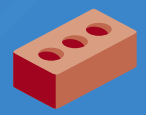
Find measurable KPIs*

Examples

- **How long** does the user take to reach the target?
- **How many** mistakes are made before succeeding at a task?
- **How accurate** is the result vs a reference?

KPI : **K**ey **P**erformance **I**ndicator

Architecture Basics



Design Patterns

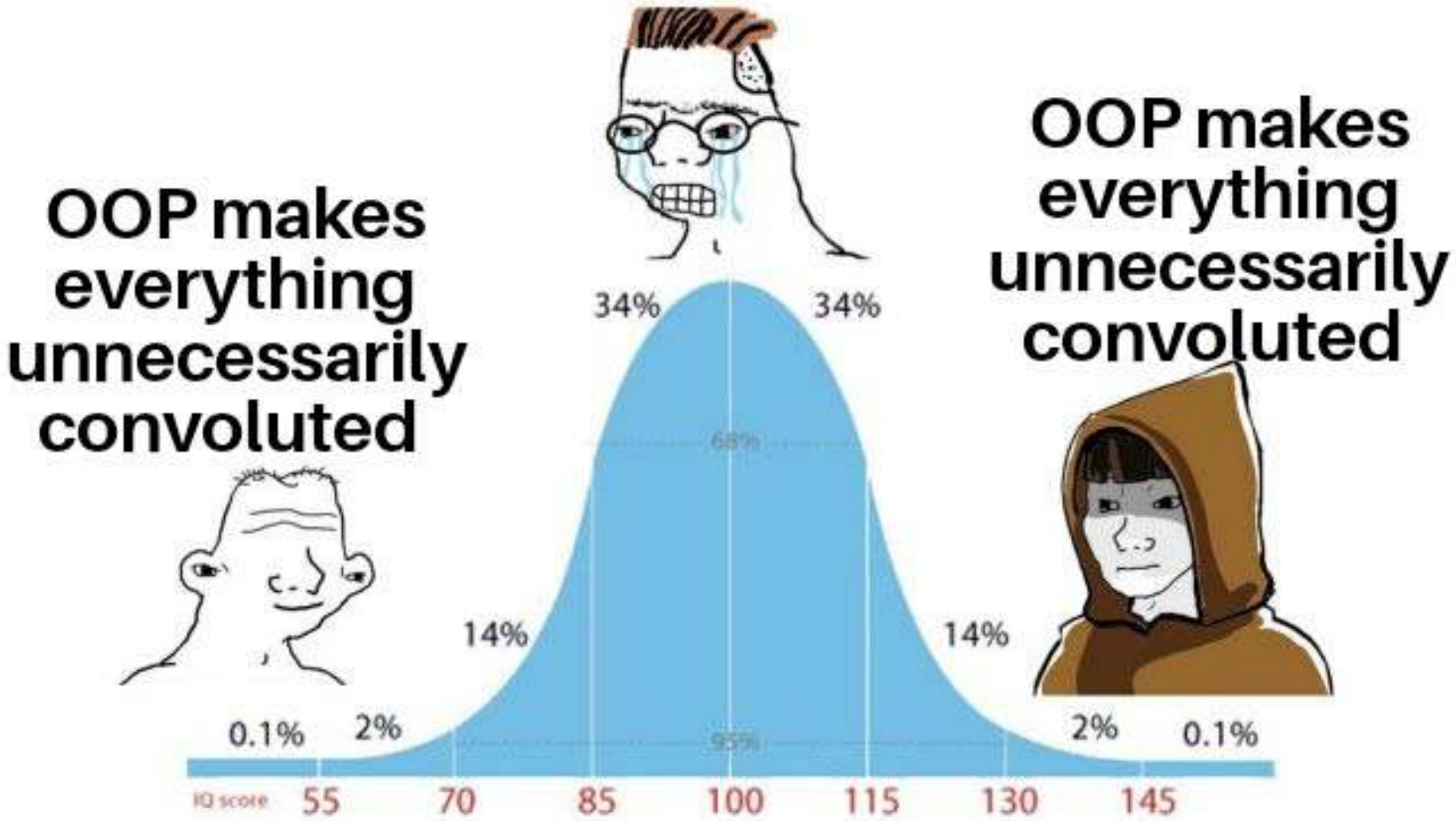
- Builder, Observer, State can be interesting
- ⚠️ avoid Singletons!
- **adapt** the pattern to your code, don't use as-is!

Game Programming Patterns 👉

- Game Loop
- Object Pool

AntiPatterns

Noooo, you need to use design patterns to make the code better. OOP helps us. Just listen to Uncle Bob



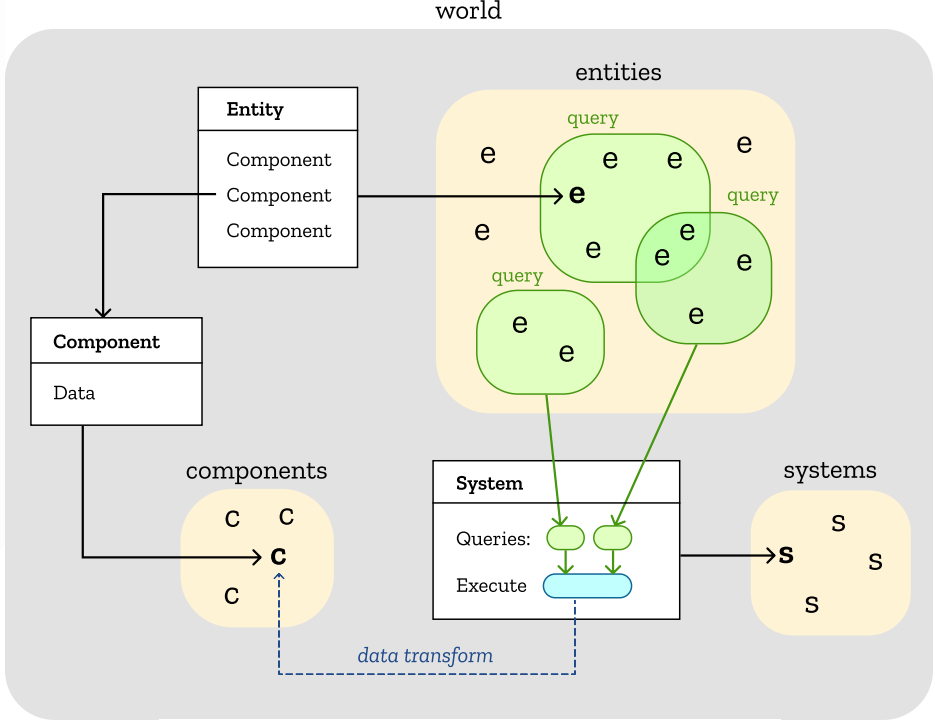
https://refactoring.guru/design-patterns/criticism

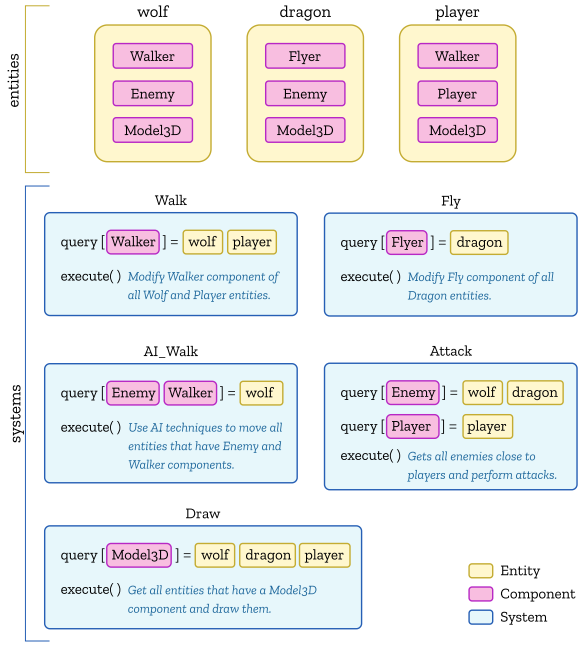
ECS

Entity **C**omponent **S**ystem

Articles

- Mozilla
- Medium
  - ECS in 99 lines of code
- ECS @ Apple (Video 25') ⭐
- used in Overwatch





Final tips

To keep your mind and code sane

Self documenting code

- document the "why?"
  - at high-level
  - if needed
- avoid documentation altogether
  - write tests, examples, tutorials instead
  - write **code as documentation**

Code as documentation

- **BAD** ❌

```
const float a = 9.81; //gravitational force
float b = 5; //time in seconds
float c = (1/2)*a*(b^2) //multiply the time and gravity together to get displacement.
```

- **GOOD** ✅

```
float computeDisplacement(float time_s) {
   const float g = 9.81;
   float displacement = (1 / 2) * g * (time_s ^ 2);
   return displacement;
}
```

← Tweet

Leon Bambrick
@secretGeek

There are 2 hard problems in computer science: cache invalidation, naming things, and off-by-1 errors.

7:50 PM · Jan 1, 2010

https://martinfowler.com/bliki/TwoHardThings.html

Naming Conventions

- "What is this object? What does it do?"
- Object, variable ➡️ noun
  - `user`, `accountNumber`, `customerEmail`
- Function, method ➡️ verb
  - `user.login()`, `shutDown()`
- Boolean ➡️ adjective
  - `allowed`, `disabled`
  - `user.active()` ❌
  - `user.isActive()` ✅

Naming tips

- convey **intention**

  `var d // elapsed time in days` ❌ vs `var elapsedTime_days` ✅

- name **arguments** too!

  `void copyChars(char a1[], char a2[])` ❌

  `void copyChars(char source[], char destination[])` ✅

- **pronunciation** (avoid abbreviations)

  `genymdms` ❌ vs `generationTimestamp` ✅

- **no magic numbers** (keep your code searchable and maintainable)

  `s / 5` ❌ vs `task / WORK_DAYS_PER_WEEK` ✅

- use **camelCase**

> Any fool can write code that a computer can understand.
> Good programmers **write code that humans can understand**.

Martin Fowler

The End!

cover photo by Megha Ajith on Unsplash