The World **needs** *Full-stack* Craftsmen

codeborne
well-crafted software

Anton Keks
@antonkeks

Tallinn, Estonia

//codeborne

well-crafted software

Founded in 2010

33 craftsmen, almost no other roles

Most biggest Estonian companies are/were our customers

Foreign customers as well, including Russia, Japan, USA, Norway, …

Incredible efficiency thanks to craftsmanship

I am going to share our secrets, don’t tell anyone!

A reminder:
IT exists only because we support businesses, governments, etc

Despite that we constantly fight each other

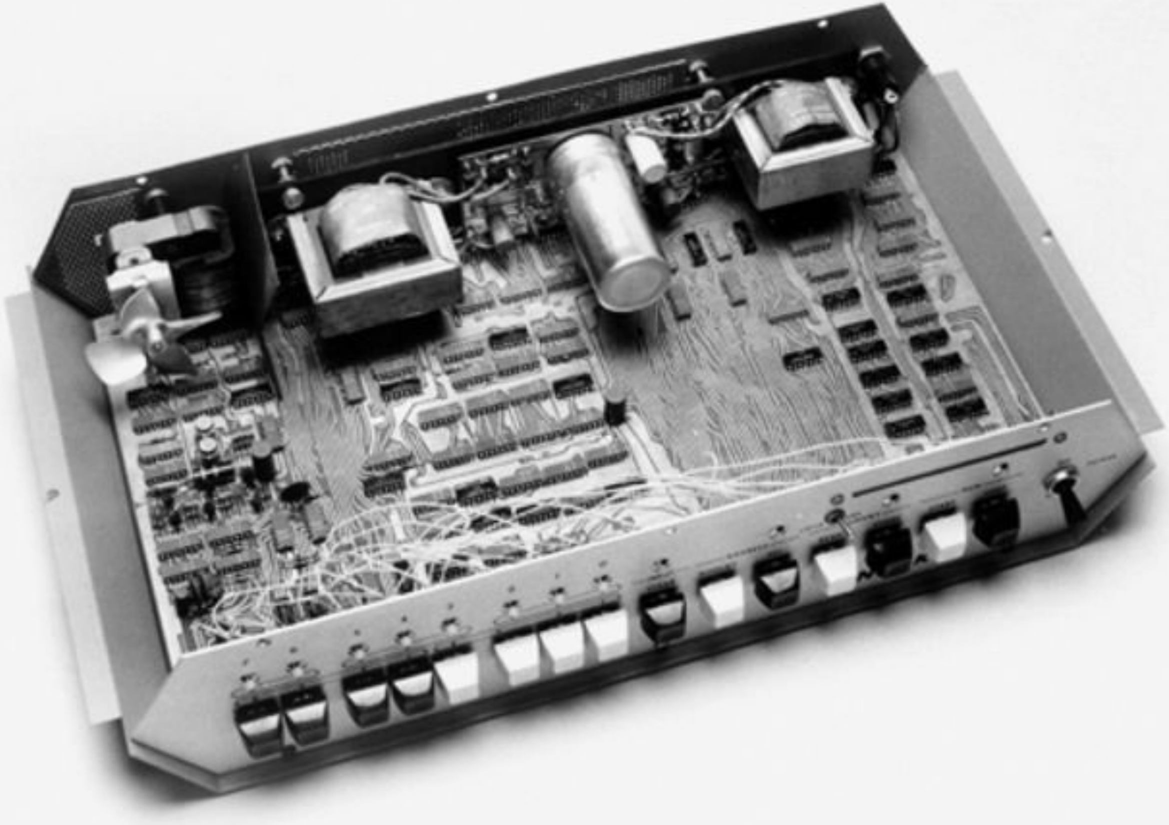
In the past, there were only full-stack craftsmen

They were called just "Programmers"

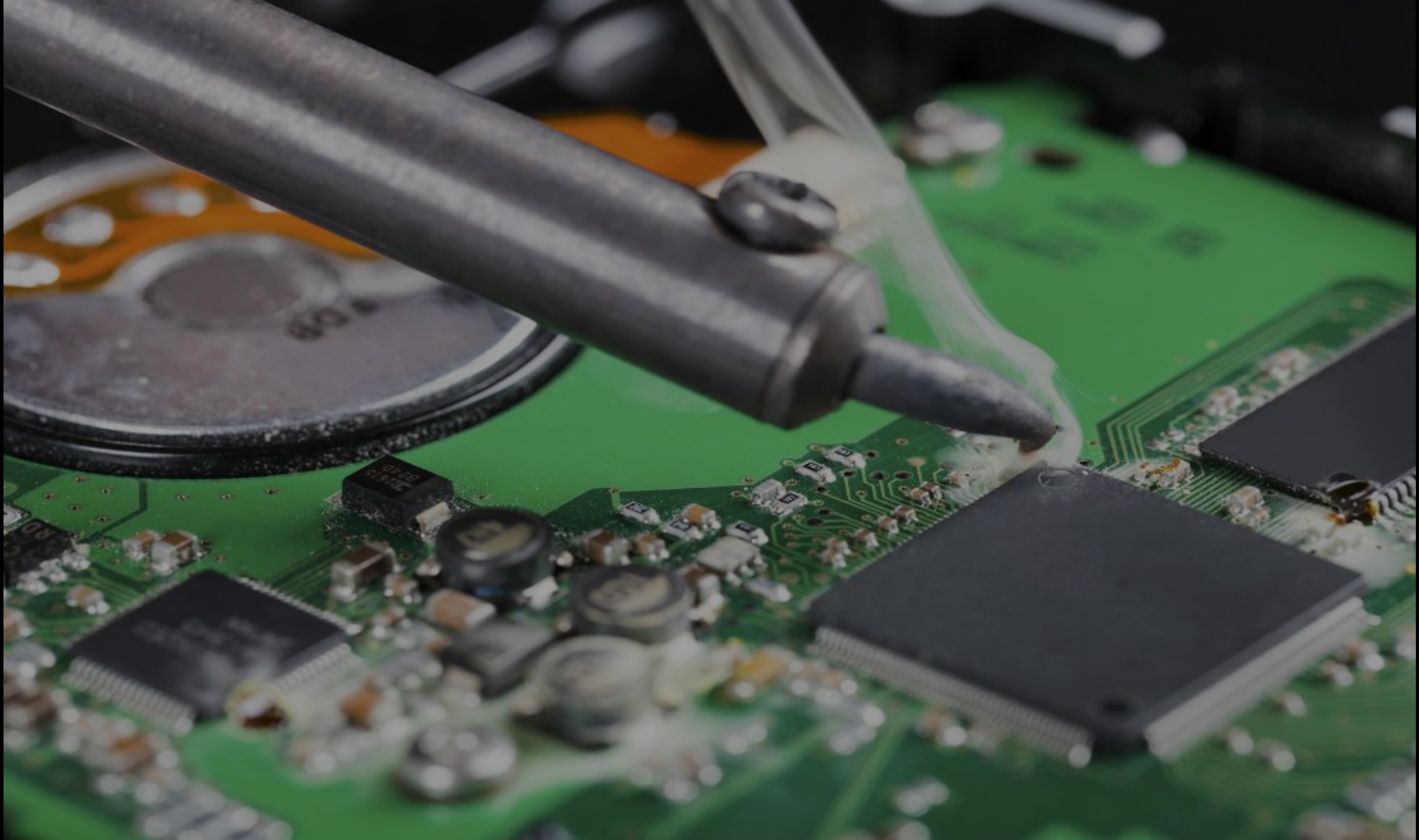
Most were actually women

Like “programmer”, the word “craftsman” is gender-neutral

You even had to deal with hardware a lot more



Then, everything has changed



Introduction to
Conflicts and toxicity in **IT**

DEVELOPER
TESTER

Admins vs Developers

IT became bigger and roles started to appear

Internet and Servers - somebody had to take care of them

Admins **hate** changes, but devs' job is to **change** things

DevOps movement is fixing this

"Giving admin rights to devs will result in chaos!" - I heard this in 2019!!!

2000s: DB vs App devs

Back in the days, Oracle has invented a new profession - DBA

Also the phrase “data assets”

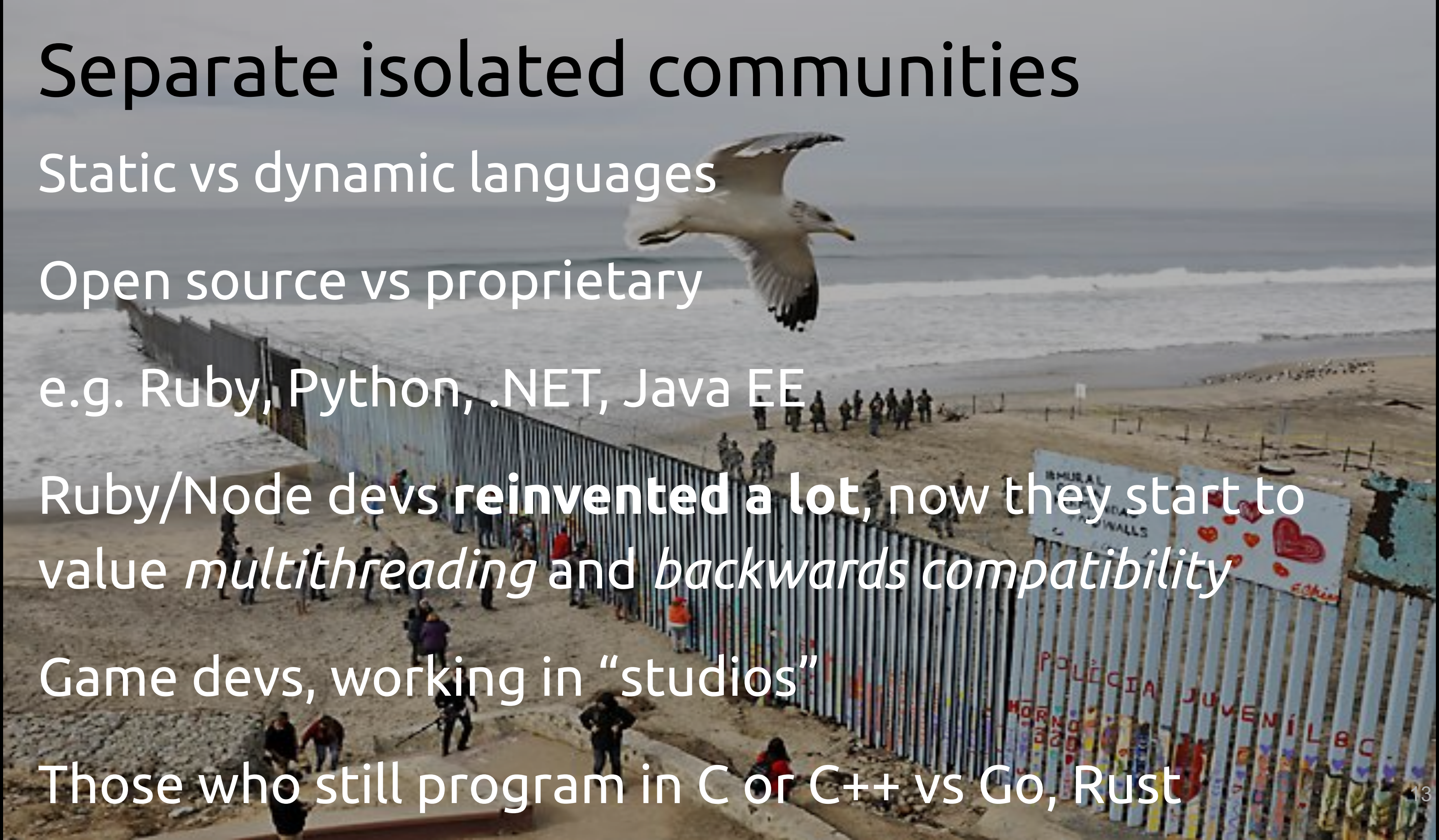
Different mindsets:
Logic **inside** or **outside**?

DB devs stayed conservative

Other extreme: let’s ditch RDBMS and re-implement everything manually with NoSQL solutions

Actual* email from MongoDB….

*minus the bro part



Separate isolated communities

Static vs dynamic languages

Open source vs proprietary

e.g. Ruby, Python, .NET, Java EE

Ruby/Node devs **reinvented a lot**, now they start to value *multithreading* and *backwards compatibility*

Game devs, working in “studios”

Those who still program in C or C++ vs Go, Rust

2010s - Rise of Frontend vs Backend devs

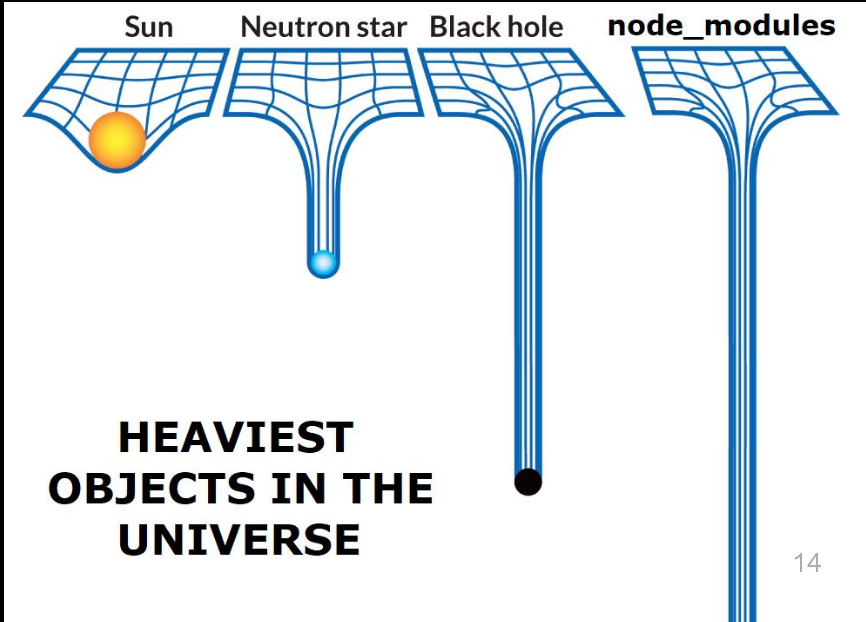
New trends in web UIs brought lots of complexity

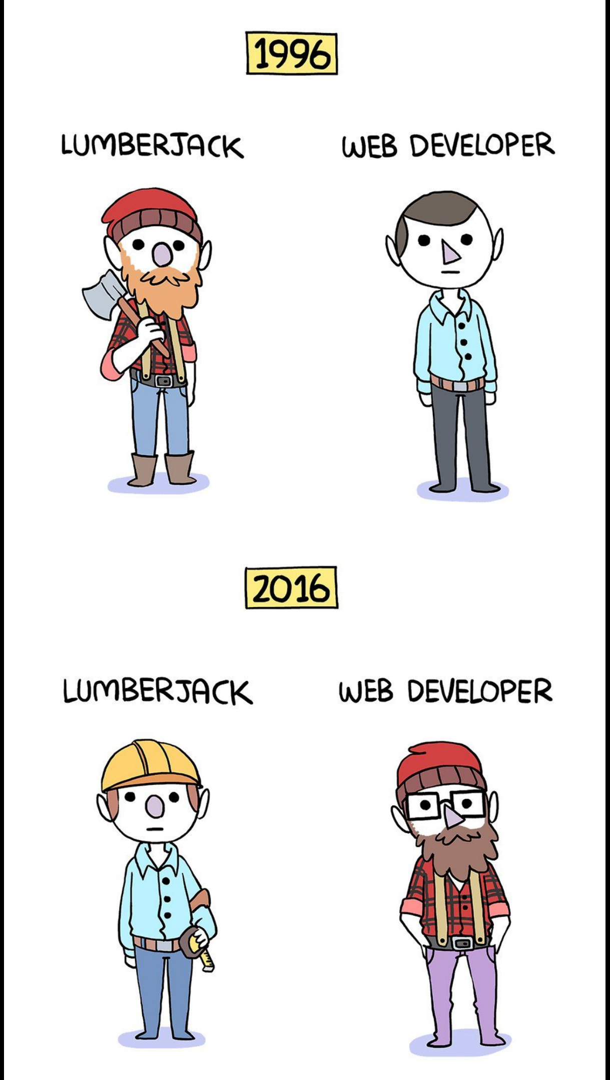
Clearer UI/backend separation was needed, different technologies

Young devs started to specialize in UI, and reinventing stuff

Ever-changing frameworks
Transpilation, Unit testing, etc

Backend devs now reduced to "API developers"




1996
LUMBERJACK
WEB DEVELOPER
2016
LUMBERJACK
WEB DEVELOPER

NOKIA
SAMSUNG
12:45
Thu, 3 May

In 2007
“The full Safari engine is inside of iPhone. And so, you can write amazing **Web 2.0 and Ajax** apps that look exactly and behave exactly like apps on the iPhone.”

Then came Jailbreak
Followed by AppStore

Overmarketing: many mobile apps are actually unnecessary, but we have more specialization now

Split communities of iOS vs Android vs ~~Windows Phone~~ vs …

Even Android devs reinvent what “backend” devs did for years

Build tools, testing, languages, patterns

Nowadays, companies reimplement the same UI at least **3x**
By separate teams, producing different bugs…

Plus there are backend devs (or even microservice teams)

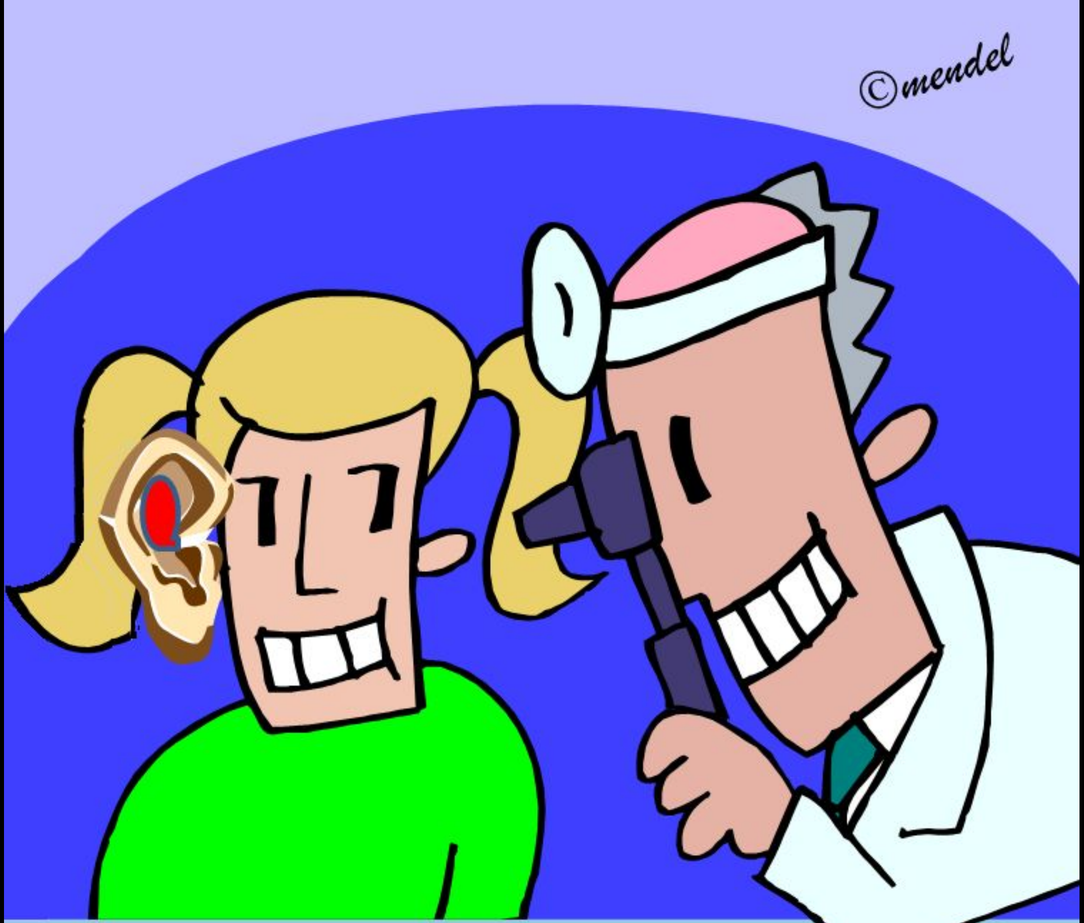
What a waste!

How incredibly similar they are!

But compare checked exceptions

I am **function-x** developer
And you are **function-y**

we are different



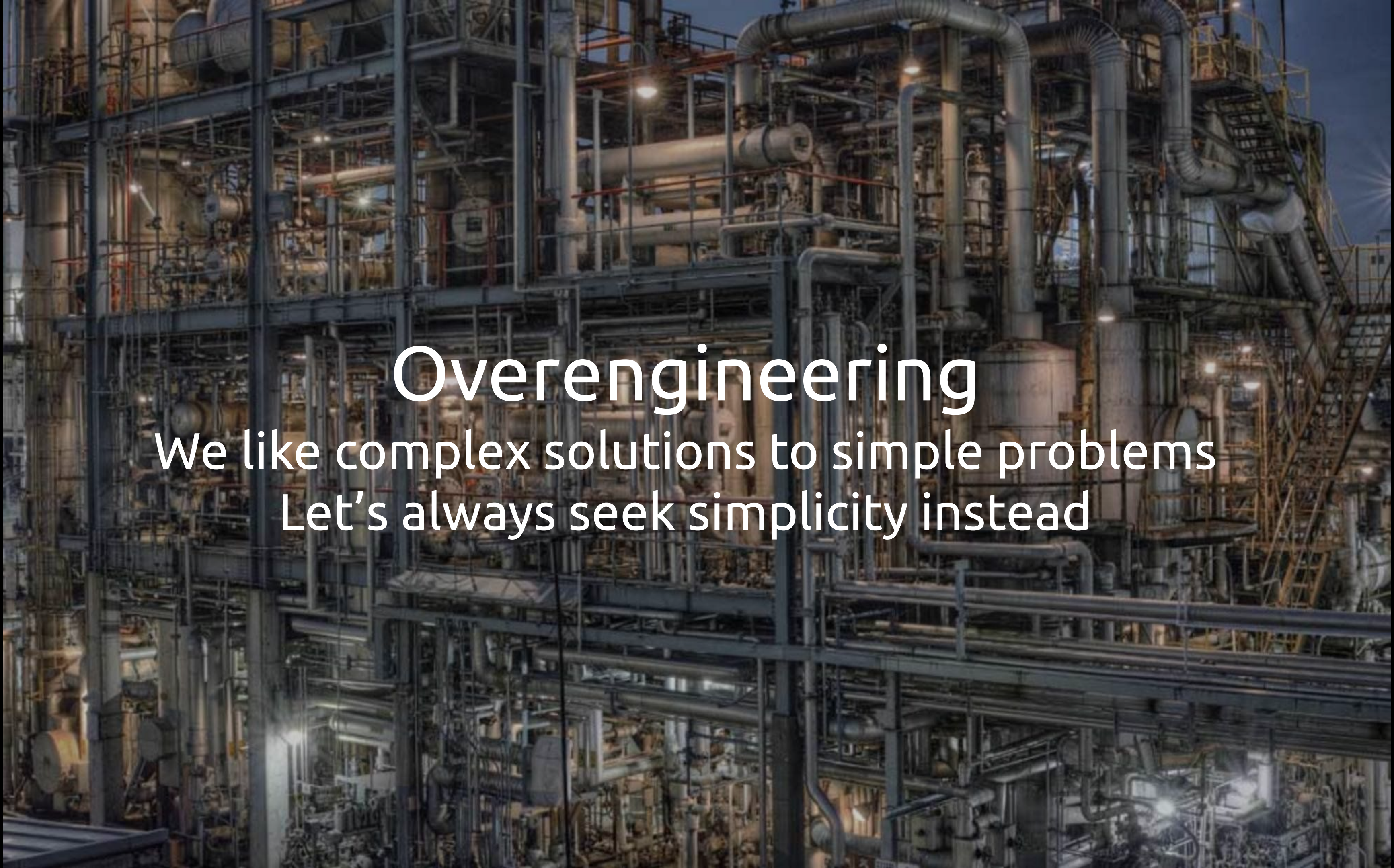
Your left ear looks great!
I’ll give you a referral to a “right ear guy”.

Overspecialization

Inflated teams

Low truck factor

Slow and expensive projects



Overengineering

We like complex solutions to simple problems
Let’s always seek simplicity instead

Full-stack developer

comes to the rescue

Broad-minded

Experienced in many fields/stacks

Can choose the right tool for the job

Can learn new technologies quickly

No need for blaming/”finger pointing”

A more rare kind in bigger markets/companies

Big = can afford being inefficient

The less important you become

THE FAILURE OF MY PROJECT CAN ONLY BE BLAMED ON. . .
BEEP BOOP BEEP
scottadams@aol.com
www.dilbert.com
I'LL BE RIGHT THERE.
9-14-05
© 2005 Scott Adams, Inc./Dist. by UFS, Inc.
THAT GUY.
© Scott Adams, Inc./Dist. by UFS, Inc.

Being a full-stack developer

XP: collective code ownership

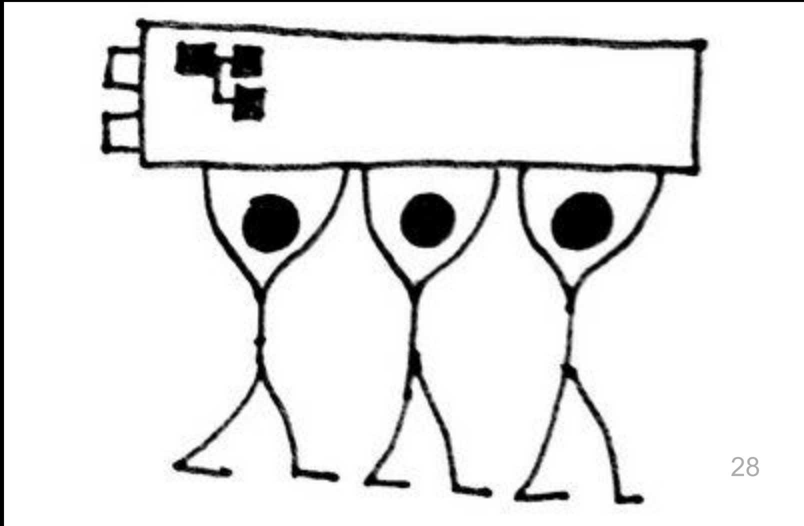
You build/learn all aspects of your project

You can contribute in any area

You don't leave anything to others

You are in control

Power = Responsibility



Becoming a full-stack developer

“Full-stack” refers to the collection of technologies needed to complete **your** project

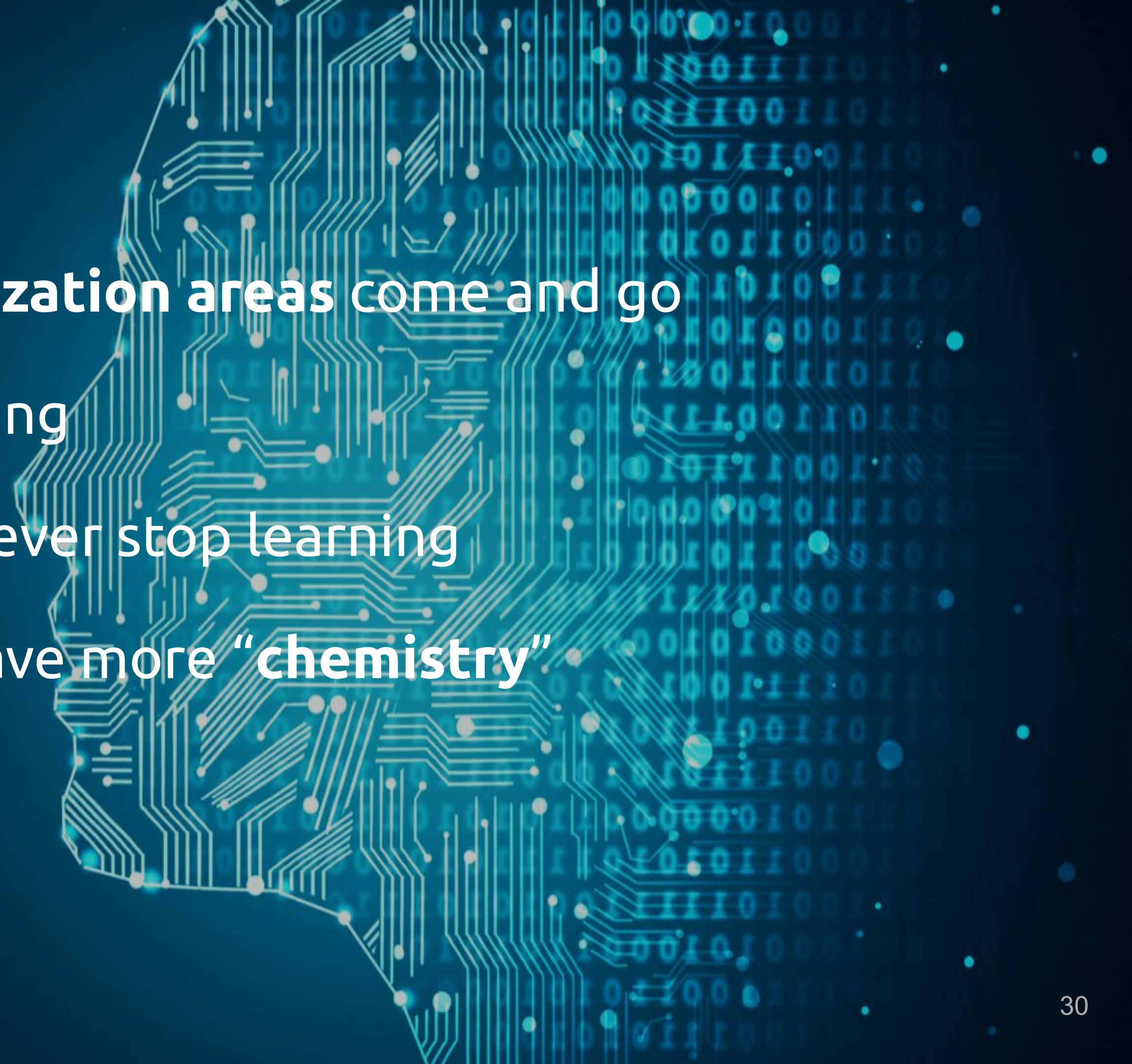
“Polyglot” developer

You learn the **essence** and can apply it in **any** language/technology

You still learn most needed parts **deeply** as you gain **experience**

Structuring, design, security, logging, auto testing, simplicity, etc

Also deployment - you don’t want to be called during the night



Why me?

Technologies and **specialization areas** come and go

AI and automation is coming

You need to be flexible, never stop learning

Multidisciplinary teams have more “**chemistry**”

Big picture

You understand where the actual problem is,
and where to apply the fix

(Instead of creating workarounds, and later fixing them again)

Easier not to overengineer

Brings **deep understanding** of how stuff works

Efficiency - less useless work, fewer mistakes

Your project is most likely **doomed** without a single Full-stack member

Architect? Not if they **overspecialize** and **stop coding!**

Side effects

You are more pragmatic

You don’t jump for every fresh and sexy thing (framework), only for it to be abandoned a year later

E.g. many webapps are better-off being server-side rendered

You know the costs of implementing things in one stack or another

You care about long-term maintainability

You avoid more invasive frameworks (that control you)

Do you know how to build your current project **from scratch**?

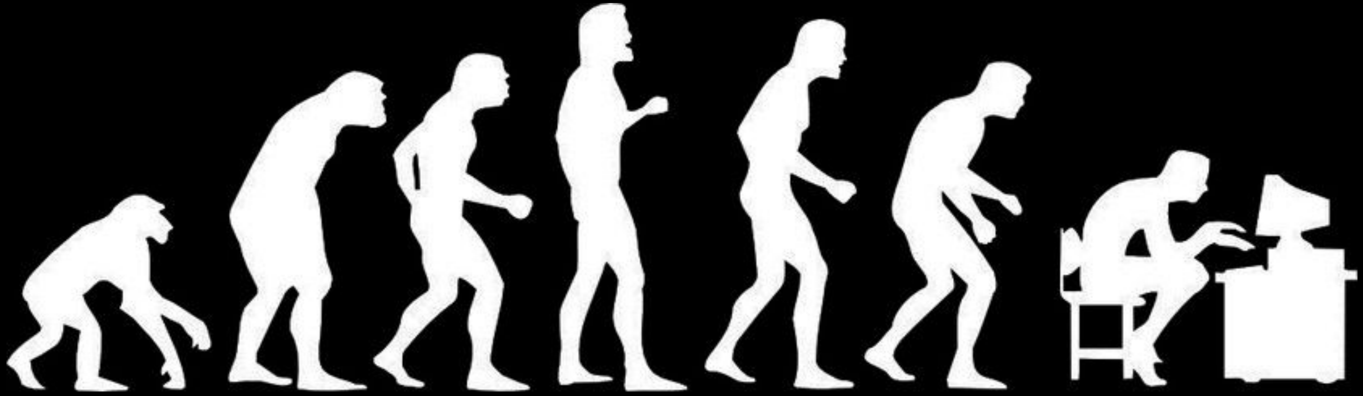
Including storytelling (requirements)?

Could you do it by yourself?

Would it be better than now?

I CAN T

Communication problems



Devs vs Testers, Analysts, PMs, ...

Swedbank in 2010: only 50 devs out of 700 IT personnel

Many “supporting roles” just because devs are not able to do their job properly

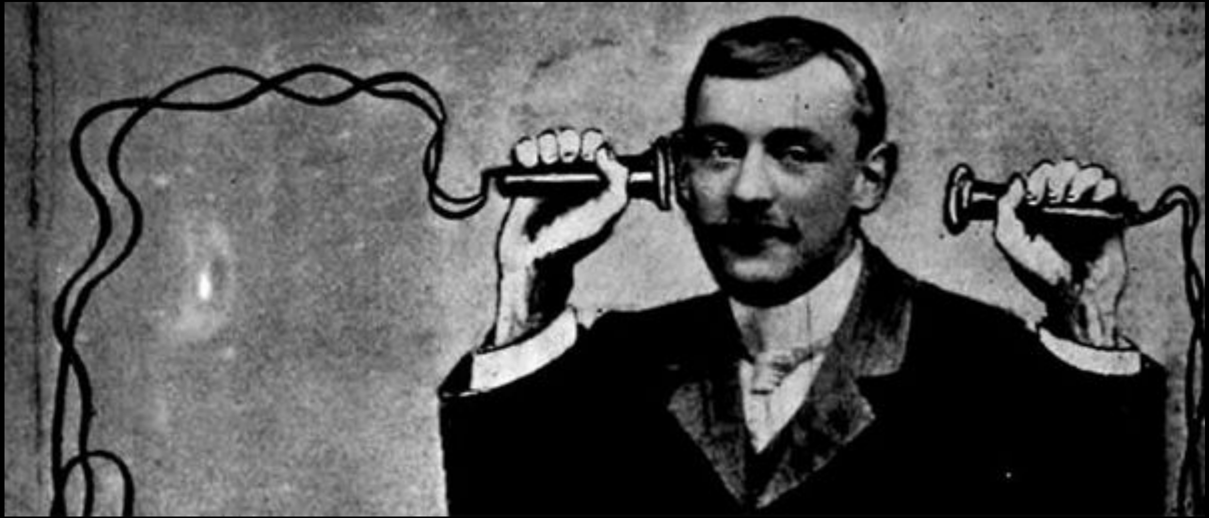
Necessity of analysts reduces devs to **code monkeys**

Allows not to develop communication skills, reducing efficiency

Lots of finger pointing, blaming, and “broken phone”

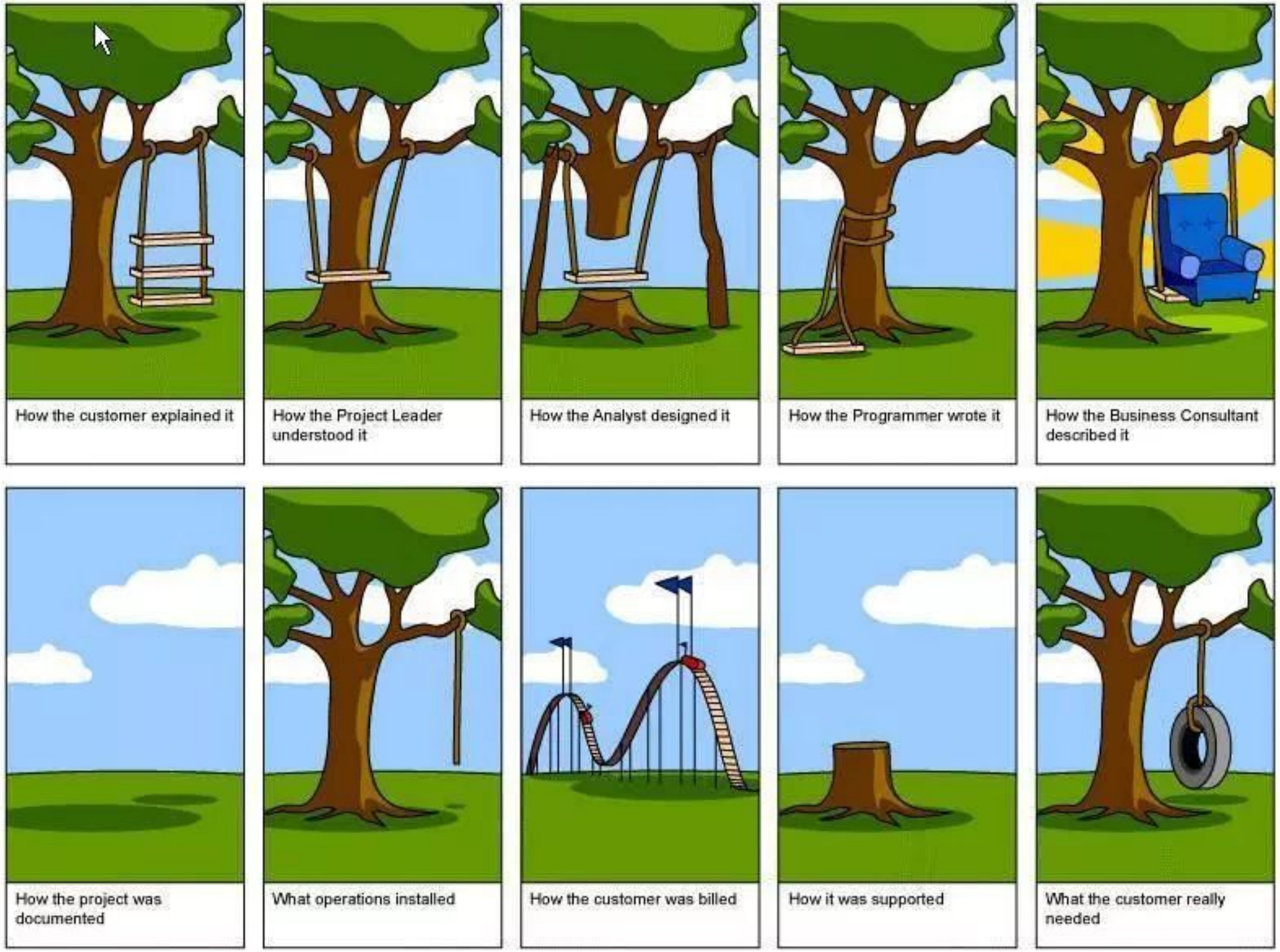
Chinese whispers / Broken phone game

Talking to customers through a middleman (proxy) makes **negotiations impossible**



Hoping for **testers** to find your bugs?

They should be the last line of defense


How the customer explained it
How the Project Leader understood it
How the Analyst designed it
How the Programmer wrote it
How the Business Consultant described it
How the project was documented
What operations installed
How the customer was billed
How it was supported
What the customer really needed

Of course, we still earn good money,
despite producing wrong results, inefficiently

Poor customers accept that they get bugs and wrong stuff from **IT**

I am sorry for those who can't get time to market of a few days because of slow **IT**

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Downfall of Agile

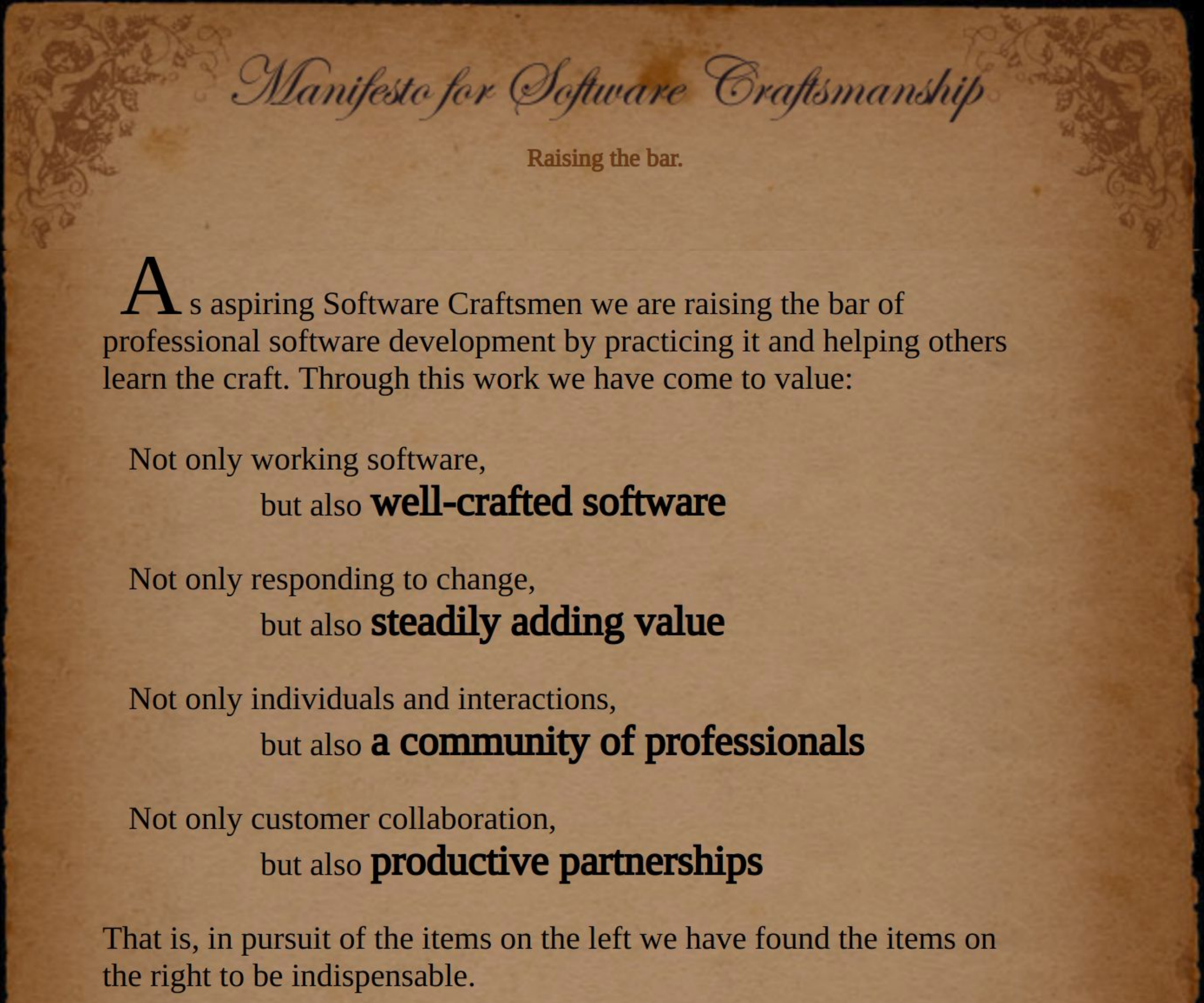
Nowadays everyone says they do some kind of Agile
Usually meaning top-bottom Scrum

Where is excellence?

It became a management **buzzword**

A **religion** that nobody knows how to practice

COME ON,
WE CAN DO BETTER THAN THIS



Manifesto for Software Craftsmanship

Raising the bar.

As aspiring Software Craftsmen we are raising the bar of professional software development by practicing it and helping others learn the craft. Through this work we have come to value:

Not only working software,
but also **well-crafted software**

Not only responding to change,
but also **steadily adding value**

Not only individuals and interactions,
but also **a community of professionals**

Not only customer collaboration,
but also **productive partnerships**

That is, in pursuit of the items on the left we have found the items on the right to be indispensable.

DEVELOPERS
DEVELOPERS
DEVELOPERS
DEVELOPERS

Craftsmen
Craftsmen
Craftsmen
Craftswomen

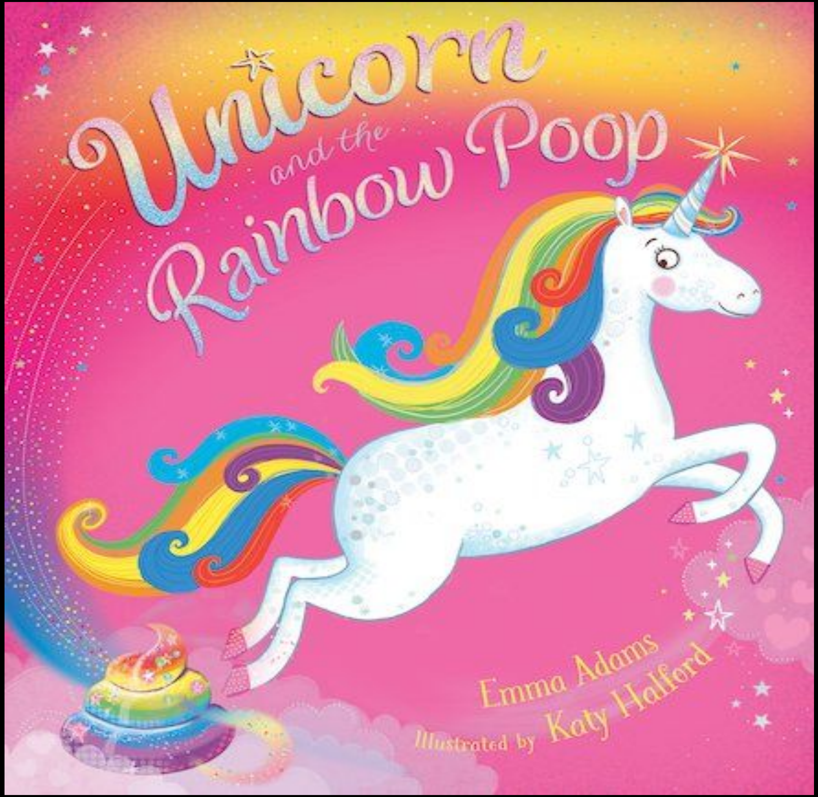
Software craftsman should be able to

- Talk to customer directly
- Understand the underlying problem, not how customer proposes to solve it
- Propose solutions
- Break the problem into small chunks, write down as user-centric stories
- Design UI flow
- Write working code ← Old-fashioned software developer
- Write automated tests to avoid regressions
- Deploy the system to the end users (“DevOps”)
- Evolve the system design/architecture by refactoring

It’s also more creative and fun that way

Codeborne and Digital Prescriptions

And what if solution is not only in the IT? It happens quite often



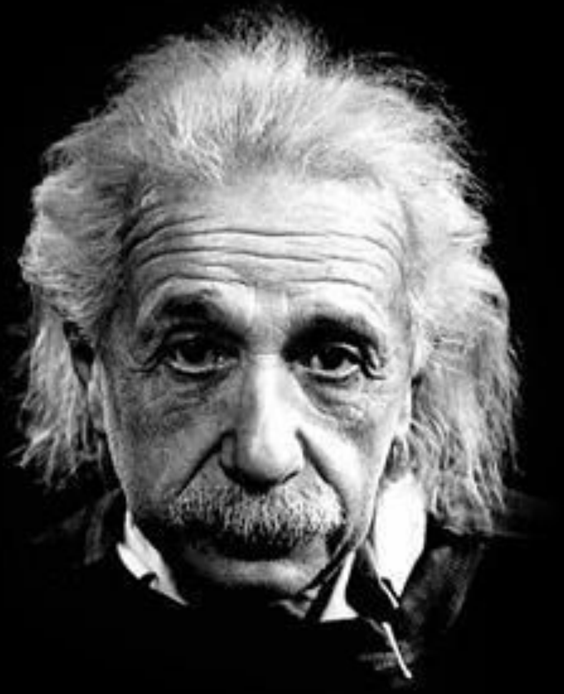
Startup people should do **full-stack craftsmanship** to survive

Everyone should try as well!

But nowadays, too much money

(and spending not your own)

makes also startups **inefficient**



"Make everything as simple as possible, but not simpler."
—Albert Einstein

We May Need More Than
Refactoring to Fix This!

Underengineering

The other extreme

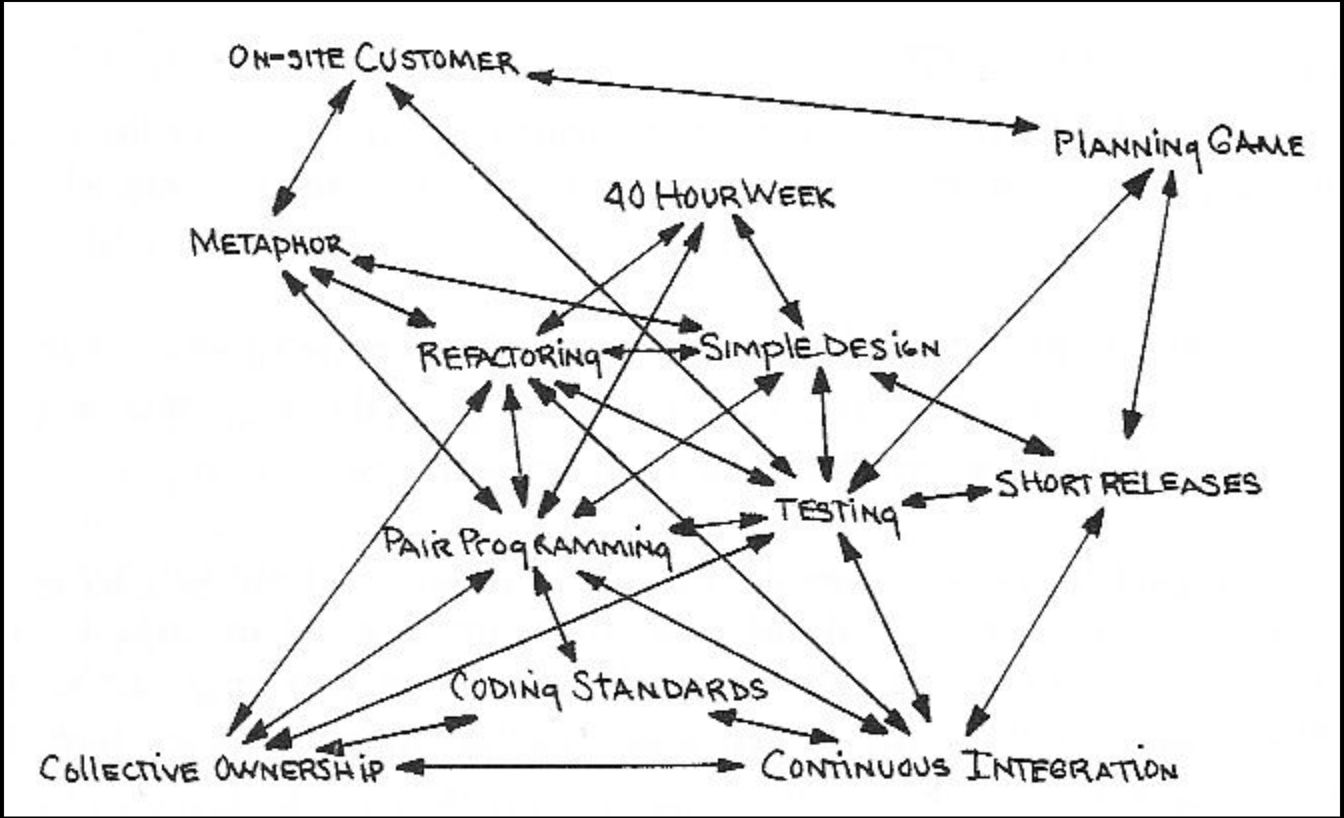
Very common in startups, and elsewhere

You don't have time to write bad code because it slows you down

Lack of basic practices

e.g. lack of Continuous Integration **always** leads to a broken project

Extreme Programming (XP) practices



At the 3rd level of professionalism

1. You are very good at doing it
2. You are so good, so you can innovate
3. You are so good, so you can teach others to innovate, too

(then your innovation becomes the new norm)

Project routine in Codeborne

- Before start: make sure we have **business** and **tech** contacts
- Kick-off meeting with them
    - Storytelling
- Iterations (1 week)
    - Stand-up meetings (mostly over video)
    - Developers focus on user stories
    - Continuous integration server builds and tests every change
    - Continuous delivery to a demo server
- Iteration planning
    - Demo
    - Prioritization & storytelling
- Until agreed deadline or can finish anytime for any reason

Advice

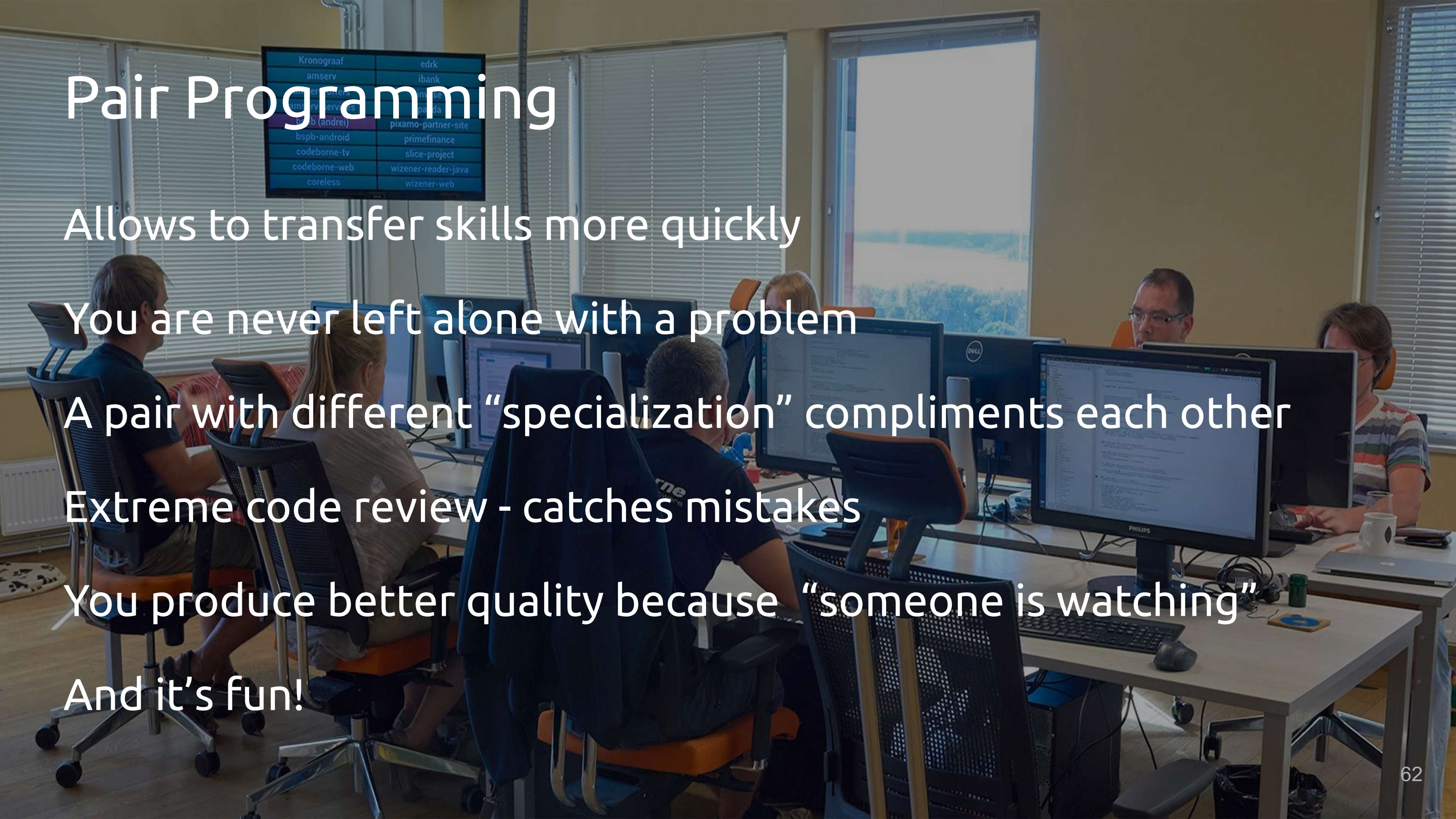
If you think you are good enough technically - try low level, drivers, assembler, etc

But also learn to explain technical concepts to normal people

Differentiate important from not-so-important

Do you have pet projects? GitHub account?

Next level: start a business



Pair Programming

Allows to transfer skills more quickly

You are never left alone with a problem

A pair with different “specialization” compliments each other

Extreme code review - catches mistakes

You produce better quality because “someone is watching”

And it’s fun!

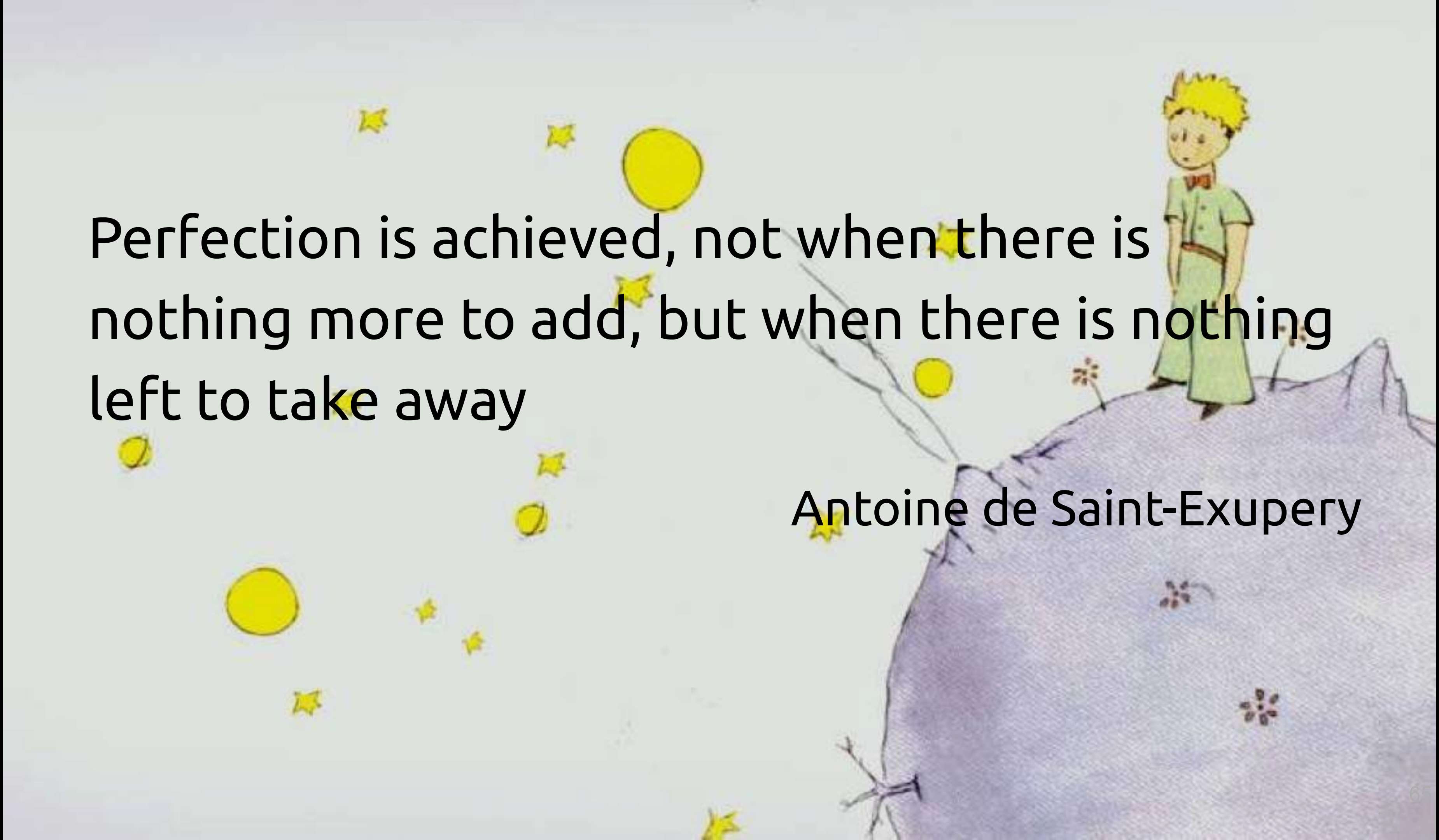
Good developer can be **5x** more productive

Craftsman can be **5x** even more efficient by knowing what **not to do**

We not only write code, but solve problems

Surely you want to be one

Wasamuseet, Stockholm



Perfection is achieved, not when there is nothing more to add, but when there is nothing left to take away

Antoine de Saint-Exupery

codeborne
well-crafted software

Anton Keks

@antonkeks