Espeo Software

Software development myths that block your career

Piotr Horzycki
peterdev.pl

Focus on the right thing

The 80/20 rule (Pareto principle)

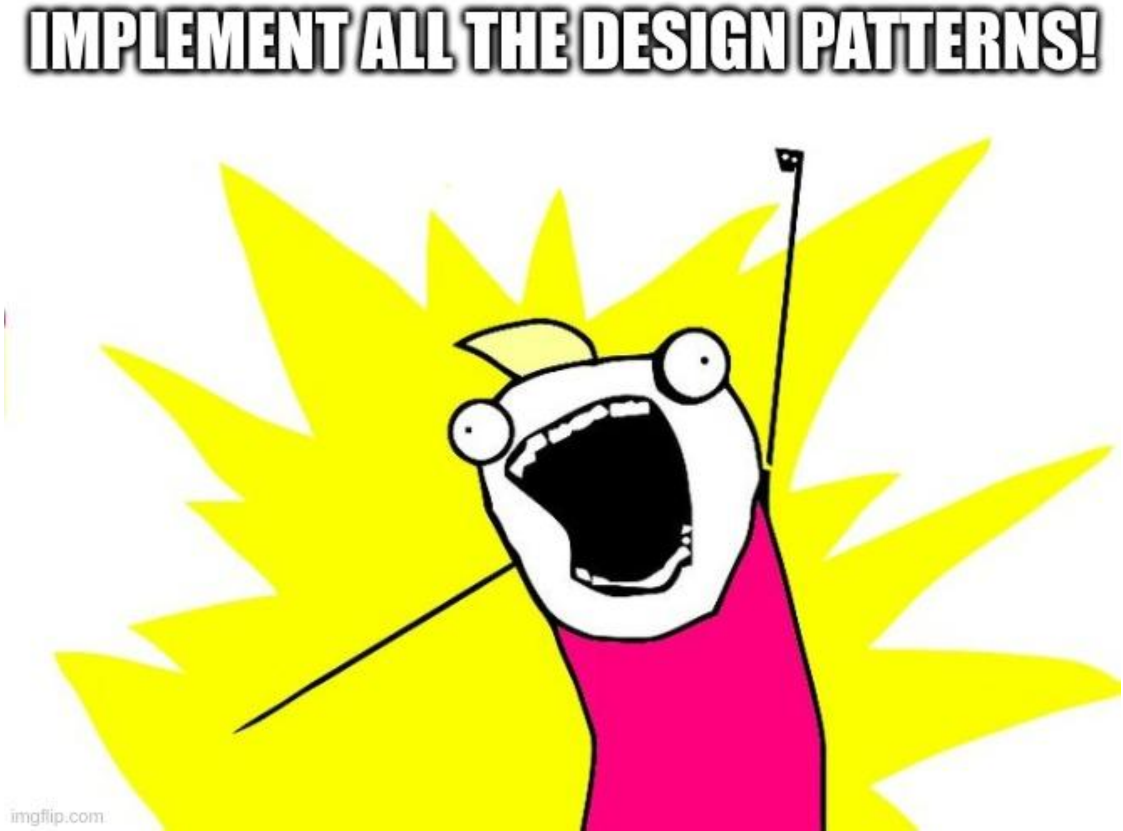
Become a problem solver, not maker!

Sophisticated architecture

+

Hype-Driven Development

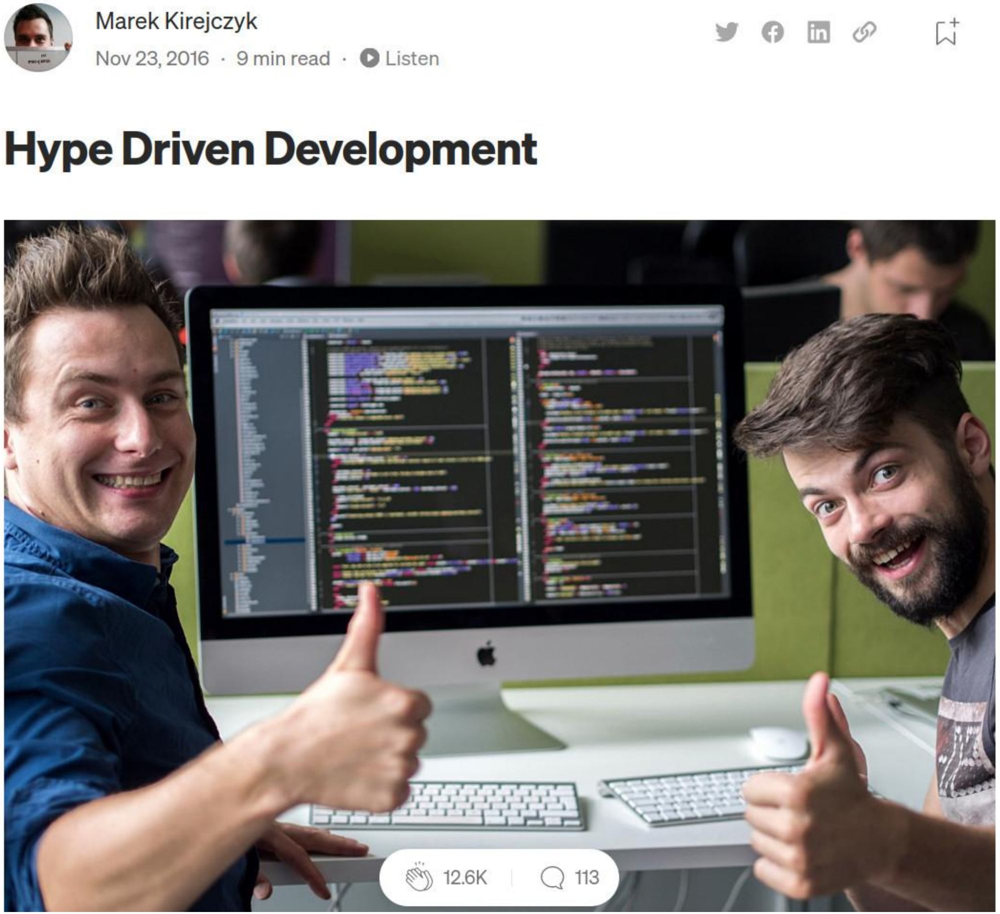
Hype-Driven Development



Hype-Driven Development

Not every system requires:

- ORM
- CQRS
- Hexagonal Architecture
- Microservices
- Machine Learning
- ...



Marek Kirejczyk

Nov 23, 2016 · 9 min read · Listen

Hype Driven Development

https://blog.daftcode.pl/hype-driven-development-3469fc2e9b22?gi=a52a6e685546

Key benefits

- You can be the voice of reason

Key benefits

- You can be the voice of reason
- Match solutions to problems

Key benefits

- You can be the voice of reason
- Match solutions to problems
- Faster development

Key benefits

- You can be the voice of reason
- Match solutions to problems
- Faster development
- Save project from missing deadlines and failure

Key benefits

- You can be the voice of reason
- Match solutions to problems
- Faster development
- Save project from missing deadlines and failure
- More time for important things

Key benefits

- You can be the voice of reason
- Match solutions to problems
- Faster development
- Save project from missing deadlines and failure
- More time for important things
- Reduce costs

Key benefits

- You can be the voice of reason
- Match solutions to problems
- Faster development
- Save project from missing deadlines and failure
- More time for important things
- Reduce costs
- Easier team building

100%

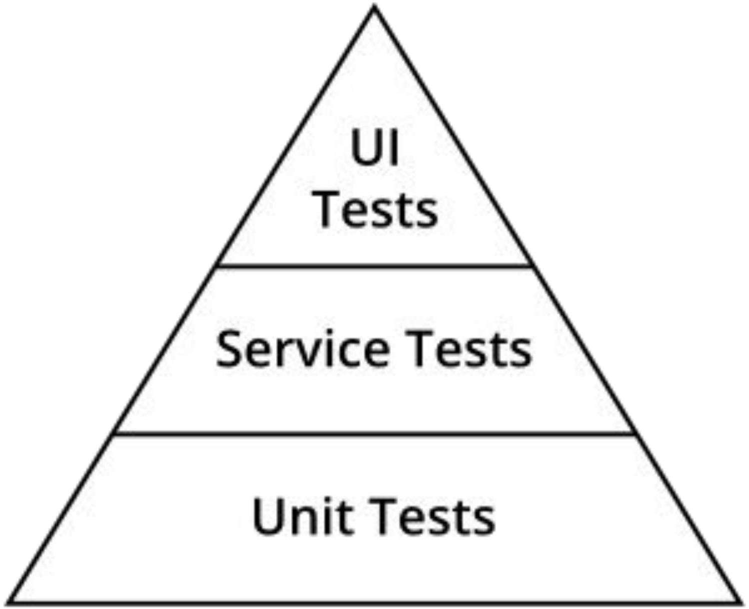
code
coverage

Know your metrics

Line coverage is a tool to find untested code.

But it’s **not enough** to determine the **quality of tests.**

Know your metrics



What’s the coverage?

On a scale from 1 to 10, how much do we **trust** these tests?

Know your metrics

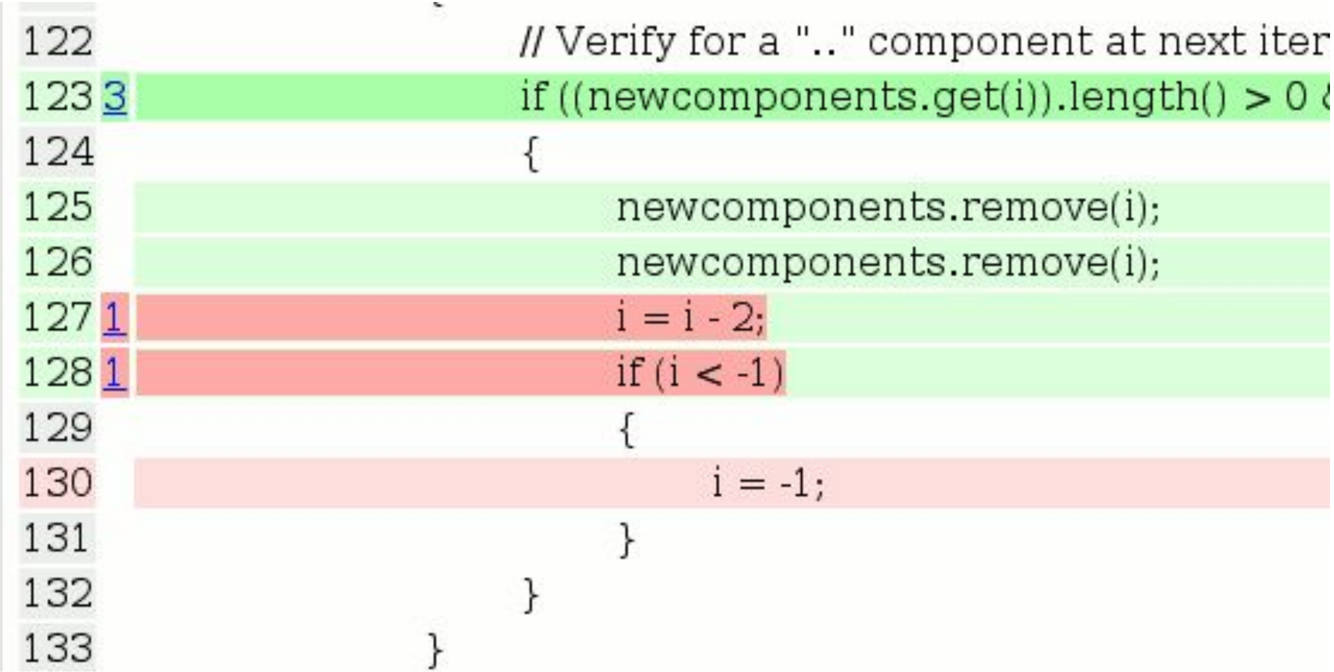
```
if (something) {
    doThis();
}

doOtherThing();
```

100% line coverage

50% branch coverage

🤷

Mutation testing



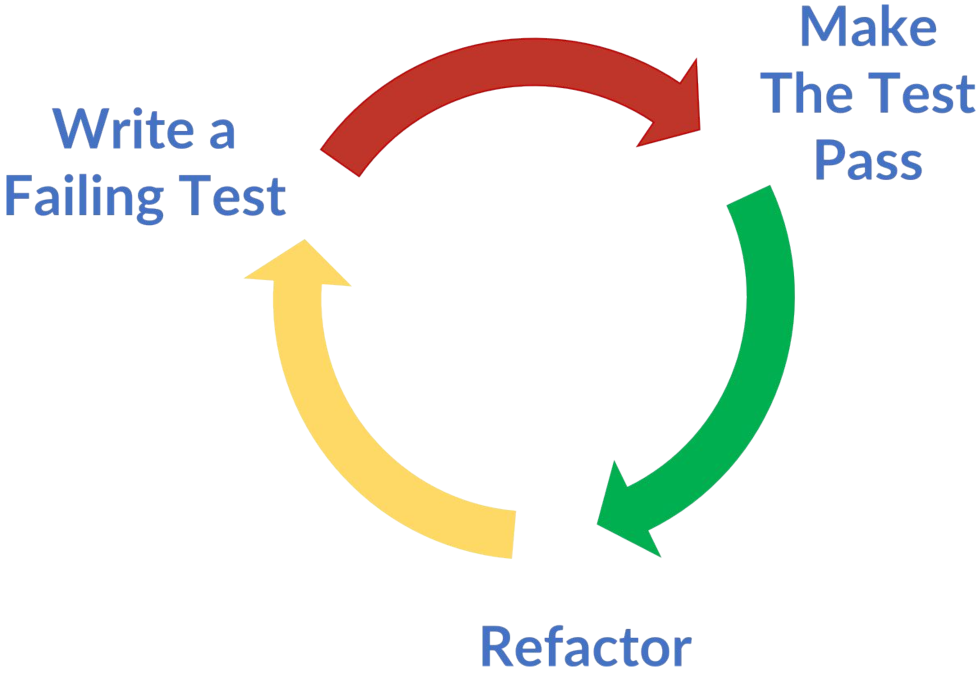
```
    // Verify for a ".." component at next iter
    if ((newcomponents.get(i)).length() > 0 &
    {
        newcomponents.remove(i);
        newcomponents.remove(i);
        i = i - 2;
        if (i < -1)
        {
            i = -1;
        }
    }
}
```

Mutation testing tells you how precise your tests are

Make refactoring easier

Too strict unit tests limit the possibility of refactoring.

```
mock1
     .method('doThis')
     .expect(times(3))
     .withArguments(...);

mock2
     .method('doThat')
     .expect(times(2))
     .withArguments(...);
```



Better testing strategy: Key benefits

- Improved quality
- Less bugs
- More stability
- Easier refactoring

Rewrite everything?!

Pros and cons of rewrites

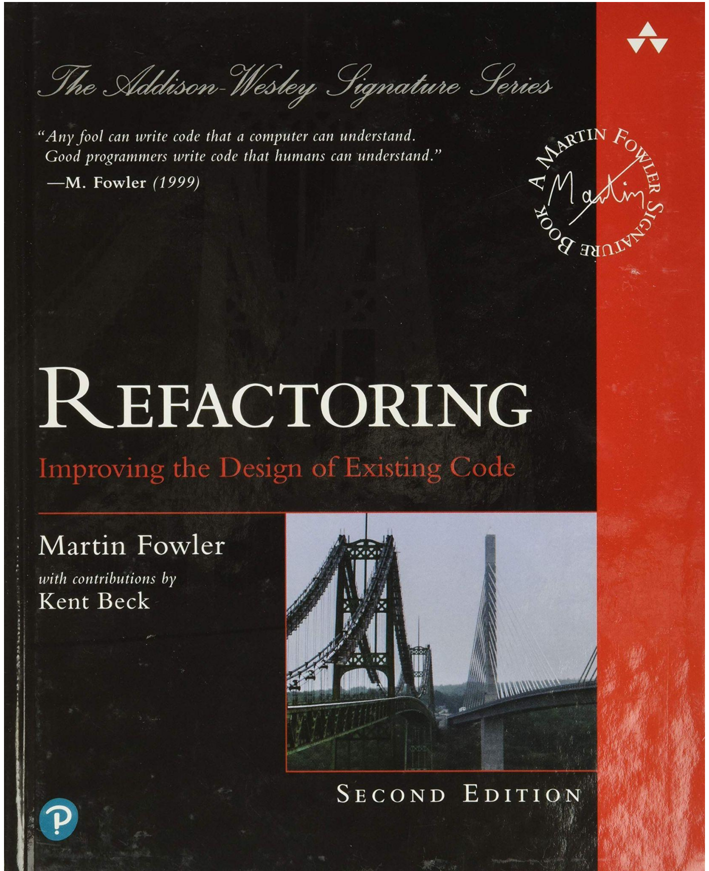
Old system = old bugs

New system = new bugs

Rewrite might be necessary (like termination of Adobe Flash)

…but usually it’s not

Pros and cons of rewrites


The Addison-Wesley Signature Series
"Any fool can write code that a computer can understand. Good programmers write code that humans can understand."
—M. Fowler (1999)
A Martin Fowler Signature Book
Martin
Refactoring
Improving the Design of Existing Code
Martin Fowler
with contributions by
Kent Beck
Second Edition

"The project management was not pleased. (...) two programmers had spent two days doing **work that added nothing** to the many features the system had to deliver in a few months' time. The old code had worked just fine. Yes, the design was a bit more "pure" and a bit more "clean." But the project had to **ship code that worked**, not code that would please an academic. (...)
Six months later, the project failed."

Strategies for legacy code

- Writing tests

Strategies for legacy code

- Writing tests
- Refactoring step by step

Strategies for legacy code

- Writing tests
- Refactoring step by step
- Event Storming

Strategies for legacy code

- Writing tests
- Refactoring step by step
- Event Storming
- Facade
- Strangler Pattern
- Anti-Corruption Layer

Strategies for legacy code

Action	Cost	Expected benefit
Rewrite the shopping cart	XL	M
Refactor user registration	L	L
Optimize SQL for listing products	M	L

Strategies for legacy code

Action	Cost	Expected benefit
Rewrite the shopping cart	XL	M
Refactor user registration	L	L
Optimize SQL for listing products	M	L

Refactoring strategy: Key benefits

- Save project from missing deadlines and failure
- Preserve the existing behavior of a system
- More time for important things

Refactoring strategy: Key benefits

- Save project from missing deadlines and failure
- Preserve the existing behavior of a system
- More time for important things
- Small step improvements are visible quickly
- Dopamine shots, avoid burnout
- Small gains accrued give big wins

We must have Scrum…?

Nervous Scrum?

Sprints can make people stressed because they believe that:

- They must do all the planned work
- They must not change the scope of the sprint

Wrong!

“Fixing” Scrum

New tools for Retrospectives

More color post-its

Better estimates

More bureaucracy

Scrum values: start from there!

Commitment

Focus

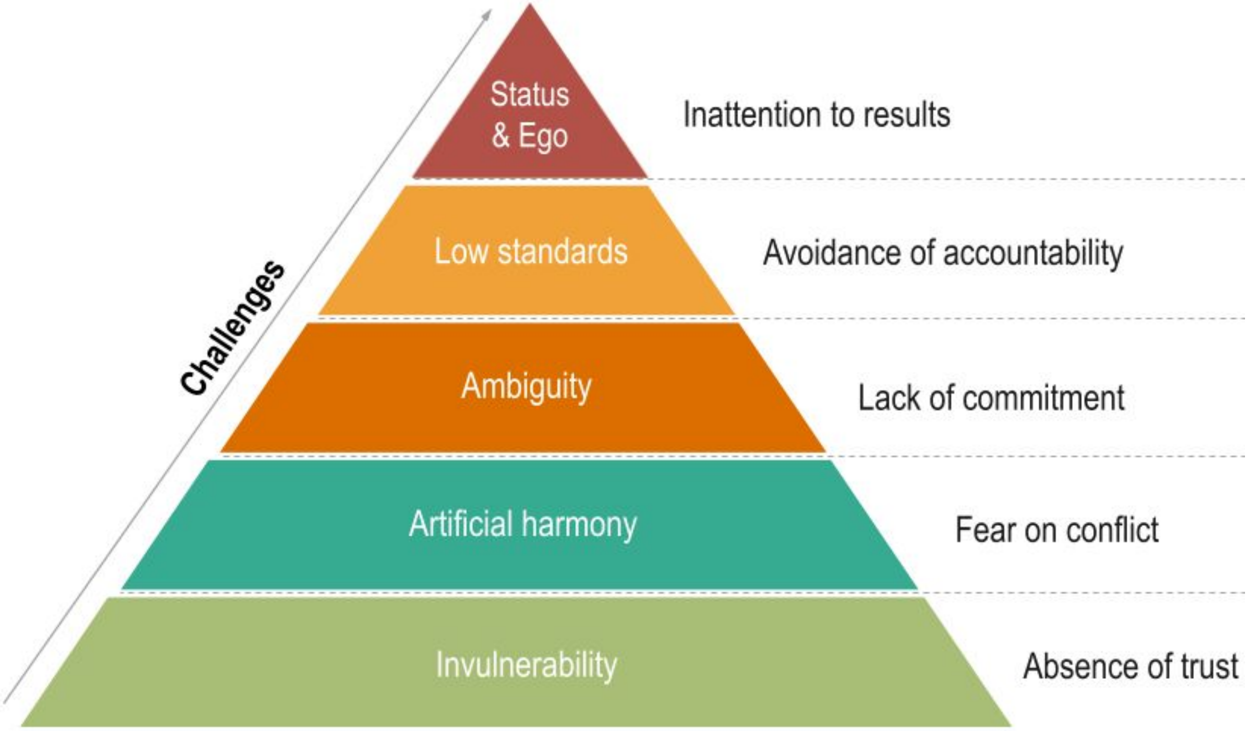
Openness

Respect

Courage

https://scrumguides.org/scrum-guide.html#scrum-values

Five dysfunctions of a team



https://www.strategypunk.com/leadership-5-dysfunctions-of-a-team-powerpoint-template/

Improving team performance

- Team Canvas
- 1 on 1
- “How-to” for other teams
- Invite other teams for your Sprint Review

Improving team performance

- Team Canvas
- 1 on 1
- “How-to” for other teams
- Invite other teams for your Sprint Review
- Conway’s Law

Improving team performance

- Team Canvas
- 1 on 1
- “How-to” for other teams
- Invite other teams for your Sprint Review
- Conway’s Law
- Extreme Ownership

Improving team performance

- Team Canvas
- 1 on 1
- “How-to” for other teams
- Invite other teams for your Sprint Review
- Conway’s Law
- Extreme Ownership
- Scrum alternatives (Kanban, Fast Agile)

Key benefits

- Improve the company culture
- Reduce stress
- Align around common goals
- Practice soft skills
- Better systems architecture

Meetings are a waste of time…?

“Ok, let’s get back to work”

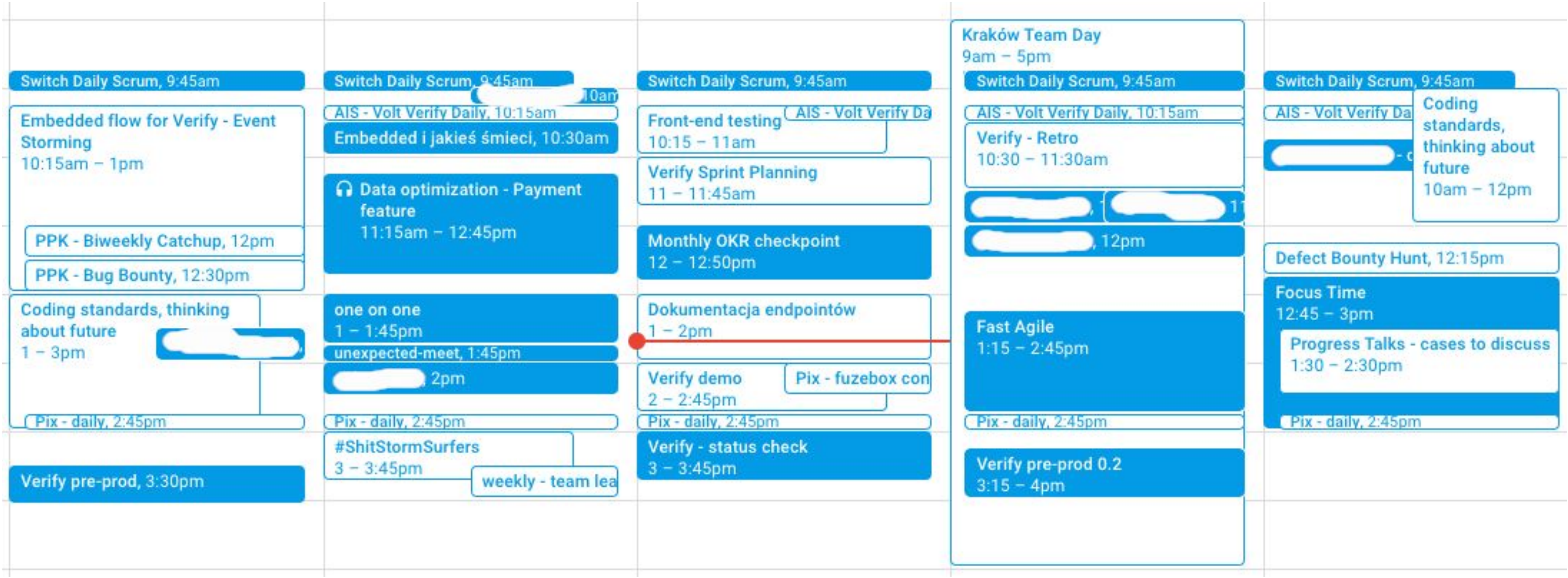
“Ok, let’s get back to work”

“This could have been an email”

Development is teamwork

You need to work together,
but also respect each other’s time

Improving meetings



Switch Daily Scrum, 9:45am

Embedded flow for Verify - Event Storming
10:15am – 1pm

PPK - Biweekly Catchup, 12pm

PPK - Bug Bounty, 12:30pm

Coding standards, thinking about future
1 – 3pm

[redacted]

Pix - daily, 2:45pm

Verify pre-prod, 3:30pm

Switch Daily Scrum, 9:45am

[redacted] 10am

AIS - Volt Verify Daily, 10:15am

Embedded i jakieś śmieci, 10:30am

Data optimization - Payment feature
11:15am – 12:45pm

one on one
1 – 1:45pm

unexpected-meet, 1:45pm

[redacted] 2pm

Pix - daily, 2:45pm

#ShitStormSurfers
3 – 3:45pm

weekly - team lea

Switch Daily Scrum, 9:45am

Front-end testing
10:15 – 11am

AIS - Volt Verify Da

Verify Sprint Planning
11 – 11:45am

Monthly OKR checkpoint
12 – 12:50pm

Dokumentacja endpointów
1 – 2pm

Verify demo
2 – 2:45pm

Pix - fuzebox con

Pix - daily, 2:45pm

Verify - status check
3 – 3:45pm

Kraków Team Day
9am – 5pm

Switch Daily Scrum, 9:45am

AIS - Volt Verify Daily, 10:15am

Verify - Retro
10:30 – 11:30am

[redacted]

[redacted] 11

[redacted] 12pm

Fast Agile
1:15 – 2:45pm

Pix - daily, 2:45pm

Verify pre-prod 0.2
3:15 – 4pm

Switch Daily Scrum, 9:45am

AIS - Volt Verify Da

Coding standards, thinking about future
10am – 12pm

[redacted] - c

Defect Bounty Hunt, 12:15pm

Focus Time
12:45 – 3pm

Progress Talks - cases to discuss
1:30 – 2:30pm

Pix - daily, 2:45pm

Improving meetings

- A meeting should have an agenda
- Do we need everyone on this call?
- Don't be afraid to reject a meeting

Improving meetings

- A meeting should have an agenda
- Do we need everyone on this call?
- Don't be afraid to reject a meeting
- Moderation, getting to the point
- Timeboxing
- Focus (avoid distraction)

Improving meetings

- A meeting should have an agenda
- Do we need everyone on this call?
- Don't be afraid to reject a meeting
- Moderation, getting to the point
- Timeboxing
- Focus (avoid distraction)
- Meeting notes, action points
- Collaboration tools

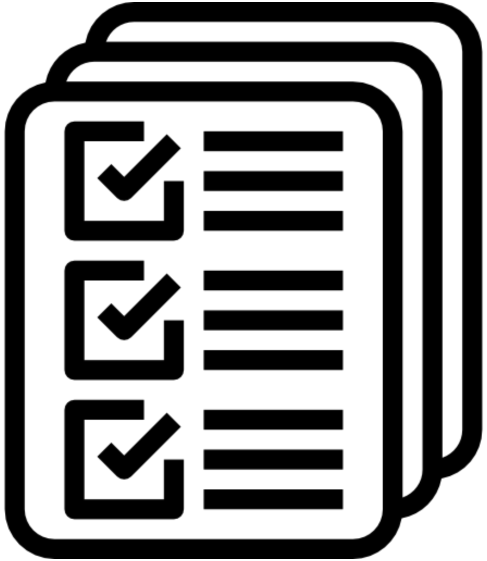
Key benefits

- Improve the company culture
- Reduce stress
- Better productivity
- Better alignment, more commitment

Ticking all the boxes

Average IT job offer

- Years of experience
- AWS, BDD, DDD, TDD, ABC, XYZ
- Tons of fancy keywords
- Gimmicks



Result: Impostor syndrome

As a candidate

You don’t have
to tick all the boxes

As a recruiter

You can’t expect people
to tick all the boxes

Philosophers team

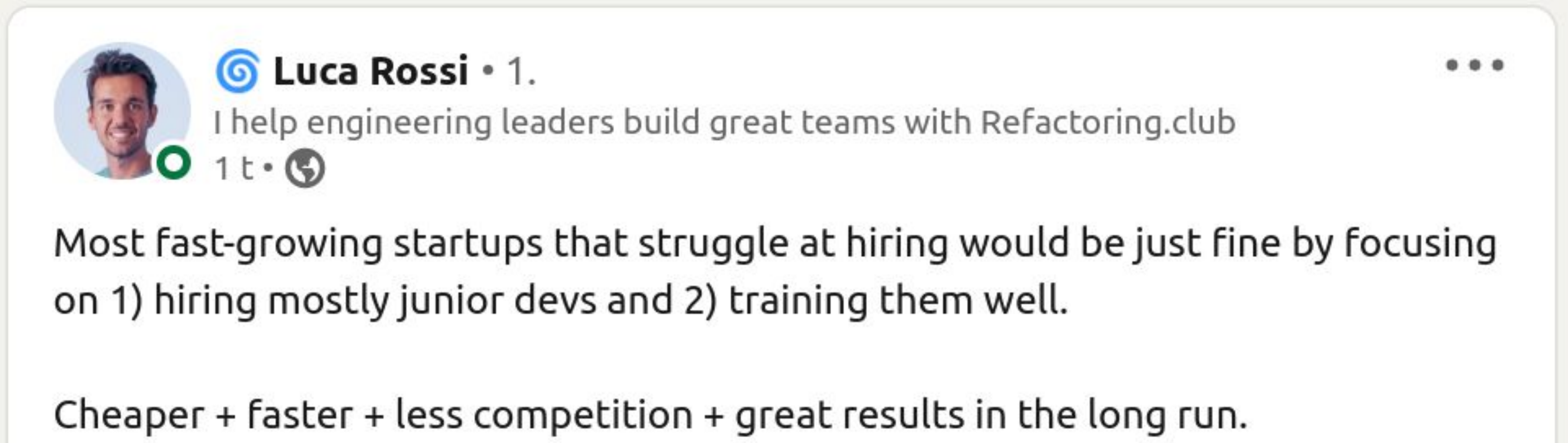
Socrates scores, got a beautiful cross from Archimedes. The Germans are disputing it.
Hegel is arguing that the reality is merely an a priori adjunct of non-naturalistic ethics.
Kant, via the categorical imperative, is holding that, ontologically, it exists only in the imagination.
And Marx is claiming it was offside.

Diverse team

Everyone’s awesome at something different:

- Architecture
- Design
- Management
- Performance
- Security
- Scalability
- Testing

Diverse team



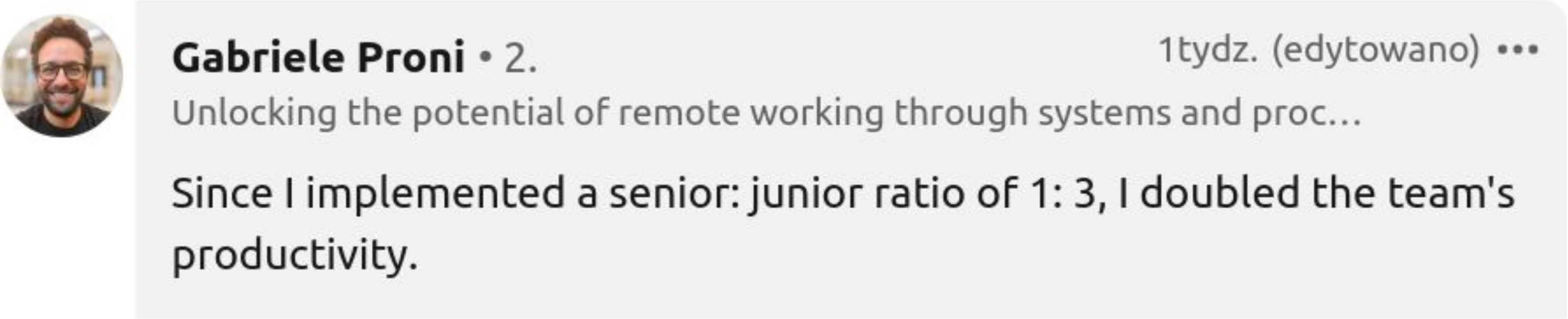
Luca Rossi • 1.

I help engineering leaders build great teams with Refactoring.club

1 t •

Most fast-growing startups that struggle at hiring would be just fine by focusing on 1) hiring mostly junior devs and 2) training them well.

Cheaper + faster + less competition + great results in the long run.



Gabriele Proni • 2.

1tydz. (edytowano)

Unlocking the potential of remote working through systems and proc…

Since I implemented a senior: junior ratio of 1: 3, I doubled the team's productivity.

Key benefits

- Fight the impostor syndrome
- Build strong, cross-functional teams
- Better software quality
- Faster onboarding
- Reduce costs

Become a real pro

by focusing on the right things :)

Espeo
Software

Thank you!

hi@espeo.eu