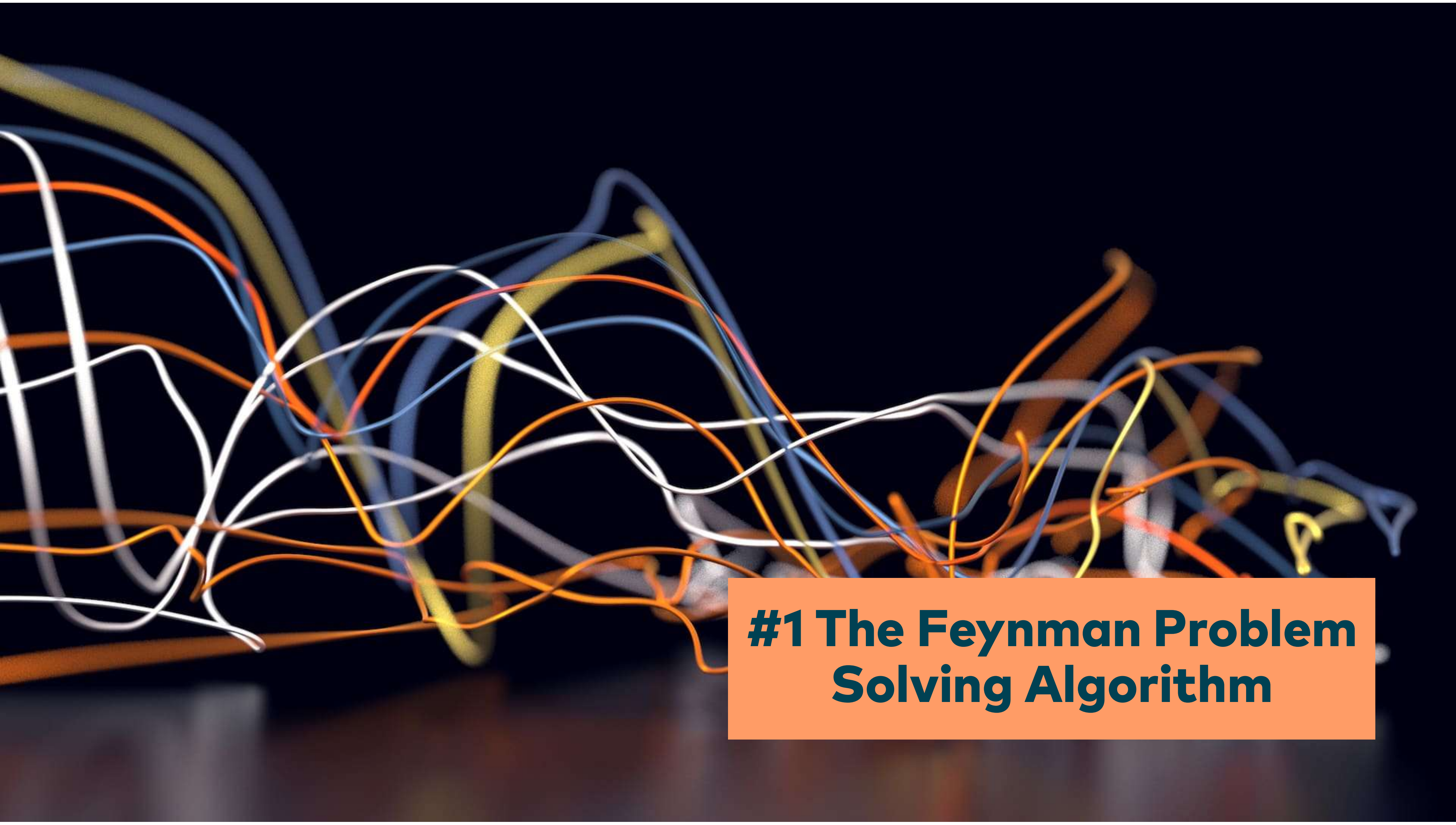
Typical Architecture Mistakes – The Third Is Shocking!

INNOQ

Eberhard Wolff
Fellow @ewolff

Why?

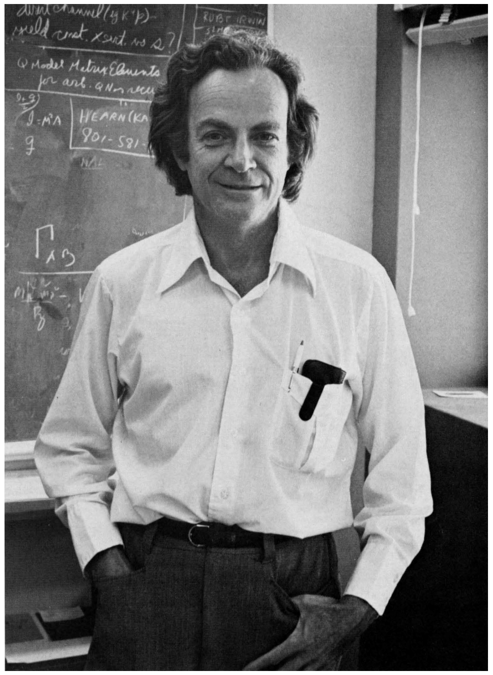
- Software architecture is important
- This is (just) my experience.
- Enjoy!



#1 The Feynman Problem Solving Algorithm

Richard Feynman

- Theoretical physicist
- Manhattan project
- Noble prize physics 1965
- Commission to investigate Space Shuttle disaster 1986
- Author of many great book



The Feynman Algorithm

1. Write down the problem

2. Think real hard

3. Write down the solution

https://wiki.c2.com/?FeynmanAlgorithm

Applying to Architecture

- We always end up with some chaotic architecture
- Not this time!
- This time we think real hard!

Applying to Architecture

- So you want to think hard.

- I.e. you have been sloppy the last time?

- Really?

The World Keeps Changing

- New requirements
- New technologies
- New quality / non-functional requirements

The World Keeps Changing

- Can't be predicted
- At best, you can get the best architecture for what you know.
- Seems hard to embrace

Repeat After Me...

- I have limited information.
- I can't come up with a perfect solution.
- I can only come up with good enough for now.

Applying to Architecture

Investing lots of time in a solution might make you defend it

…even if it's really no good solution

any more

Solution

- Don't fall for the Feynman algorithm!
- Learn to accept good enough!
- However, don't try to be stupid.
- You ain't gonna need it (YAGNI) is not always true…

Solution: Improvement Process

- Establish a process to improve the architecture!
- Schedule reviews?
- Architecture violation or contradiction: hint for an improvement?

Look at Science

- Contradiction to a scientific theory is a hint to build a new one.

- E.g. classical physics
  vs. quantum physic
  & relativity

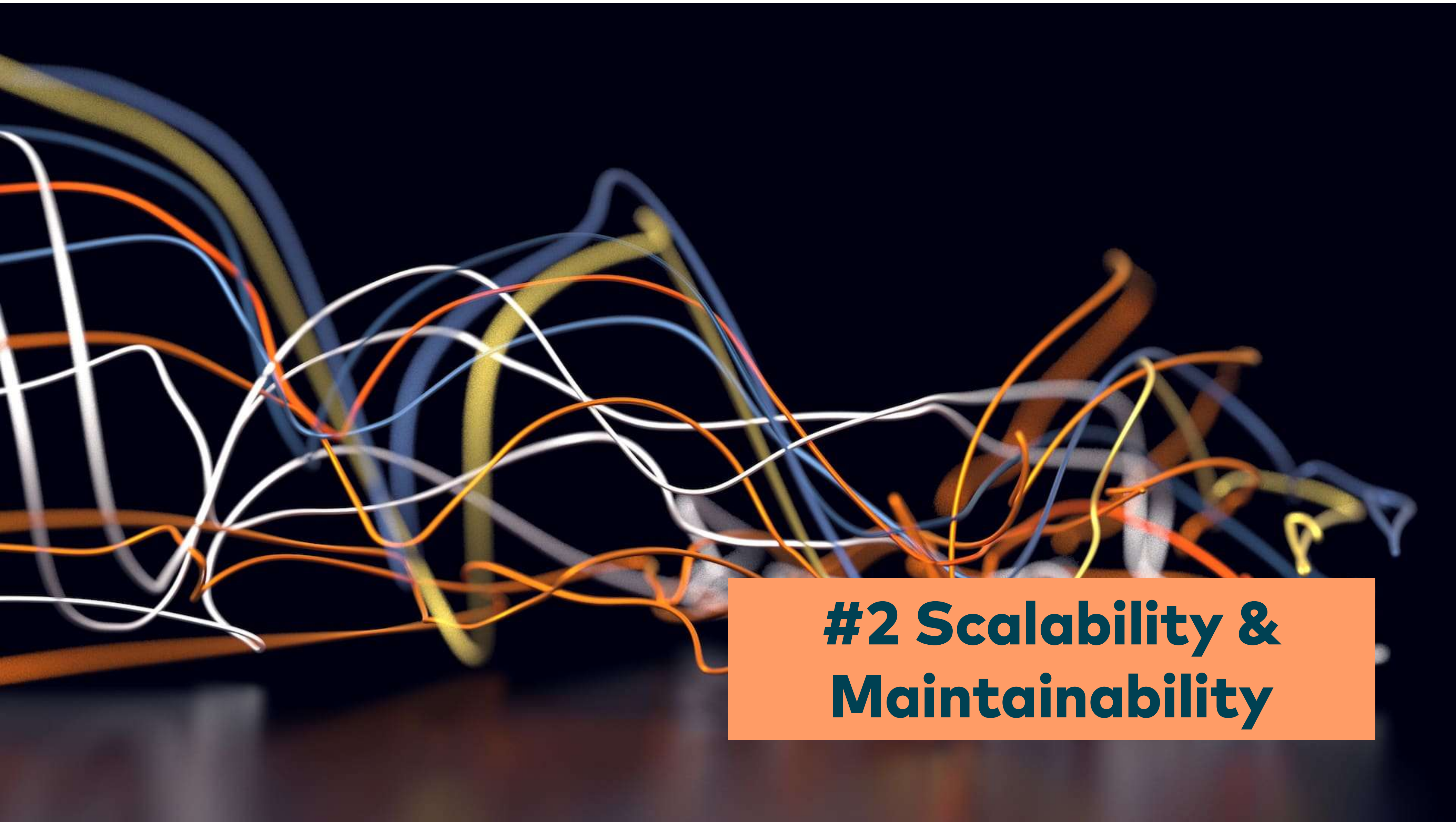
Solution: Stepwise

- Hard / expensive to build new features?

- What is an architecture adjustment to make it easier?

- Implementing a completely new architecture is expensive!

...probably impossible?



#2 Scalability & Maintainability

Scalability and Maintainability

- A well-architected system is scalable and maintainable.

- Seem to be default properties of a great architecture.

Contact Tracing App

- Build a contact tracing app for COVID19!

- Perfectly scalable & maintainable
- There will be lots of data!

Contact Tracing App

- Chaos Computer Club: "Doesn't fit any of our 10 criteria for contract tracing!"
- Hard to actually trace contacts
- This is actually about life & death.
- Good architecture?

- 🤷 Might still make you millions

...and give you access to valuable data.

We are using CQRS

Why?

Scalability

What is your main challenge?

Get users to use the application.

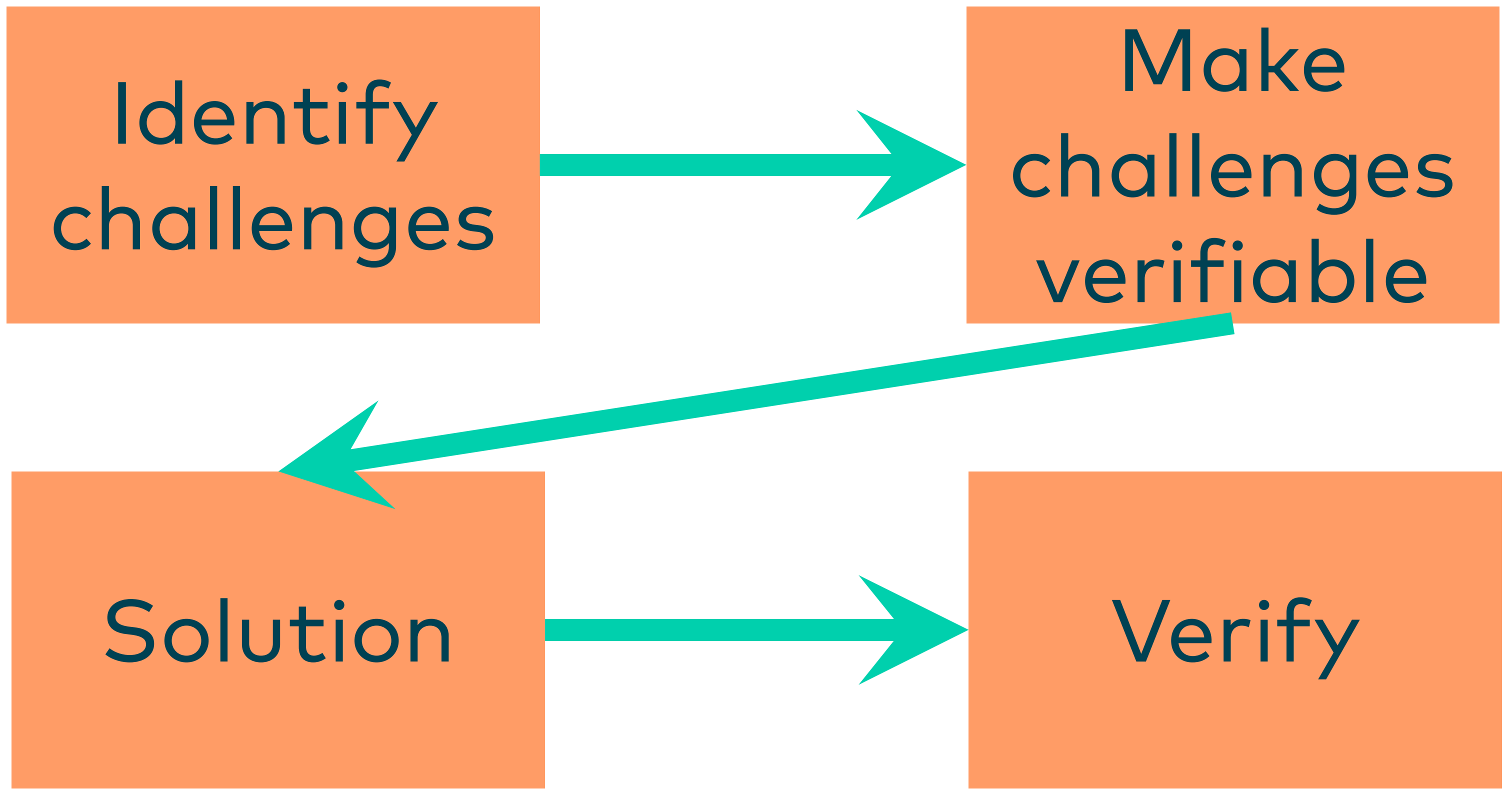
How do ensure that?

🤷

I.e. CQRS solves a problem (scalability)
...that you won't have
...unless you solve the main challenge (user acceptance)
...that you are unsure how to solve

#2 Lack of Technical Requirements and Solutions

The Essence of Architecture



The Essence of Architecture

Identify challenges

User acceptance

The Essence of Architecture

Make challenges verifiable

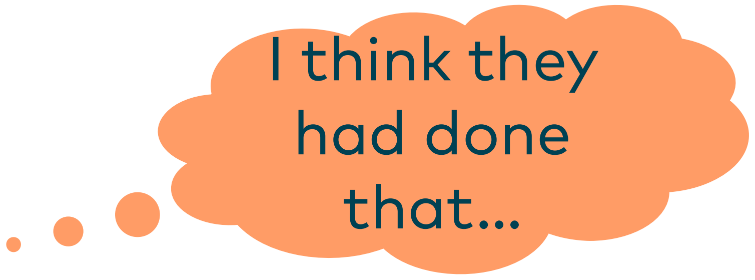
Users award the application 4.5/5 points

Over half of the users would recommend the application

The Essence of Architecture

Solution

Hire UX expert



Usability tests + evaluation by users

The Essence of Architecture

Verify

User reviews

Verification

- For performance:
  based on load scenarios

- For contract tracing app:
  e.g. based on CCC's criteria

My Experience

Identify challenges

Sometimes known

...but usually not in writing

My Experience

Make challenges verifiable

Usually non-existent

“10ms latency” probably too general

Break down per functionality?

My Experience

Solution

No clear relation to challenges

CQRS won't solve user acceptance

Visible in diagrams?

My Experience

Verify

Users / customer will verify whether your solutions fits

...but there might be unpleasant surprises.

Why not check the architecture of your current project?

Do you start the description of your architecture with a problem statement?

Do you start the description of your architecture with a problem statement?
Or rather some diagram of the structure?

Some Hints

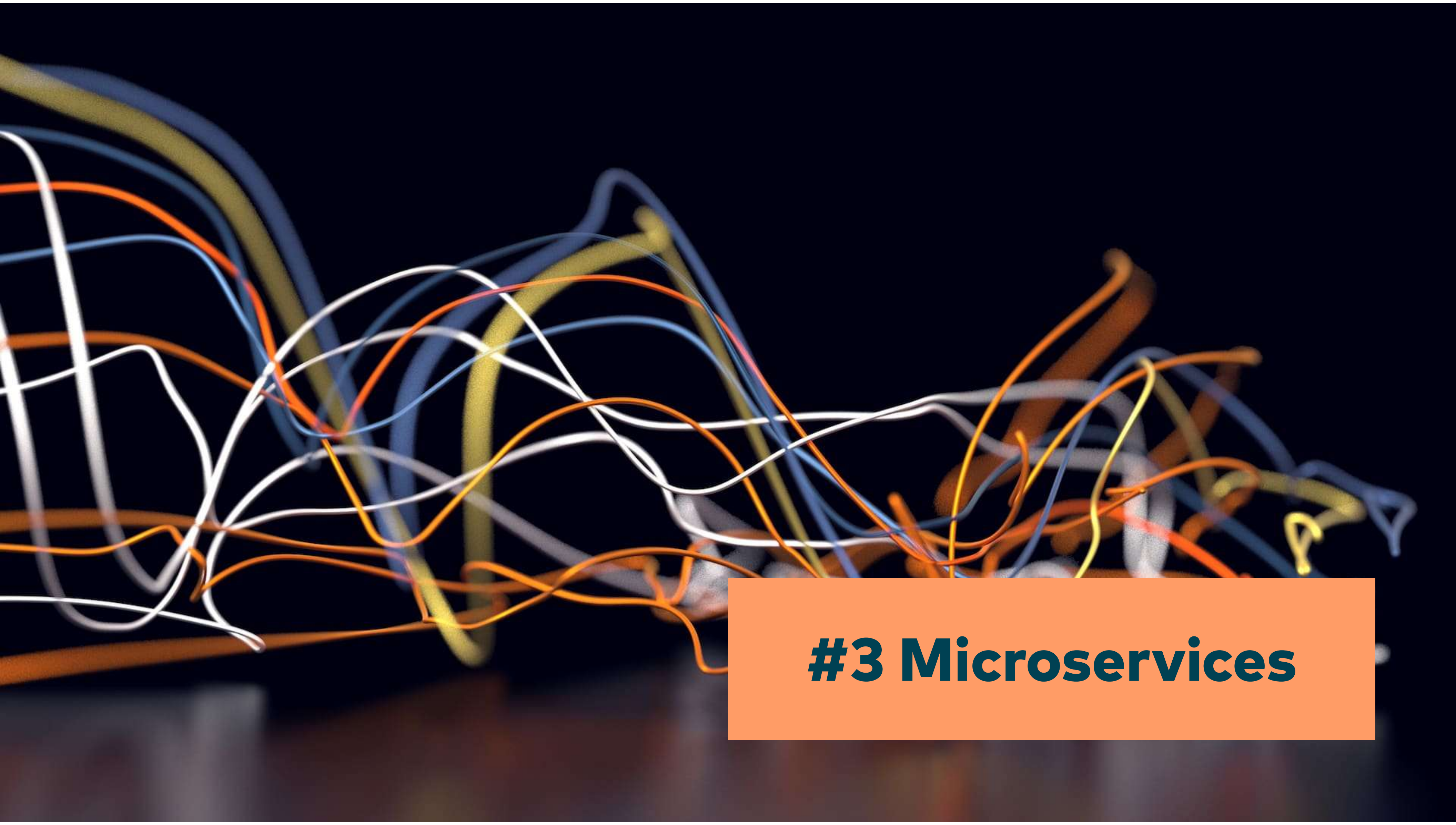
- Qualities and quality scenarios as defined e.g. by arc42

- ATAM Architecture Trade-Off Analysis Method

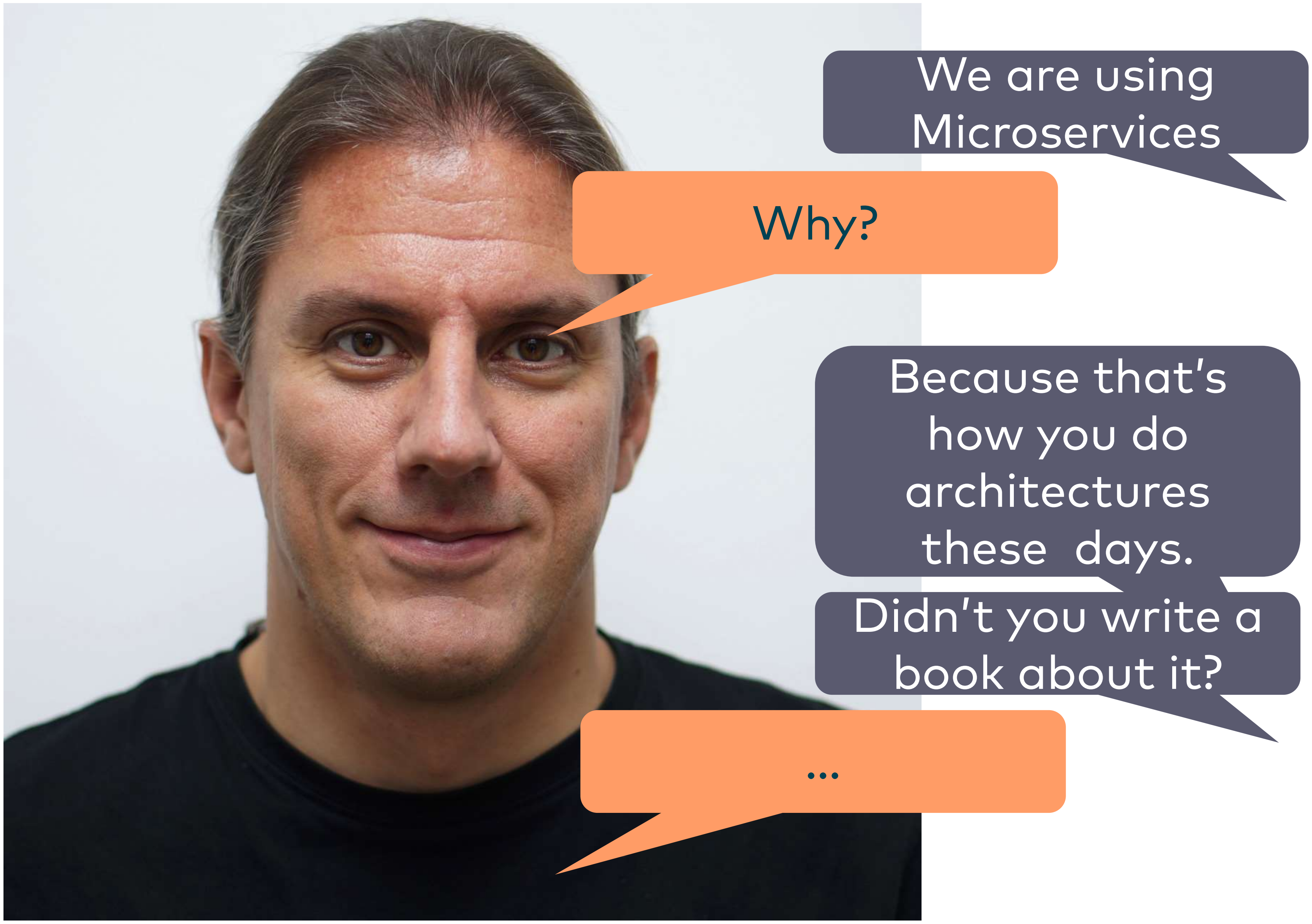
Validate architecture based on qualities and verifiable quality scenarios.

Three is a Magic Number

DE LA SOUL
BUDDY
THE MAGIC NUMBER
(FEAT. THE JUNGLE BROTHERS, MONIE LOVE, QUEEN LATIFAH & Q-TIP)



#3 Microservices


We are using Microservices
Why?
Because that's how you do architectures these days.
Didn't you write a book about it?
...

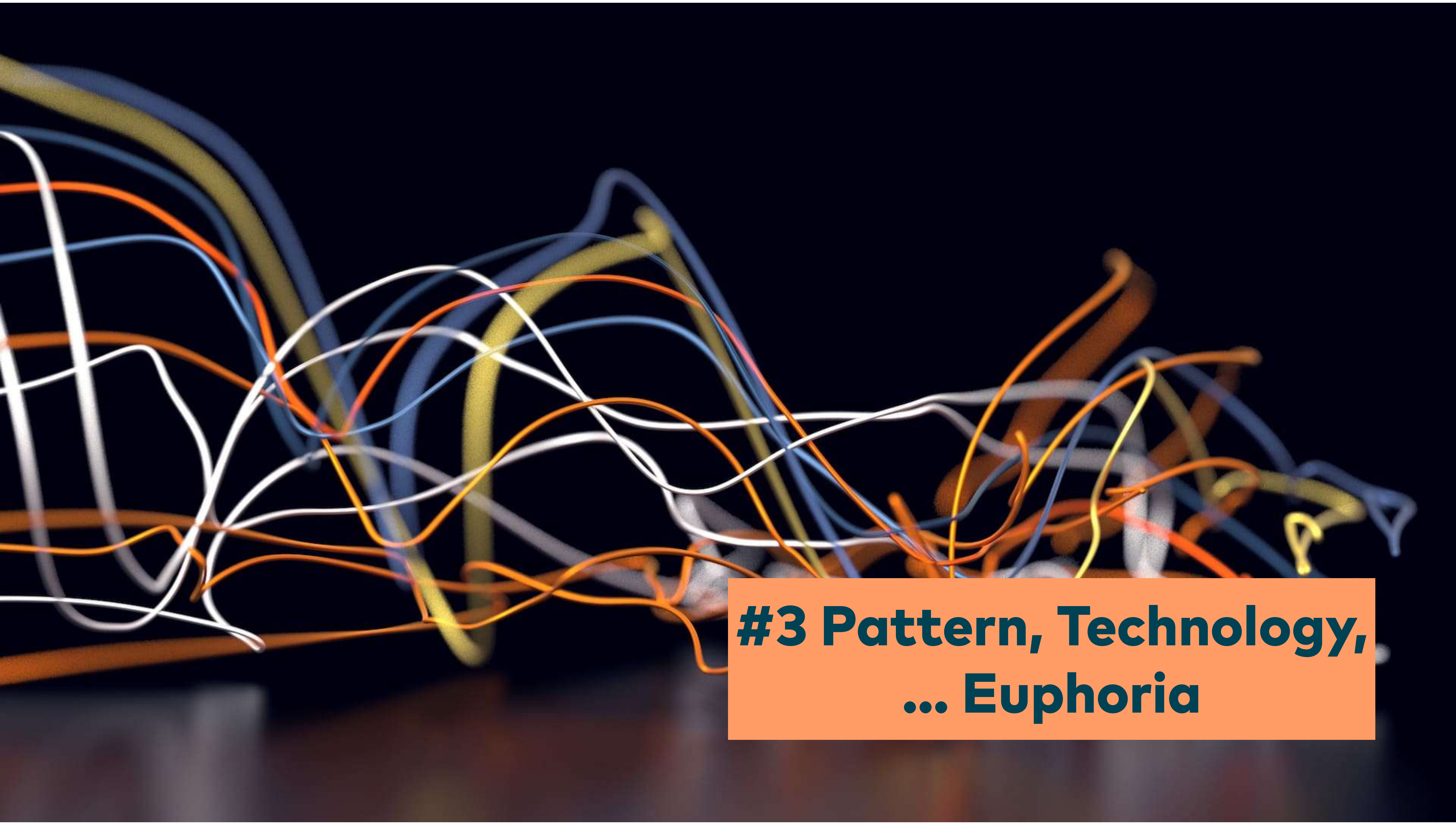
Just Microservices?

- Can happen with any

...architecture approach

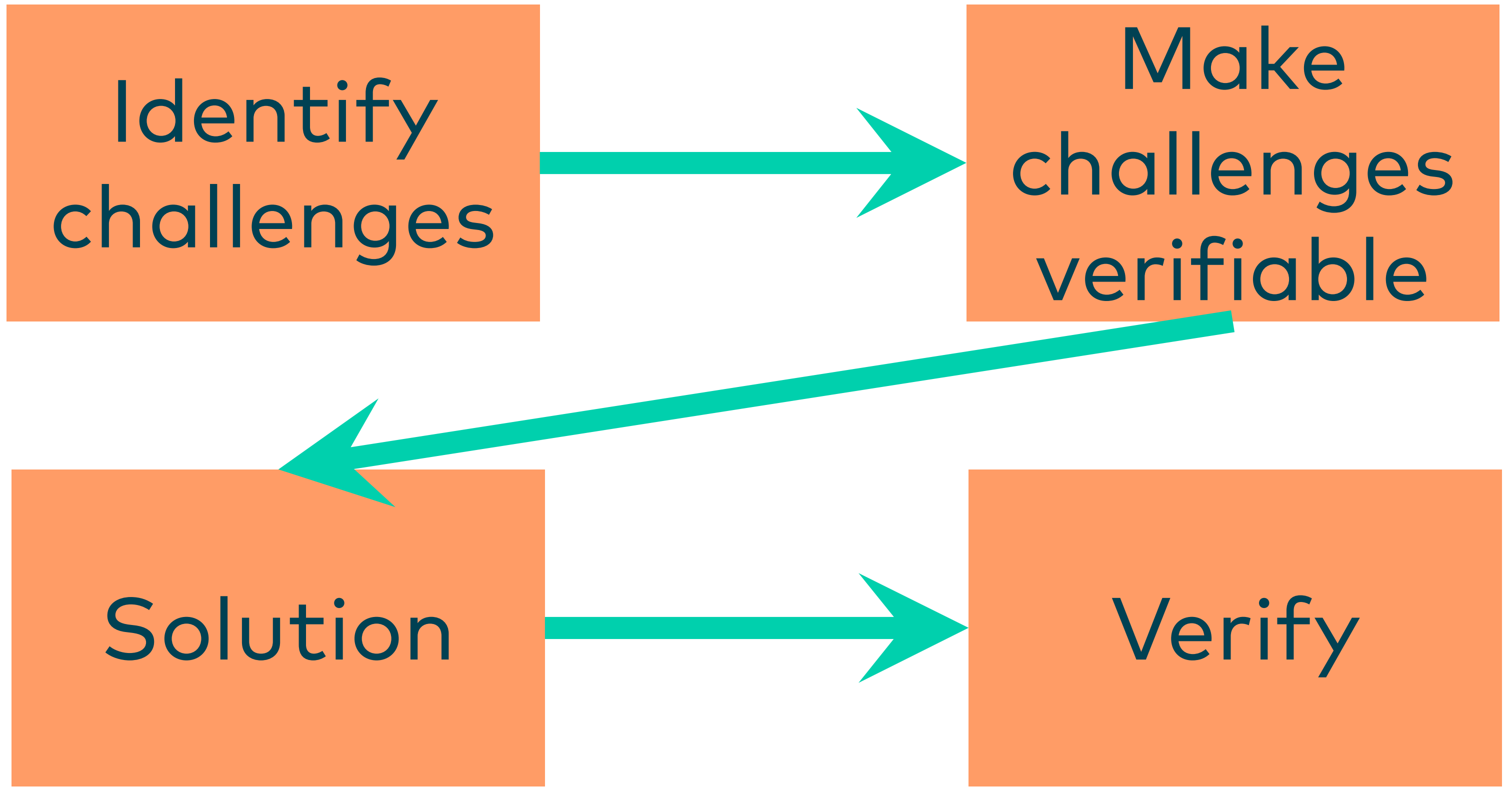
...technology

...and lots of other things

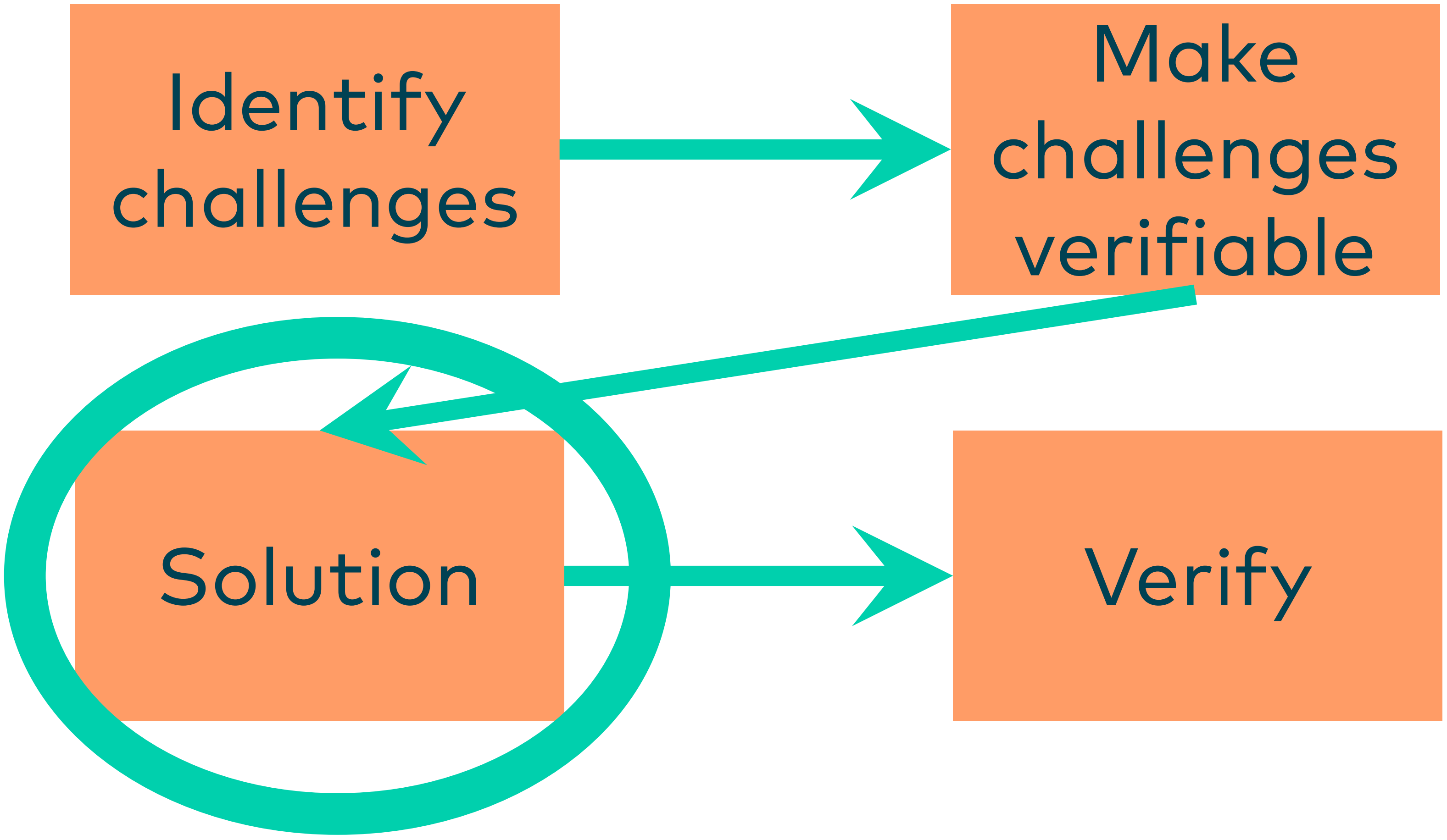


#3 Pattern, Technology, ... Euphoria

The Essence of Architecture



The Essence of Architecture



Solution

- Technologies / patterns etc are just another solution.
- I.e. they have to solve some problem.
- They are usually a trade-off.
- They solve some problem

...but cause others.

Microservices on One Slide

- Solves

Independent deployment

Independent technology choices

Independent scalability

- Cause

More effort for operations

Microservices on Two Slides

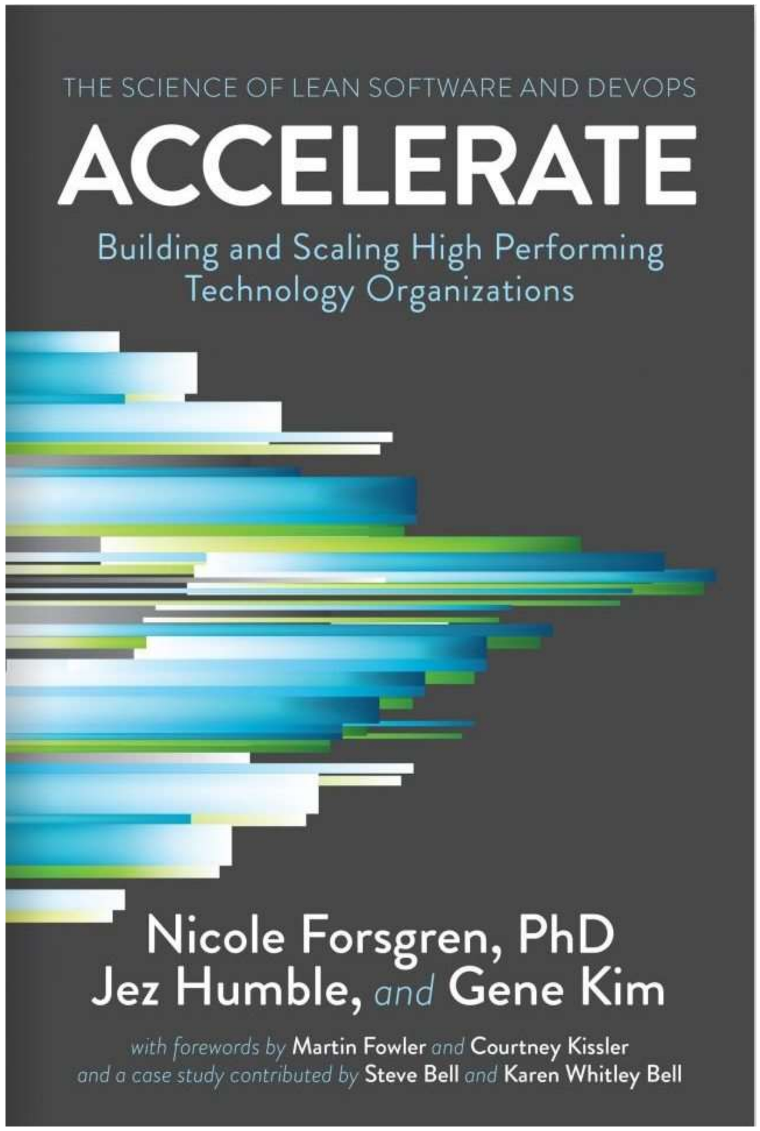
- Don't Solve

Clear and good cut into modules

Just a different implementation of modules

- Microservices is a talk in itself

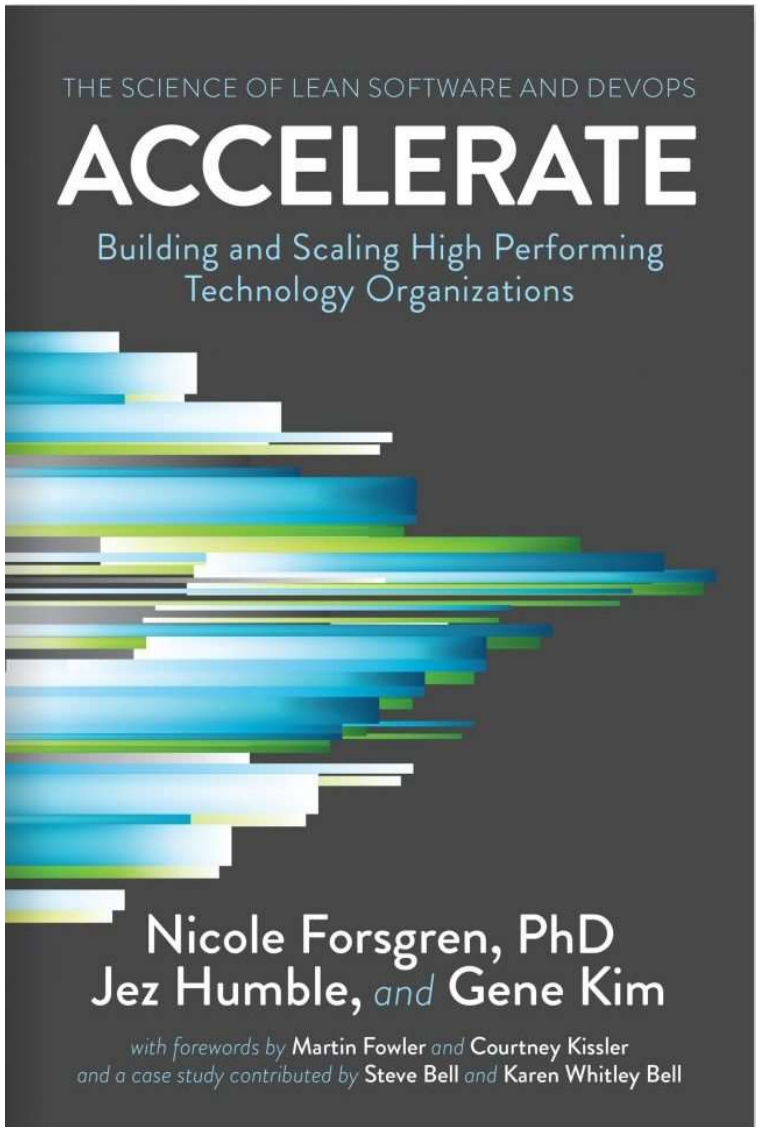
What is important
is **enabling teams**
to make changes to their
products or services
without depending on **other teams** or **systems**.


THE SCIENCE OF LEAN SOFTWARE AND DEVOPS
ACCELERATE
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim
with forewords by Martin Fowler and Courtney Kissler
and a case study contributed by Steve Bell and Karen Whitley Bell

Discussion … often focus on tools & technologies.

- Should the organization adopt **microservices**?
- Serverless?
- Kubernetes?
- Mesos?

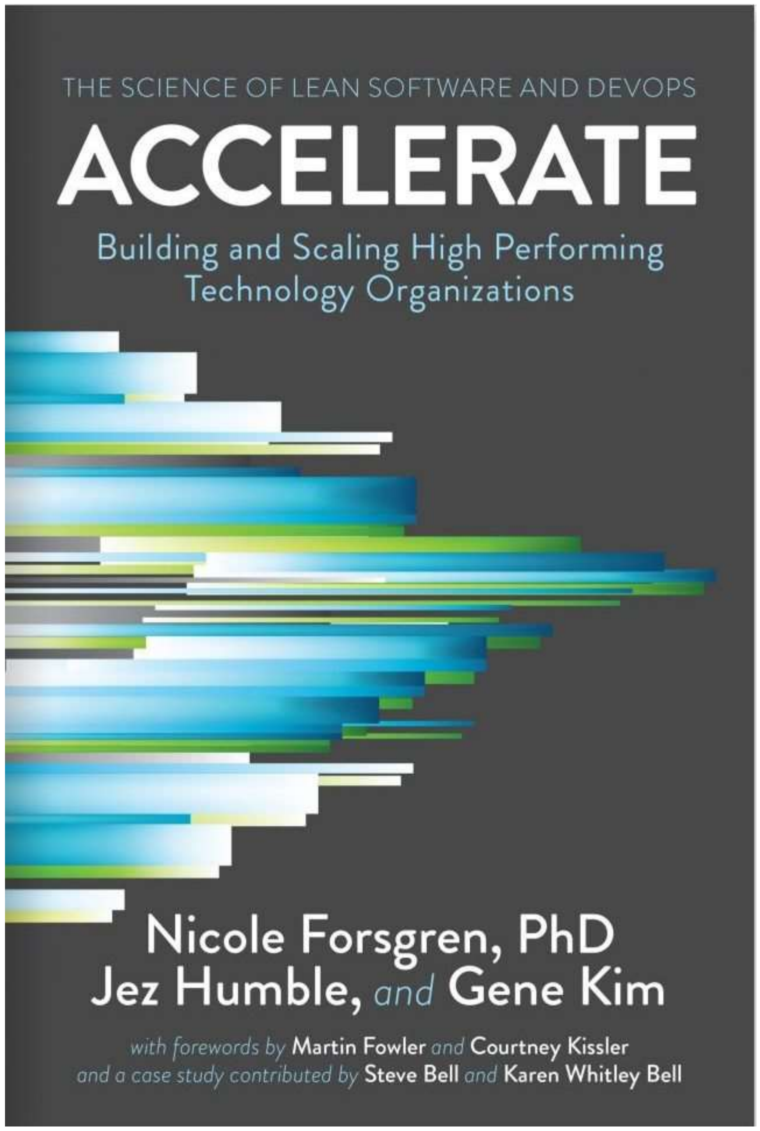
Our research shows these are the **wrong questions** to focus on.


THE SCIENCE OF LEAN SOFTWARE AND DEVOPS
ACCELERATE
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim
with forewords by Martin Fowler and Courtney Kissler
and a case study contributed by Steve Bell and Karen Whitley Bell

… **tools** [are] … **irrelevant** … if … people hate them …

or … they don't achieve the outcomes

and [don't] enable the behaviors we care about.



THE SCIENCE OF LEAN SOFTWARE AND DEVOPS

ACCELERATE

Building and Scaling High Performing Technology Organizations

Nicole Forsgren, PhD
Jez Humble, *and* Gene Kim

with forewords by Martin Fowler *and* Courtney Kissler
and a case study contributed by Steve Bell *and* Karen Whitley Bell

Why This is Hard

- We are technical people
- So we like technology
- Hard to limit yourself to what is really needed

Why This is Hard

- No clear approach towards new techniques and technologies
- “That’s just a hype!”
- But also: “Nowadays, you do that differently!”

Why This is Hard

- We don't like to be stuck with old technologies.

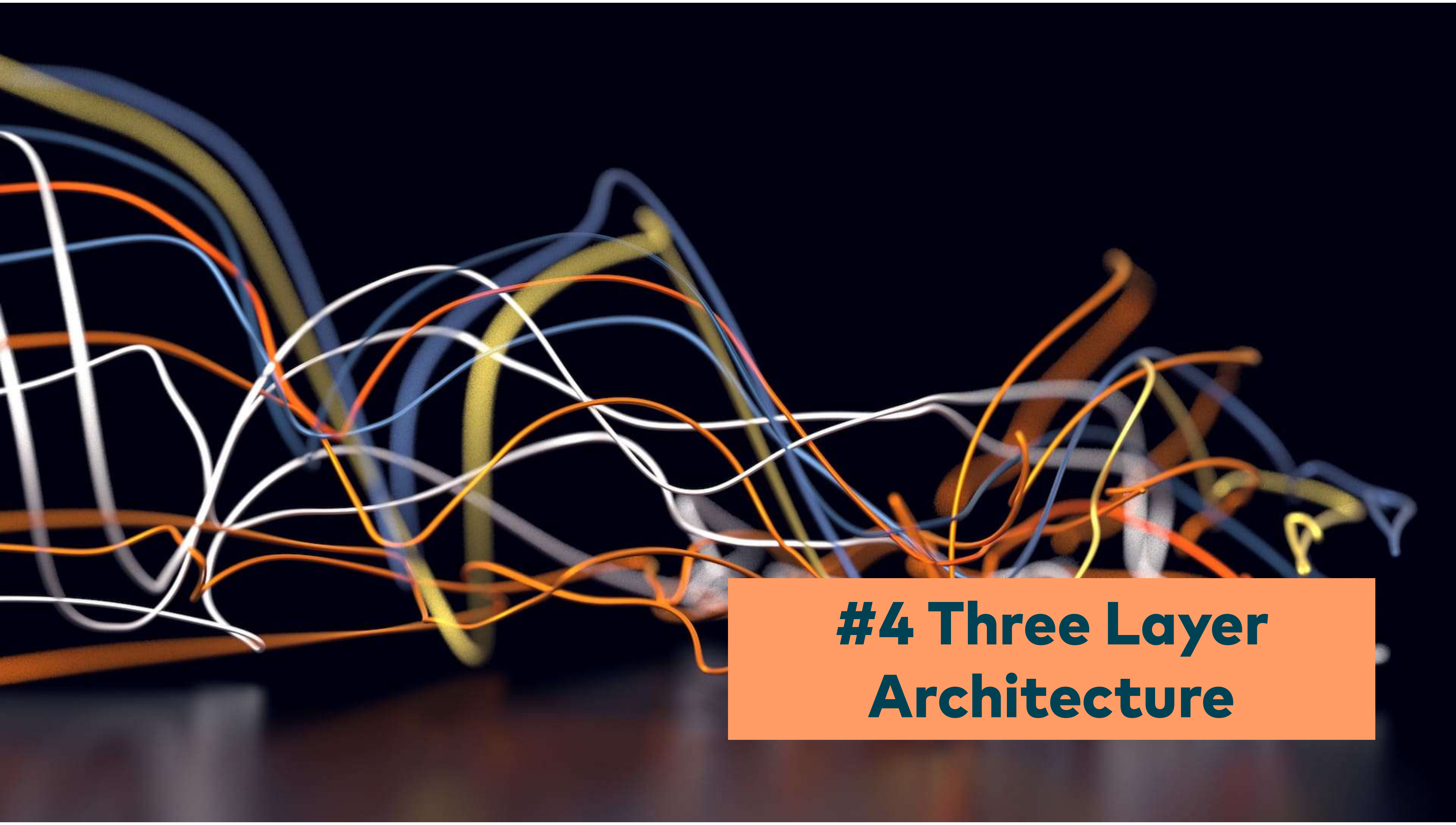
- And there is technological progress.

- But conservative is often less risk

How much technology do you really need for your project?

When do you need the (new) technologies?

Let’s build a monolith as a start?



#4 Three Layer Architecture

Architecture

UI

Logic

Persistence

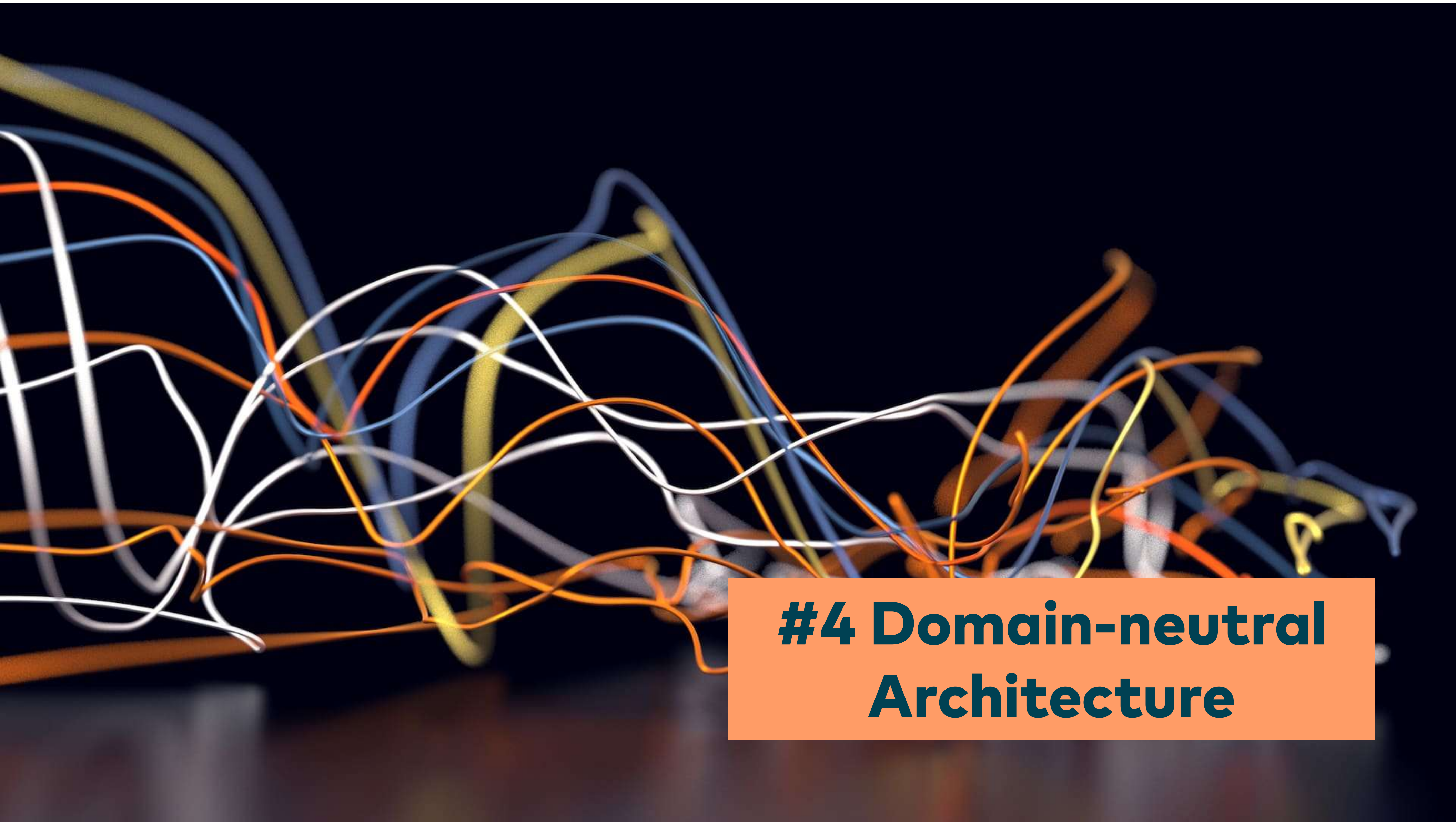
Can I take this architecture...

...and use it to build something different?

A videogame?

How does it solve their specific challenge?

Here is the architecture!
Your products are highly customizable, right?
Yip
How does it support customization?
10 minutes about stuff not in the diagram



#4 Domain-neutral Architecture

Architecture

- Should solve a domain problem
- Architecture should define how to split the domain.
- Smell: Architecture does not take domain into account.

Solution

- Domain-driven Design

…as we all probably know.

- Is it really that simple?

Reasons

- Architects not interested in domain
- Architects = technical people
- But: There is so much to tinker around with in a domain!

Reasons

- Unclear domain requirements
- Communication problem PO / architects

- Architects / developers will come up with

...well, *something*.

...that won't take the domain into account.

Worst Case

- Unclear domain requirements

- So build something that can cover everything that might ever come up.

- “Generic solution”
- But really: A mess

Worst Case

- Unclear requirements lead to failure
- Tough problem

- Work with what is there

…and try to learn more

- Talk to end users
- Clearly linked to project success

#5

I'm an architect.
I have great web skills.
Web developers here have no clue.
Really?
So what about X1 to support them?
Won't work.
Repeat for X2-X10

Architect

- Developers & others do the actual work
- Architects are meant to support them.
- If you can't support them, you failed.

- Hard because it's a social issue.

Why You Fail

- Architecture = decisions about code
- If developers ignore the architecture, it is pointless.

Why You Fail

- Developers are experts

...at least on the existing code

...but usually also on technologies etc

- If developers don't provide feedback, you can't adjust the architecture.

Welcome to the Ivory Tower

- Too little control

i.e. decisions pointless

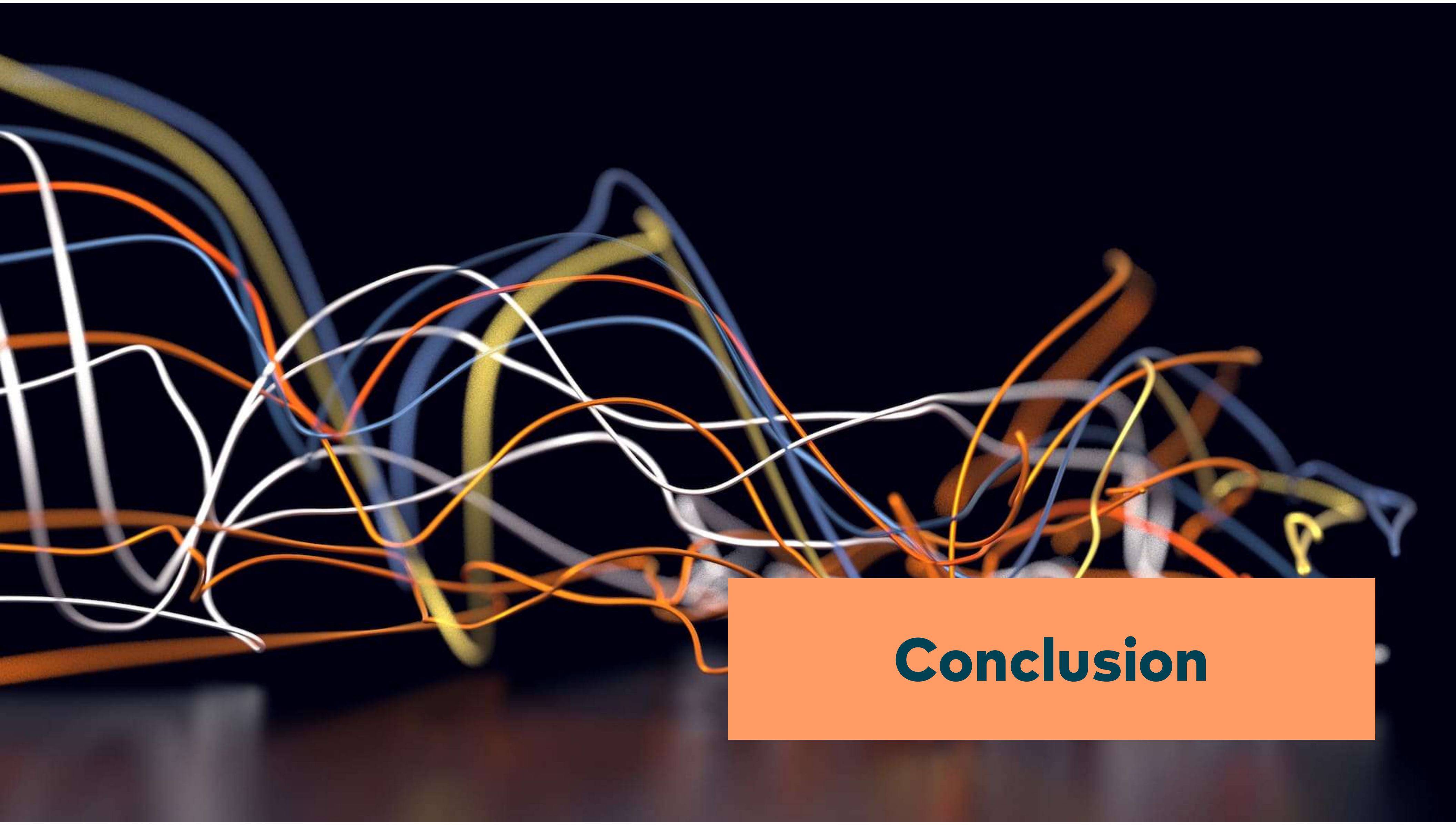
- Too little feedback

i.e. you don't know what's going on.

#5 Lonely Instead of Together

Architecture = Collaboration

- Talk to developers!
- Identify stakeholders!
- Talk to stakeholders!
- More moderation
- Less decision



Conclusion

Conclusion

1. The Feynman Algorithm
2. Lack of technical requirements & solutions
3. Euphoria (technologies, patterns)
4. Domain-neutral architecture
5. Lonely instead of Together

Send email to technologyday2021@ewolff.com
Slides

+ Service Mesh Primer EN
+ Microservices Primer DE / EN
+ Microservices Recipes DE / EN
+ Sample Microservices Book DE / EN
+ Sample Practical Microservices DE/EN
+ Sample of Continuous Delivery Book DE

Powered by Amazon Lambda
& Microservices

EMail address logged for 14 days,
wrong addressed emails handled manually