Software Architecture for Humans!

INNOQ

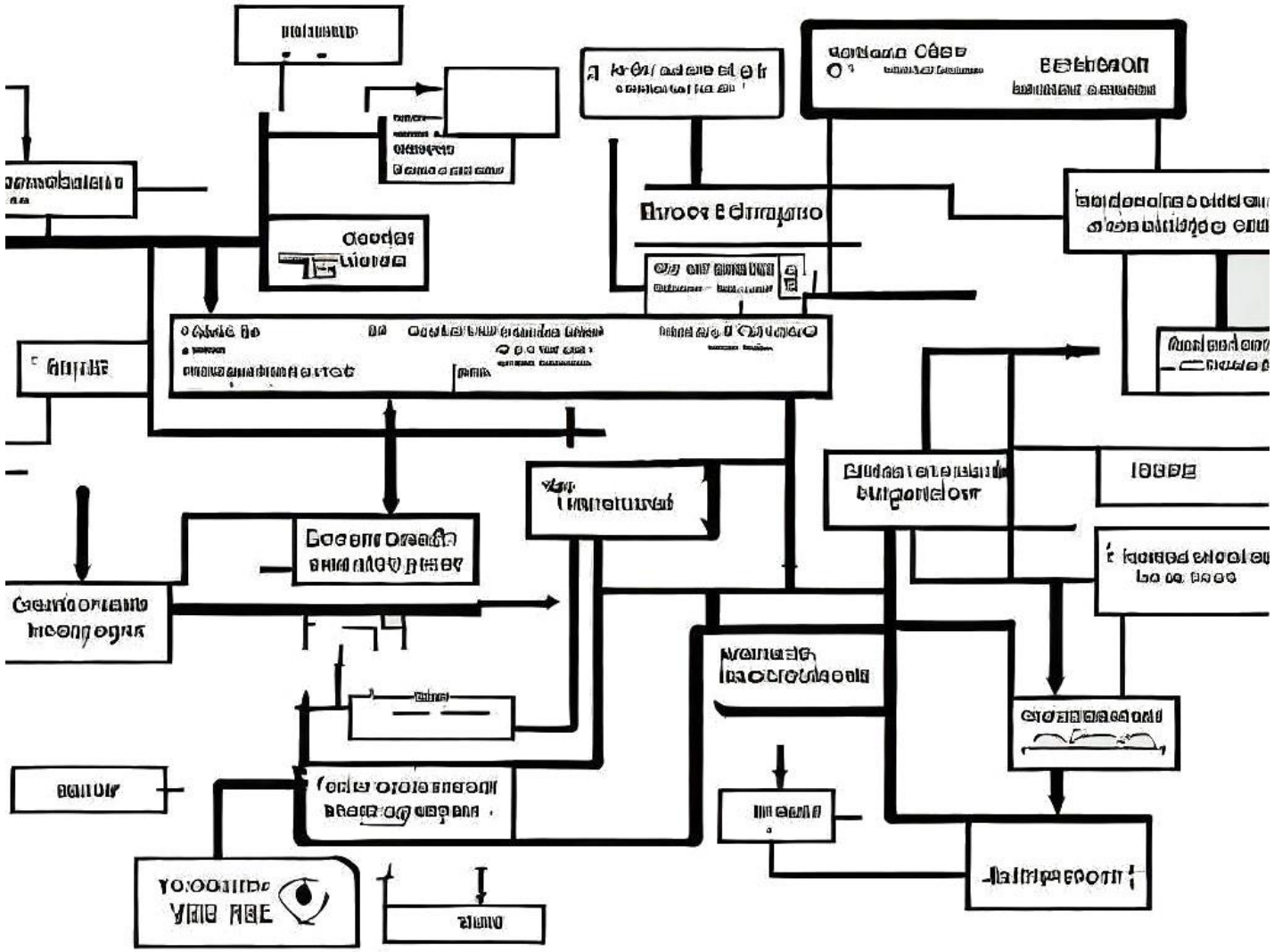
Eberhard Wolff
Fellow
@ewolff
https://mastodon.social/@ewolff

Architecture?

- This presentation:
  Architecture = structure
- Architecture goal: Maintainability

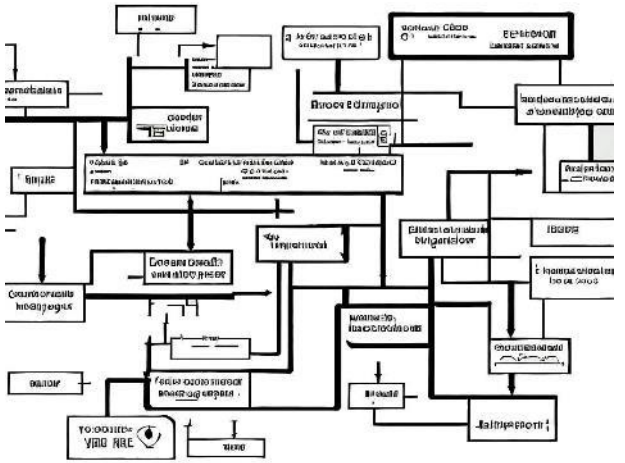
- Architecture should take all quality goals into account!
- …and tackle them!

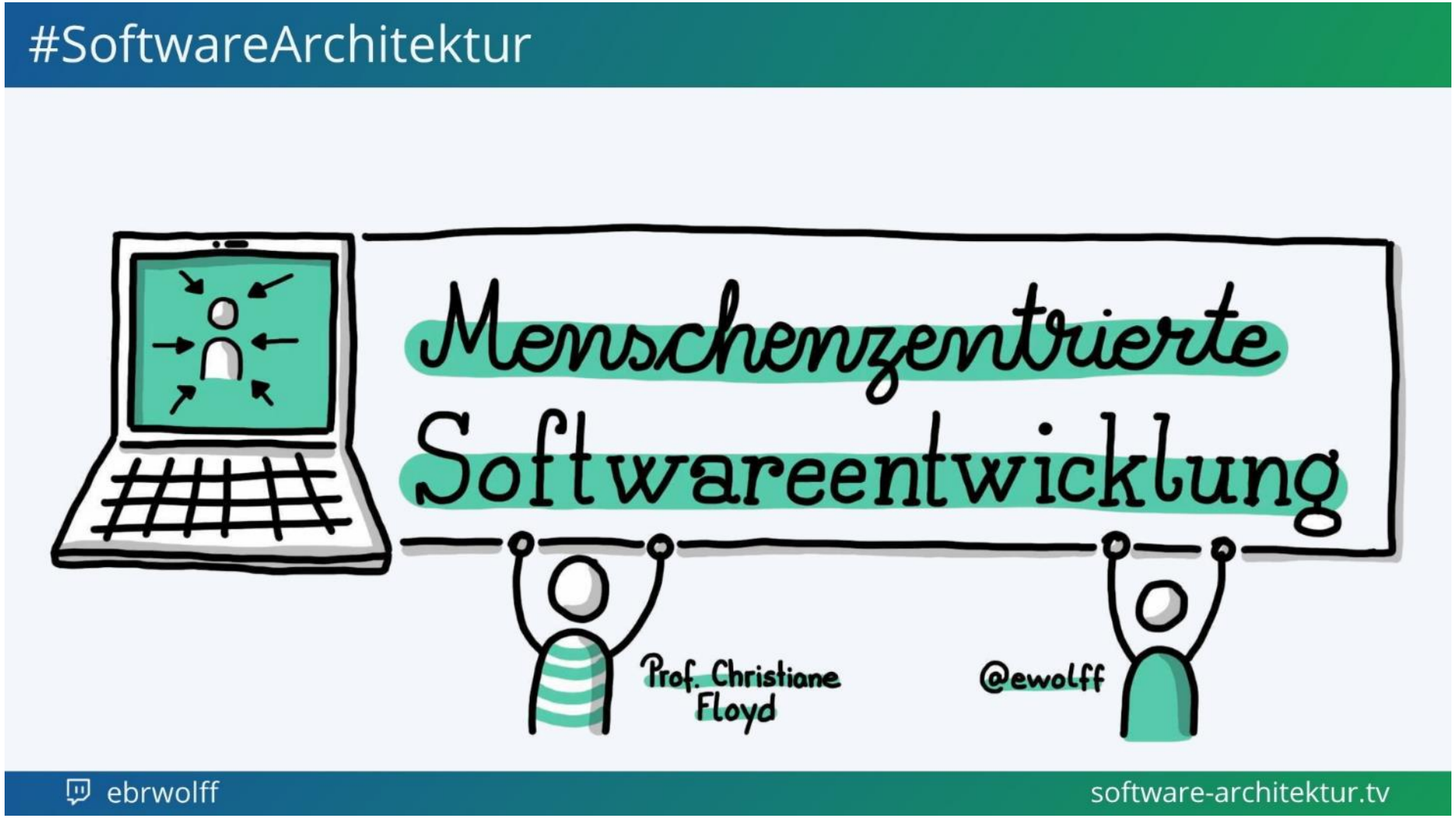
Is this a Great Architecture?



Why are we Doing Architecture?

- Human have limited mental capacity
- Humans must be able to modify the system
- Architecture should allow humans to change a system with limited knowledge





https://software-architektur.tv/2021/07/09/folge66.html

Was ist menschenzentrierte Softwareentwicklung?

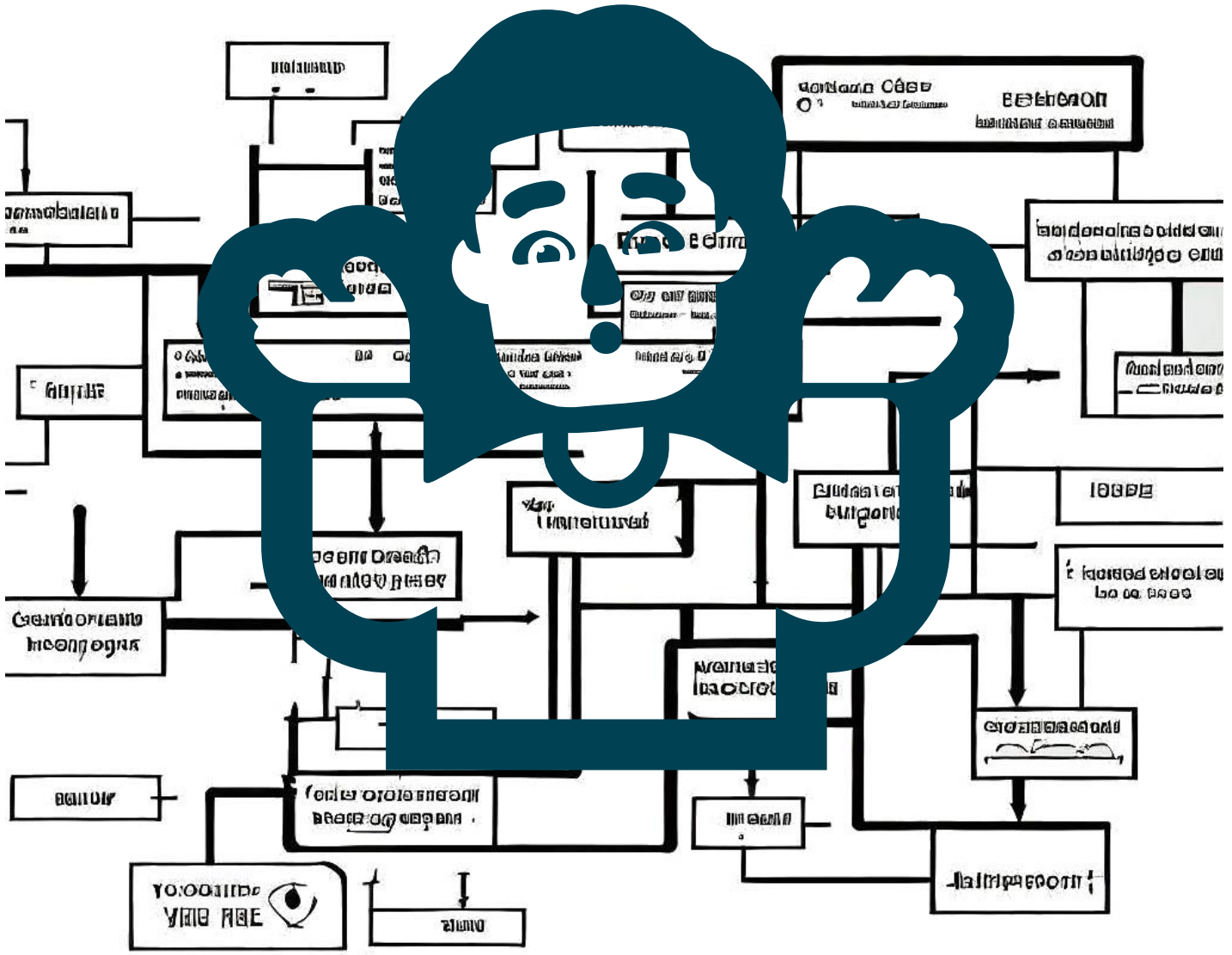
Entwicklungsprozess + Anwendung

↳ manchmal antizipieren, lieber zusammen mit Nutzern

Zusammenarbeit & Kommunikation

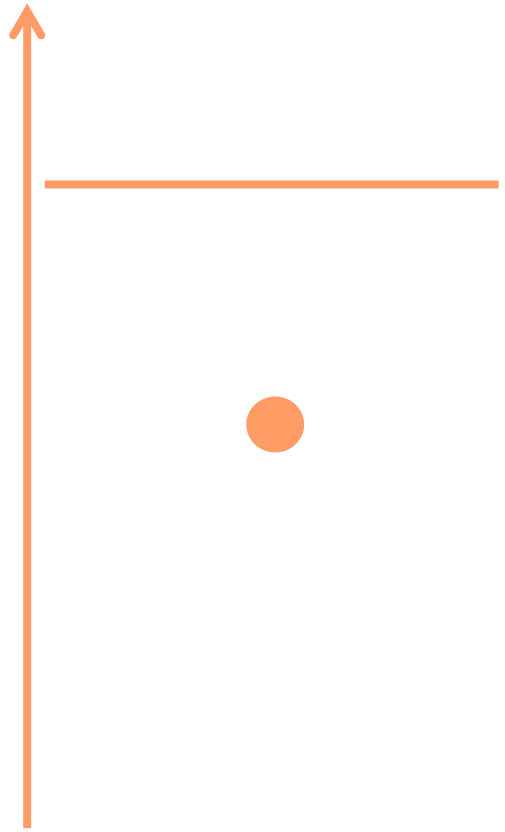
Modularisierung

Is this a Great Architecture?



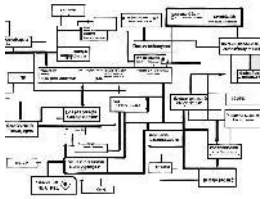
Is this a Great Architecture?

For whom?



Complexity

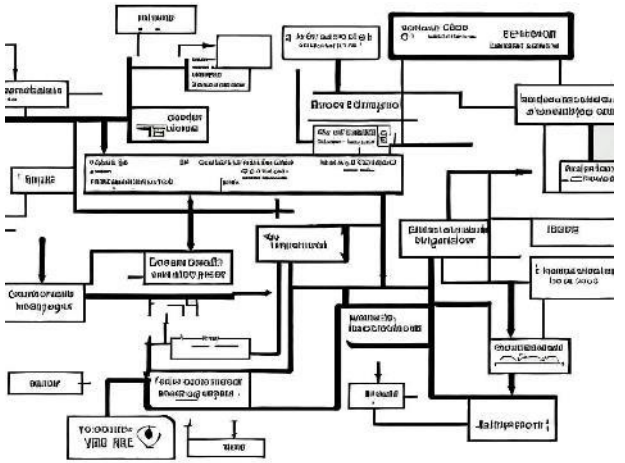
Maximum complexity team can handle efficiently



Actual complexity of the system

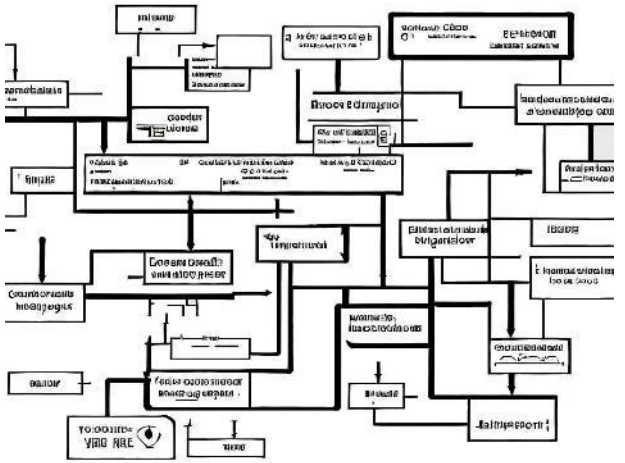
Is this a Great Architecture?

- Can only review architecture when considering the people, too.
- I.e. there is no "absolute great architecture"!
- Use metrics with care!



Is this a Great Architecture?

- Interviews: Where are the problems?
- Support findings by metrics
- Think about improvements

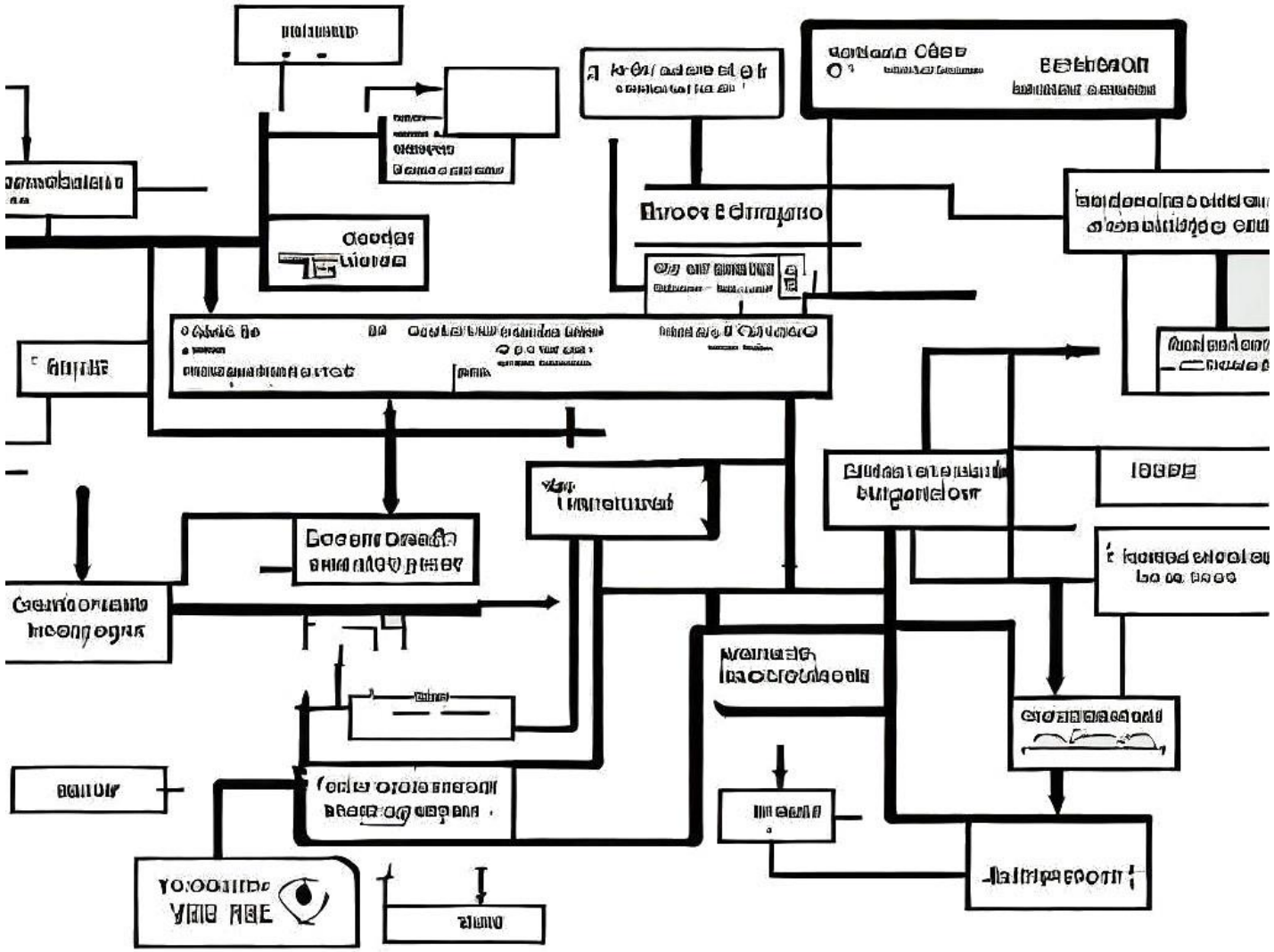


Consider Social Aspects

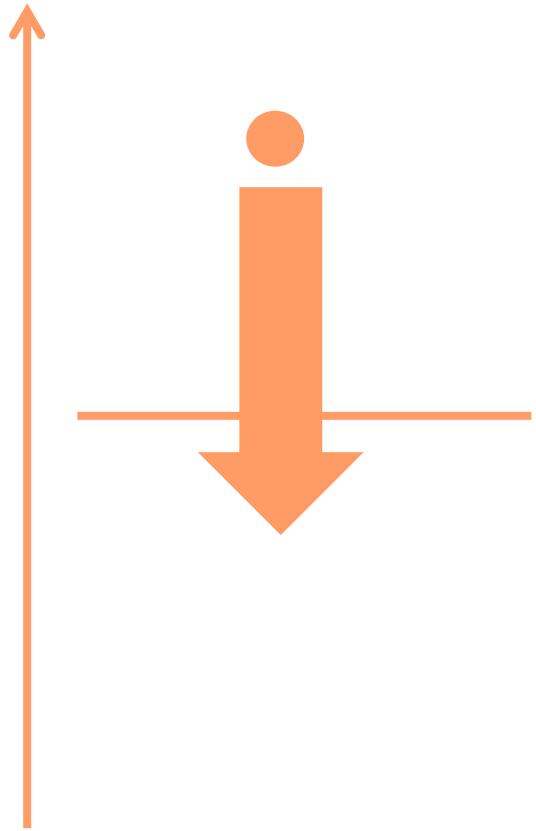
- Who changes what?
- What is changed frequently?
- What is changed seldomly?
- …

How Do You Improve an Architecture?

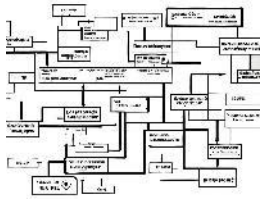
Obvious: Optimize Dependencies



Traditional Fix: Reduce Complexity



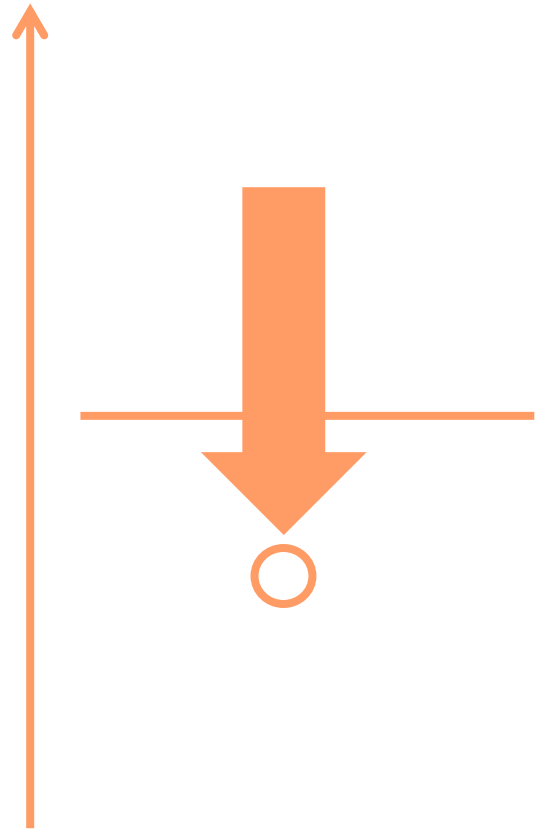
Complexity



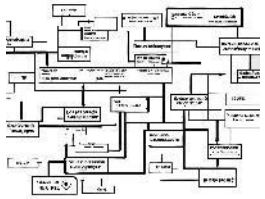
Actual complexity of the system

Maximum complexity team can handle efficiently

Traditional Fix: Reduce Complexity



Complexity



Actual complexity of the system

Maximum complexity team can handle efficiently

Broken?

- Team fine with one system
- Team: This other system is really bad!
- Metric: Other system is well-structured
- ...but it was handed over to the team.
- Team never really learned the system.

What if interviews show that an architecture with well-structure dependencies is really broken?

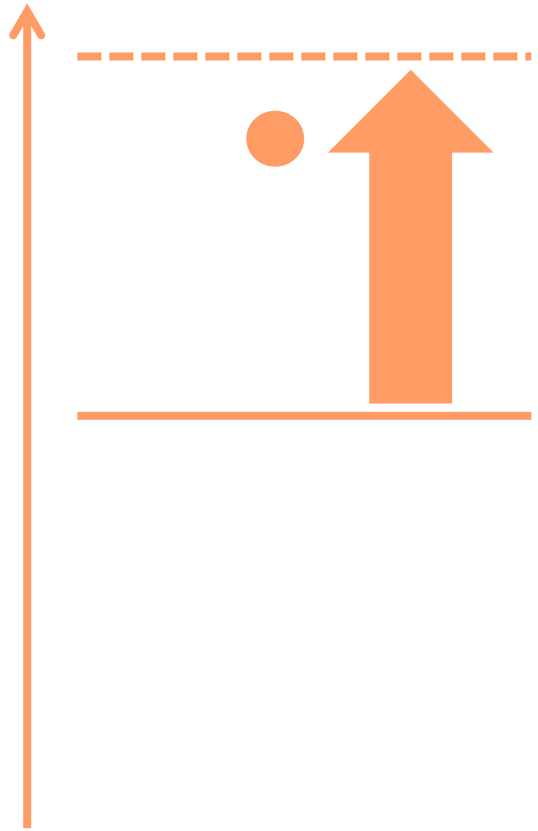
Broken but Well-Structured?

- Well-structure code is not enough
- Developers must understand the system.
- Ever tried to understand a system you developed a few years back? 😐

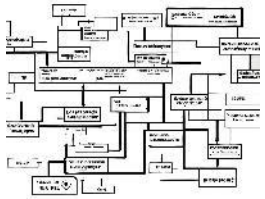
Improve People not Software

- Figure out why developers don't understand the system.
- Educate about the architecture!

Fix: Education



Complexity



Actual complexity of the system

Maximum complexity team can handle efficiently

Learn the System

Educate the team

Reading Code

- Code is read more frequently that written.
- Learn how to read code!
- Felienne Hermans researches this subject.

https://codereading.club/

https://software-architektur.tv/2021/10/13/epsiode81.html

Legacy: A Social Problem?

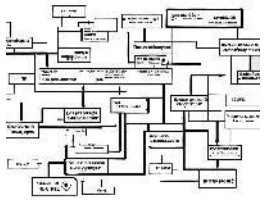
Legacy: Traditional Explanation

Software rot

Technical debt

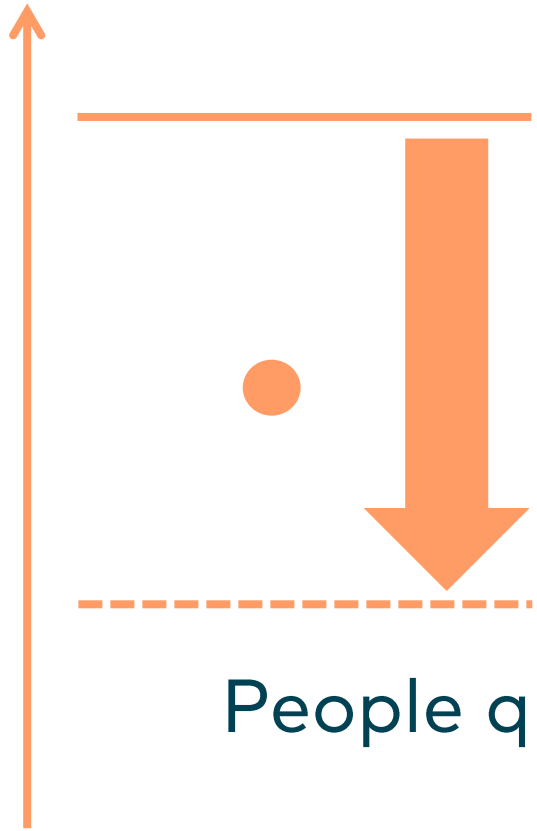
Complexity

Maximum complexity team can handle efficiently

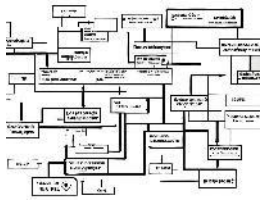


Actual complexity of the system

Legacy: Social Explanation

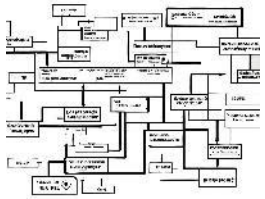


Maximum complexity team can handle efficiently



Actual complexity of the system

Legacy: A Social Problem



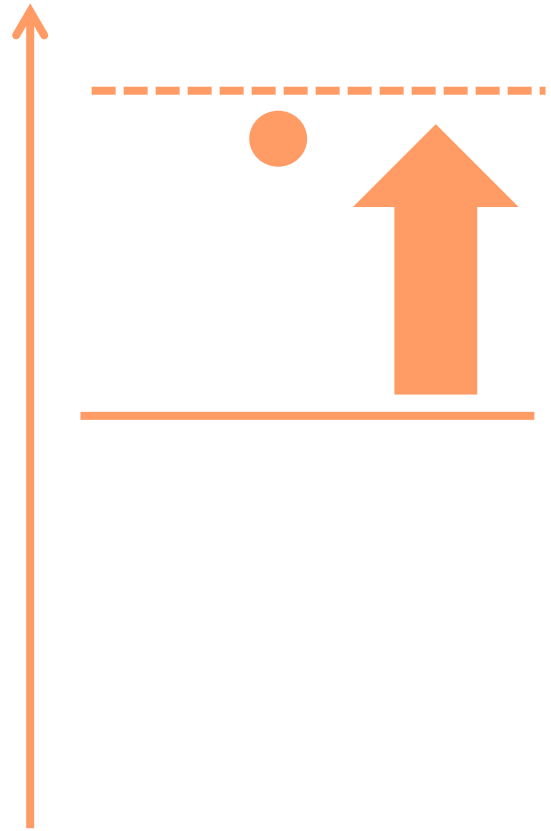
Actual complexity of the system

Maximum complexity team can handle efficiently

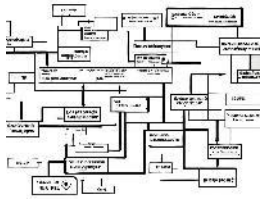
THESE people cannot handle the complexity of THIS system efficiently

Complexity

Fix: Education



Complexity



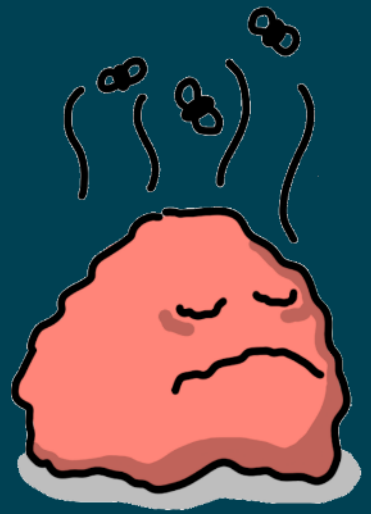
Actual complexity of the system

Maximum complexity team can handle efficiently

Learn the System

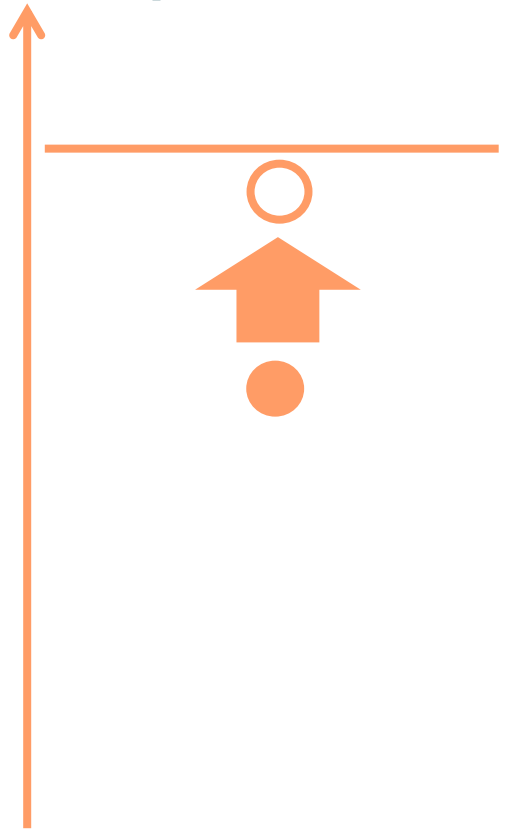
Educate the team

Big Ball of Mud



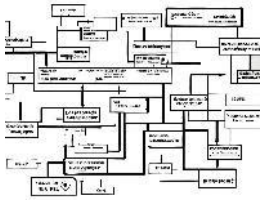
Icon: Lisa Moritz

Increasing Complexity: Fine?



Complexity

Maximum complexity team can handle efficiently



Actual complexity of the system

Still maintainable

Cheaper

some additional complexity is a given

Increasing Complexity: Fine?

- Must stay efficiently maintainable!
- Careful: Consequences of too low quality might be disastrous!
- But: There is no such thing as a perfect system.

Big Ball of Mud: Pattern

Icon: Lisa Moritz

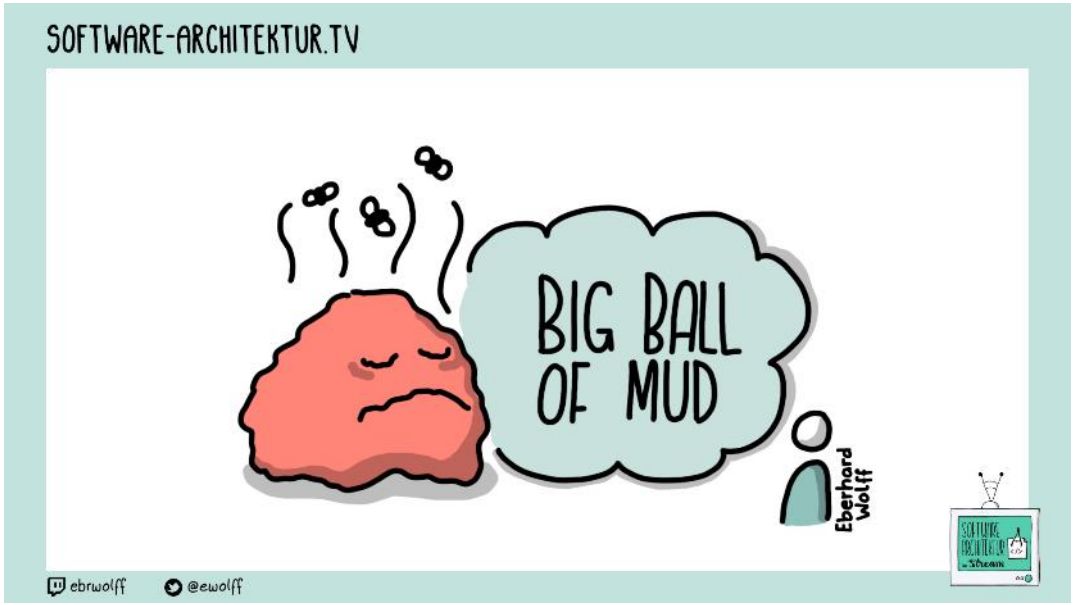
A Big Ball of Mud is **haphazardly structured**, sprawling, sloppy, duct-tape and bailing wire, spaghetti code jungle.

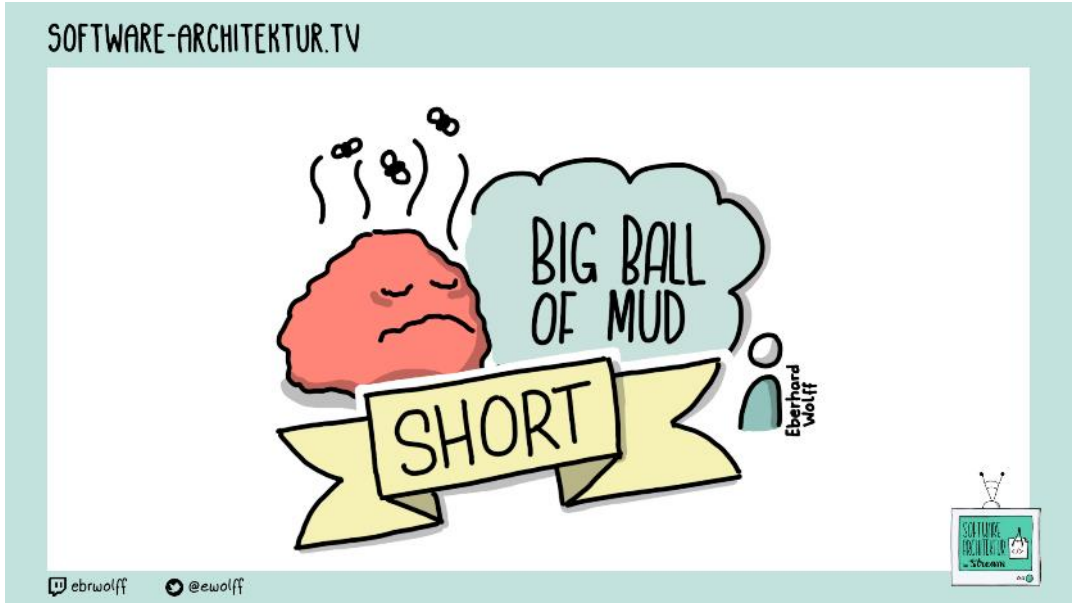
Why is this architecture so **popular**?

You need to deliver **quality** software **on time**, and **under budget**.

Therefore, focus first on **features** and **functionality**, then focus on **architecture** and **performance**.

Big Ball of Mud, Brian Foote & Joseph Yoder
http://www.laputan.org/mud/

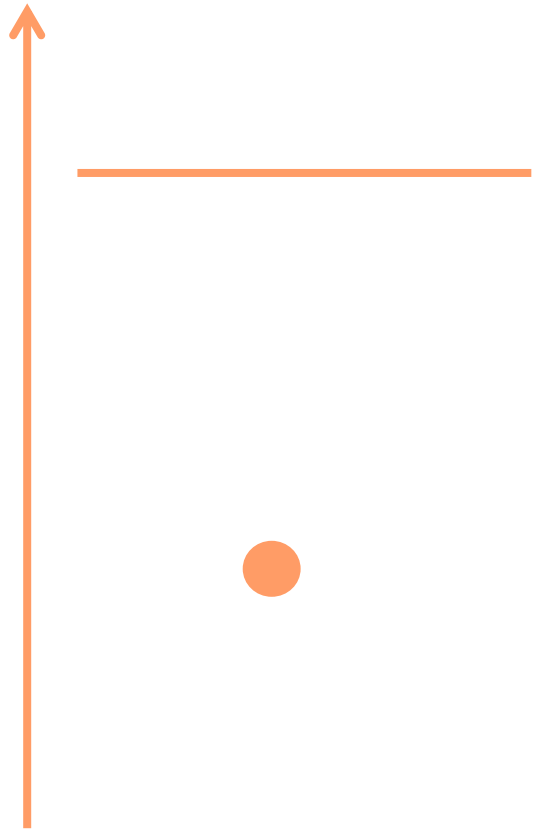




https://software-architektur.tv/2023/03/31/folge159.html

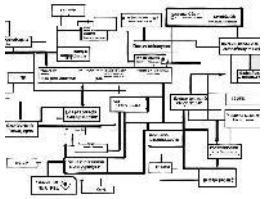
Would you like to be called a good developer?

Would you like to be praised for being a good developer?

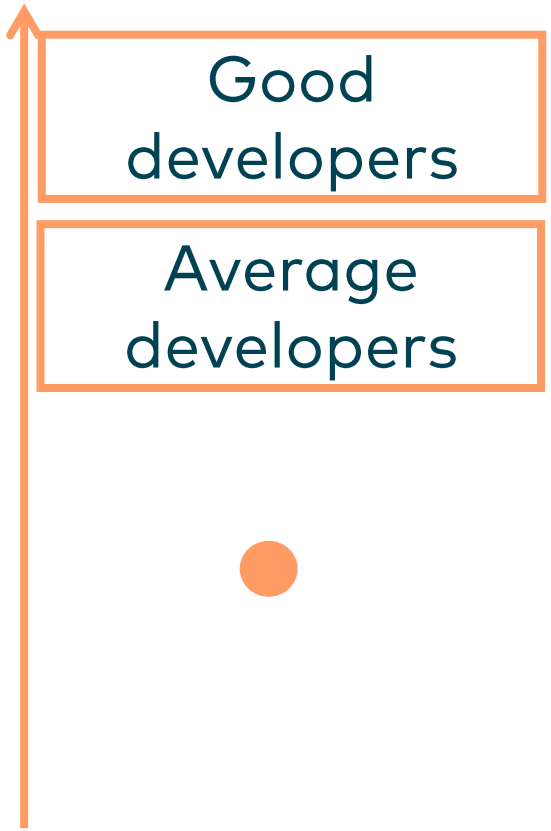


Complexity

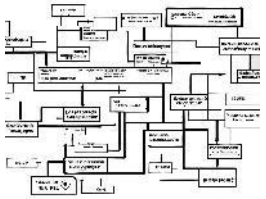
Maximum complexity team can handle efficiently



Actual complexity of the system



Maximum complexity team can handle efficiently

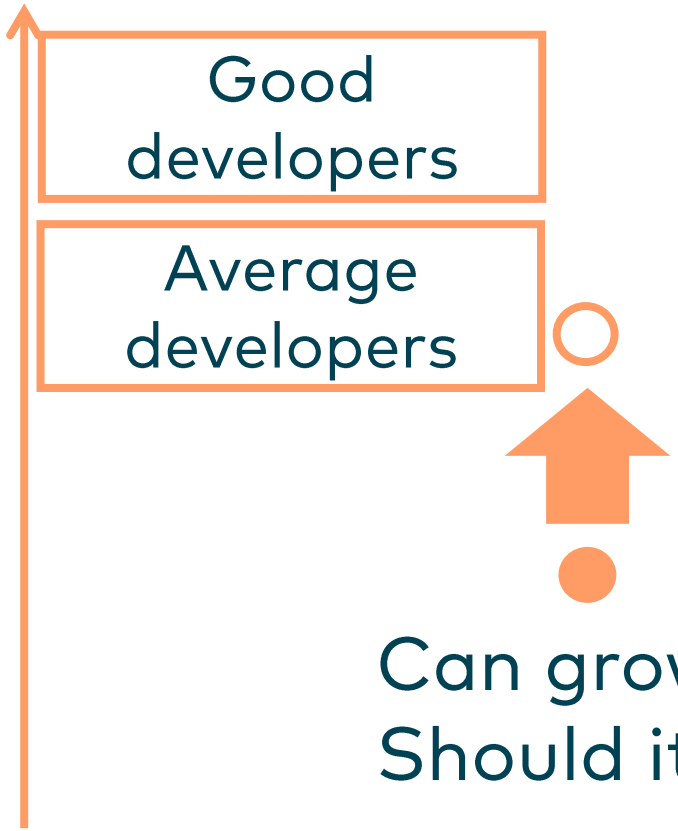


Actual complexity of the system

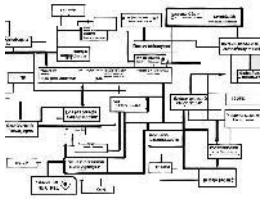
Vs. Good Architecture

Icon: Lisa Moritz

- Good architecture: changeable
- Big Ball of Mud: Not really changeable
- Every architecture has weak spots.
- How many weak spots are acceptable?



Maximum complexity team can handle efficiently



Actual complexity of the system

Complexity

Can growing complexity be avoided? Should it?

Practical solution vs “theoretical”

“Clean” is really hard and requires lots of effort.

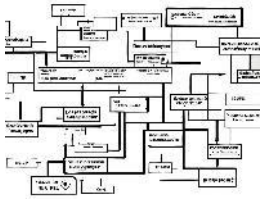
You saved the day!

You are great developers!

Good developers

Average developers

Maximum complexity team can handle efficiently



Actual complexity of the system

Complexity

More complexity = more praise for good developers?

and job security?

and interesting challenges?

DE https://youtu.be/p7r6IE7TkpU
EN https://youtu.be/3MP-4UcAYJU

Those are not good developers!

Those are not good developers!
I would love to agree!

Java Certification

I'd rather not work in a project that requires understanding such code or where people write such code.

So why would we ask for such knowledge?

```
public class Client {
   static void doCalc(byte... a)  {
      System.out.print("byte...");
   }
   static void doCalc(long a, long b) {
      System.out.print("long, long");
   }
   static void doCalc(Byte s1, Byte s2) {
      System.out.print("Byte, Byte");
   }
   public static void main (String[] args) {
      byte b = 5;
      doCalc(b, b);
   }
}
```

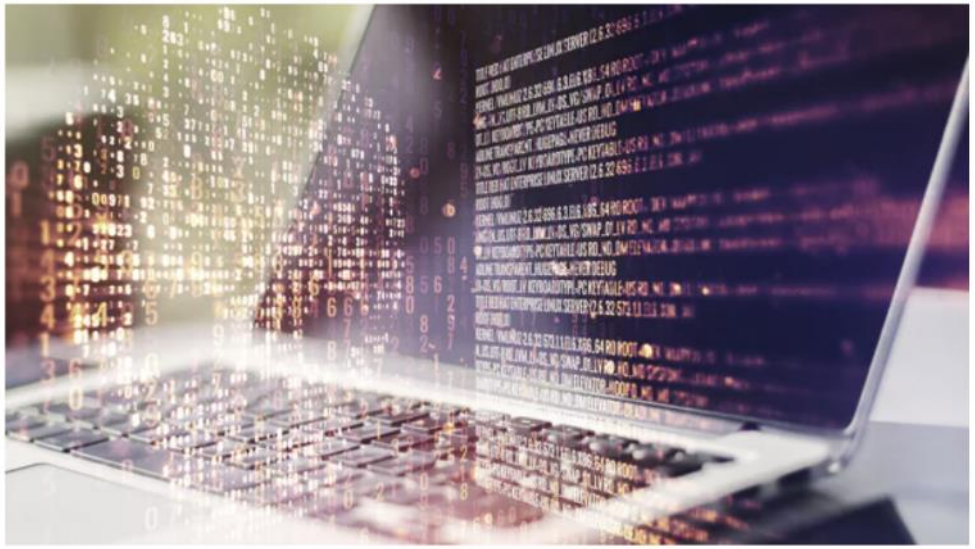
A. byte...
B. long, long
C. Byte, Byte
D. compilation error

https://blogs.oracle.com/oracleuniversity/post/test-your-java-knowledge-with-free-sample-questions

Developer: Natürliche Feinde der Softwarearchitektur?

Softwarearchitektur beeinflusst den Erfolg eines Projekts erheblich. Aber ausgerechnet "gute" Entwickler und Entwicklerinnen können Feinde der Architektur sein.

Lesezeit: **5 Min.** In Pocket speichern 132



(Bild: Pixels Hunter/Shutterstock.com)

20.04.2023 18:52 Uhr | Developer

Von Eberhard Wolff

DE https://www.heise.de/blog/Entwickler-innen-natuerliche-Feinde-der-Softwarearchitektur-8971097.html

Big Ball of Mud

Icon: Lisa Moritz

- Developers should really be afraid of complexity.
- Being able to handle it might actually be bad.

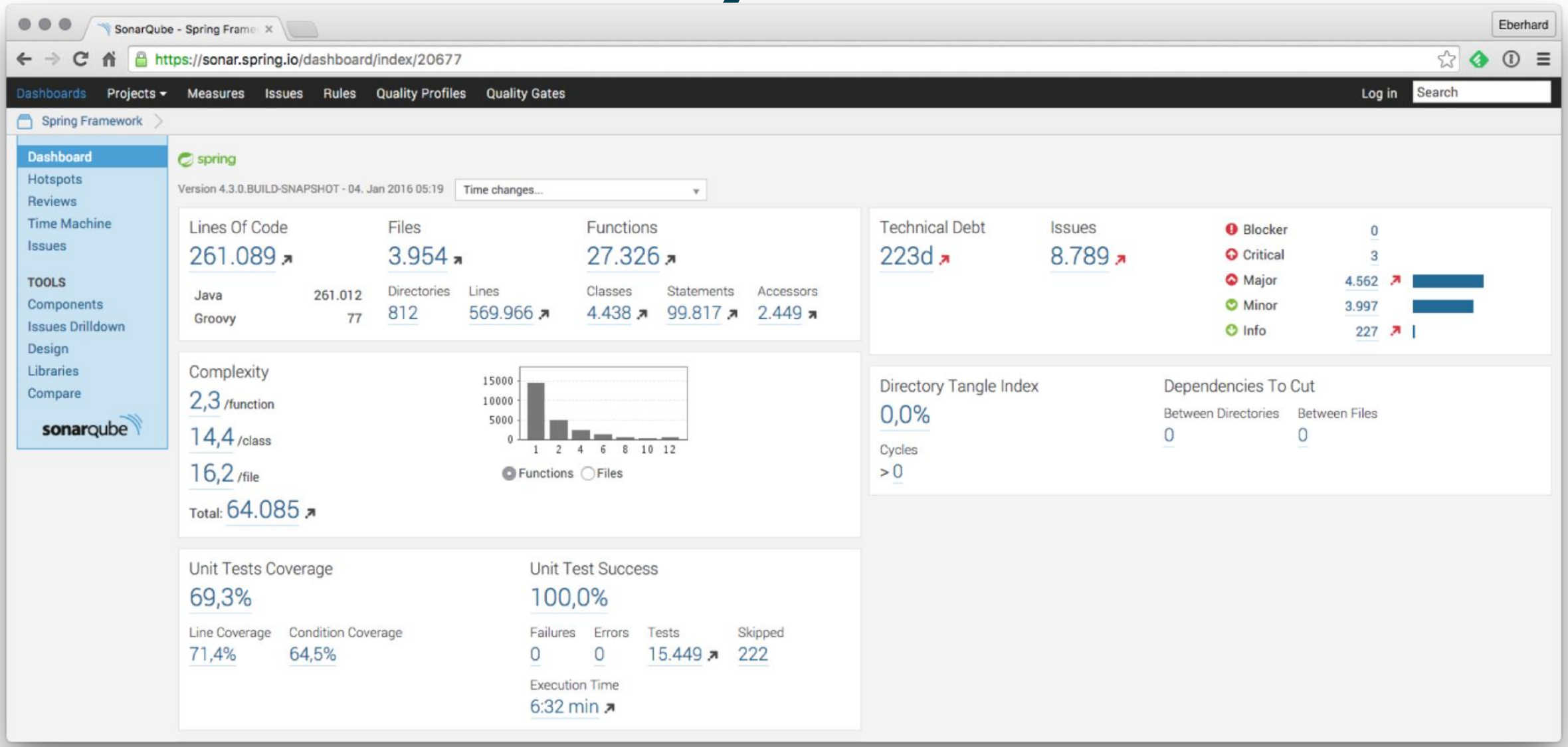
Micro- / Macro-Architecture

Micro- / Macro-Architecture

- Delegate decisions
- Macro architecture:
  Binding for all modules
- Micro architecture:
  Potentially different for all modules
- Micro architecture can be left to the teams

Micro- / Macro-Architecture: Static Code Analysis

Static Code Analysis



Should Static Code Analysis be Part of the Macro Architecture?

- Vote:

Yes, pre-defined metrics

Yes, teams decides about metrics

No

Micro- / Macro-Architecture

- Delegate decisions
- Macro architecture:
  Binding for all modules
- Micro architecture:
  Potentially different for all modules
- Micro architecture can be left to the teams

Should Static Code Analysis be Part of the Macro Architecture?

- IMHO No
- Goals: Teams should act autonomously.
- Teams must deliver a certain quality.
- They decide how to do that.

...with or without static code analysis.

Trust

- I trust the teams to deliver quality
- They will choose the means to do that.
- That might or might not include static code analysis

Limit: Trust

- Teams may not be trusted.
- E.g. external teams that are known to deliver poor quality.
- Manage quality via static code analysis?

Goodhart's Law

- Every measure which becomes a target becomes a bad measure.
- https://en.wikipedia.org/wiki/Goodhart%27s_law

Micro- / Macro-Architecture: Requirements Approach

Requirements: Different Approach

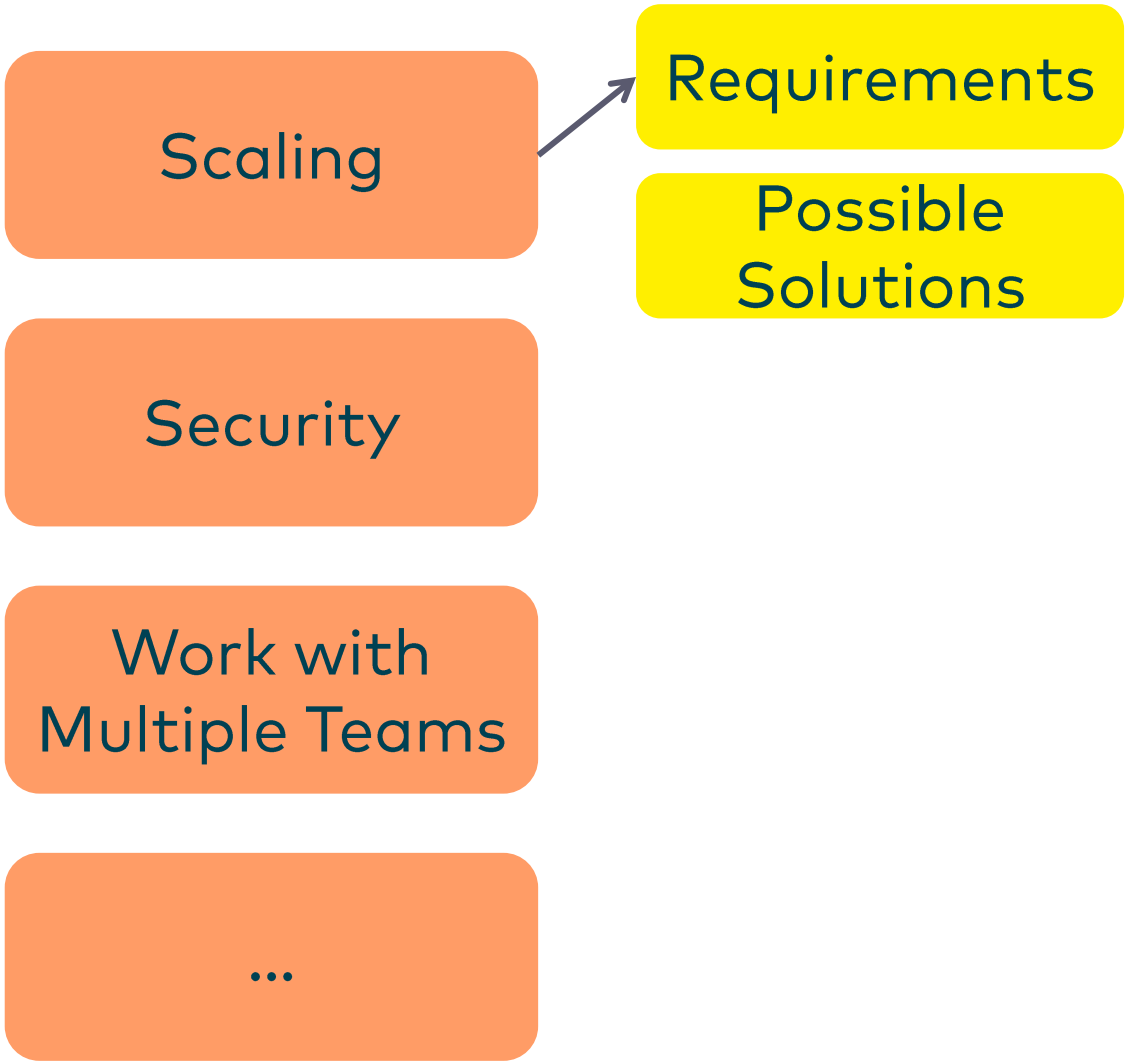
- Document that talks about requirements

...and how to handle them.

Chapters

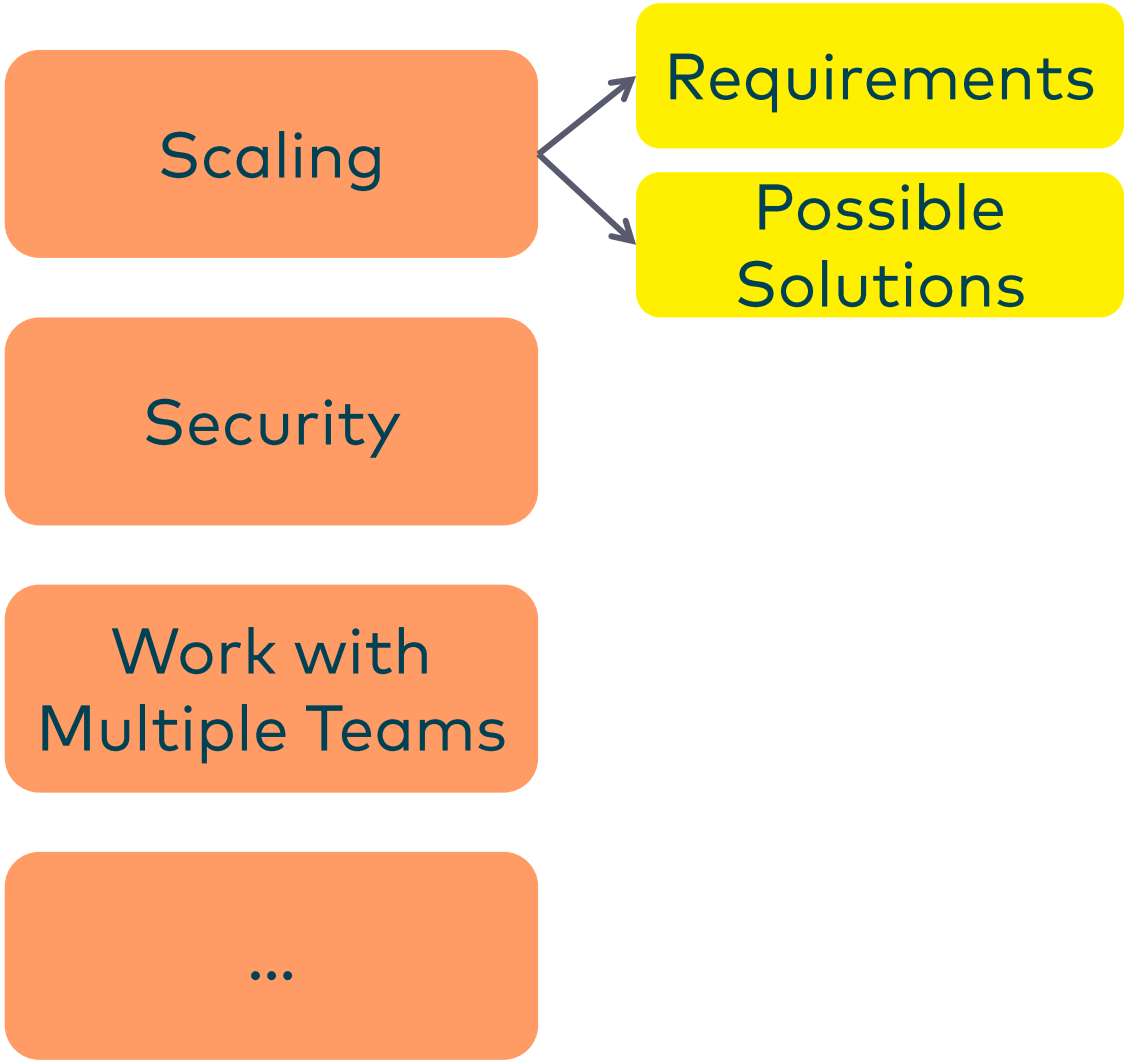
- Scaling
- Security
- Work with Multiple Teams
- ...

Scaling: Requirements



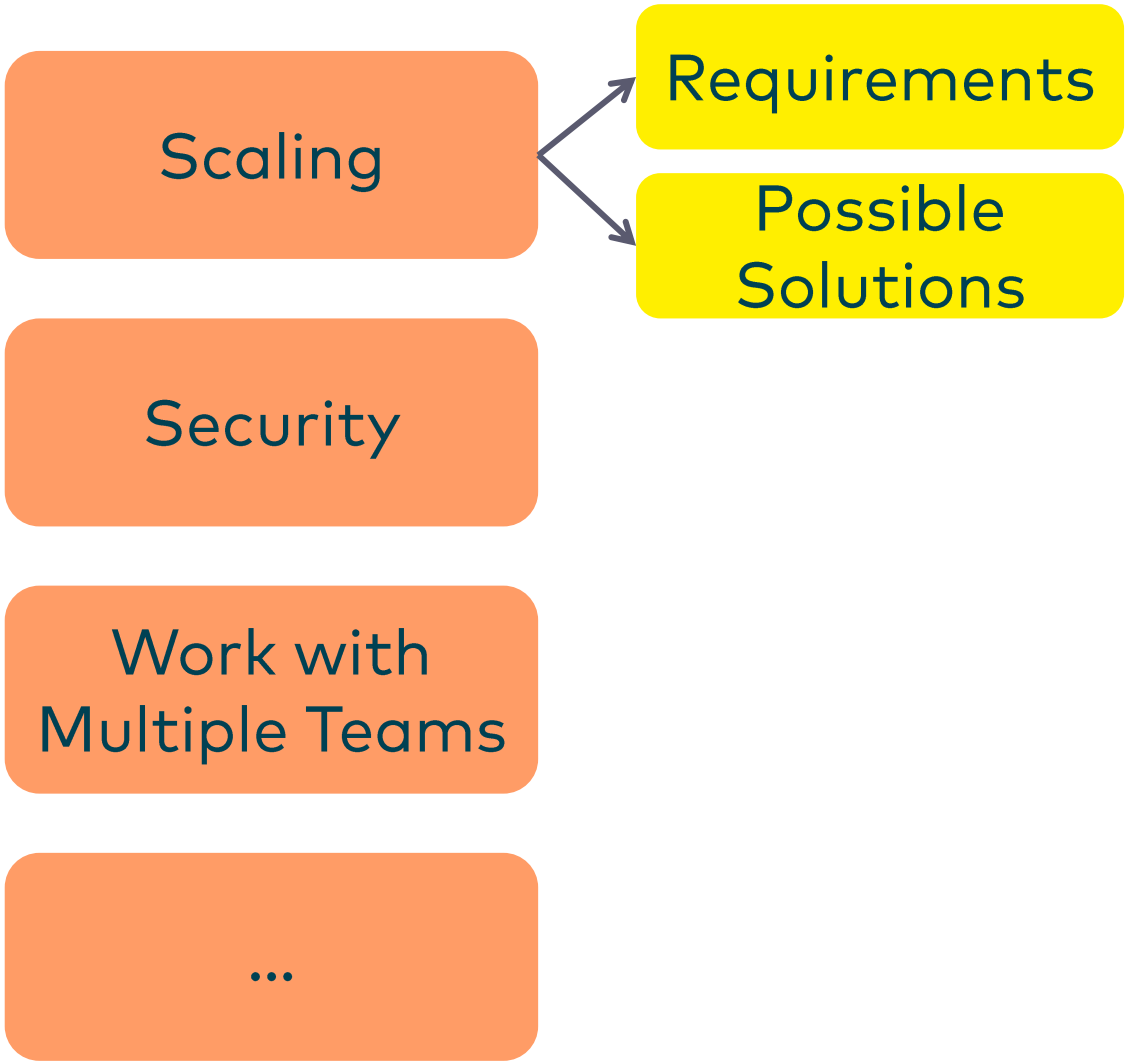
- Plan for growth!
- Refer to the business goals for details.
- Business goals are usually increased.
- Prepare for unplanned peaks!

Scaling: Possible Solutions



- Scale up
- Horizontal scaling
- Sharding
- Graceful degradation
- Asynchronous integration

Scaling: Possible Solutions



- Description

+ List of experts

+ Advantages / disadvantages

Requirements: Take Away

- Communicates trade-offs – the essence different solutions.
- Allows teams to make their own decisions – the essence of architecture.
- Actually focuses on supporting teams.
- More autonomy

Trust

- Trust teams fully to solve the problem

...or speak up.

- Support teams.
- Control?

Micro- / Macro-Architecture: Conclusion

When Chose What?

- Depends on persons, culture, and trust
- Some need to be controlled ☹
- Some want to be told what to do

Guidance / support

- Some want to decide by themselves

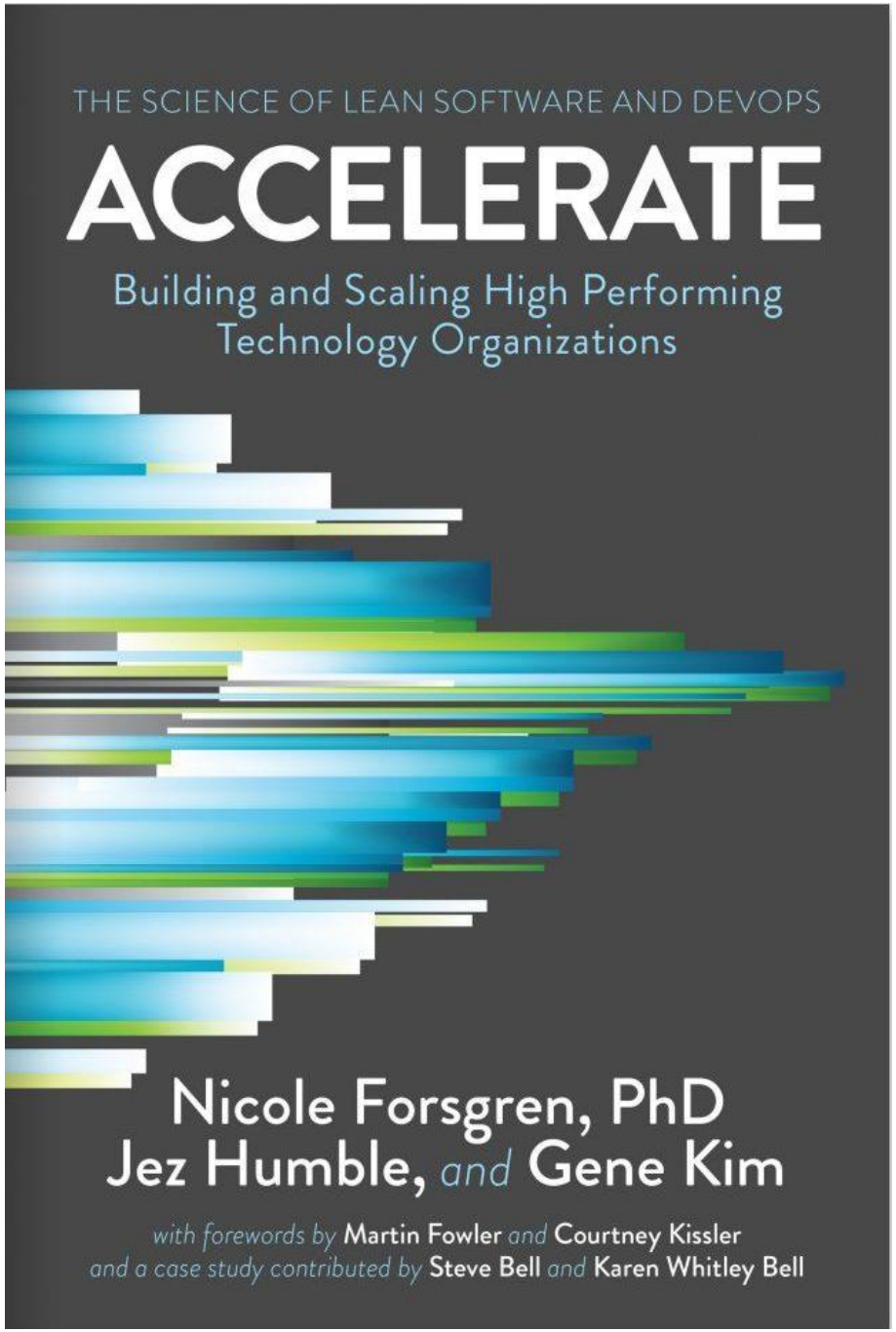
Really autonomous teams

What is important

is **enabling teams**

to make changes to their products or services

without depending on **other teams** or **systems**.

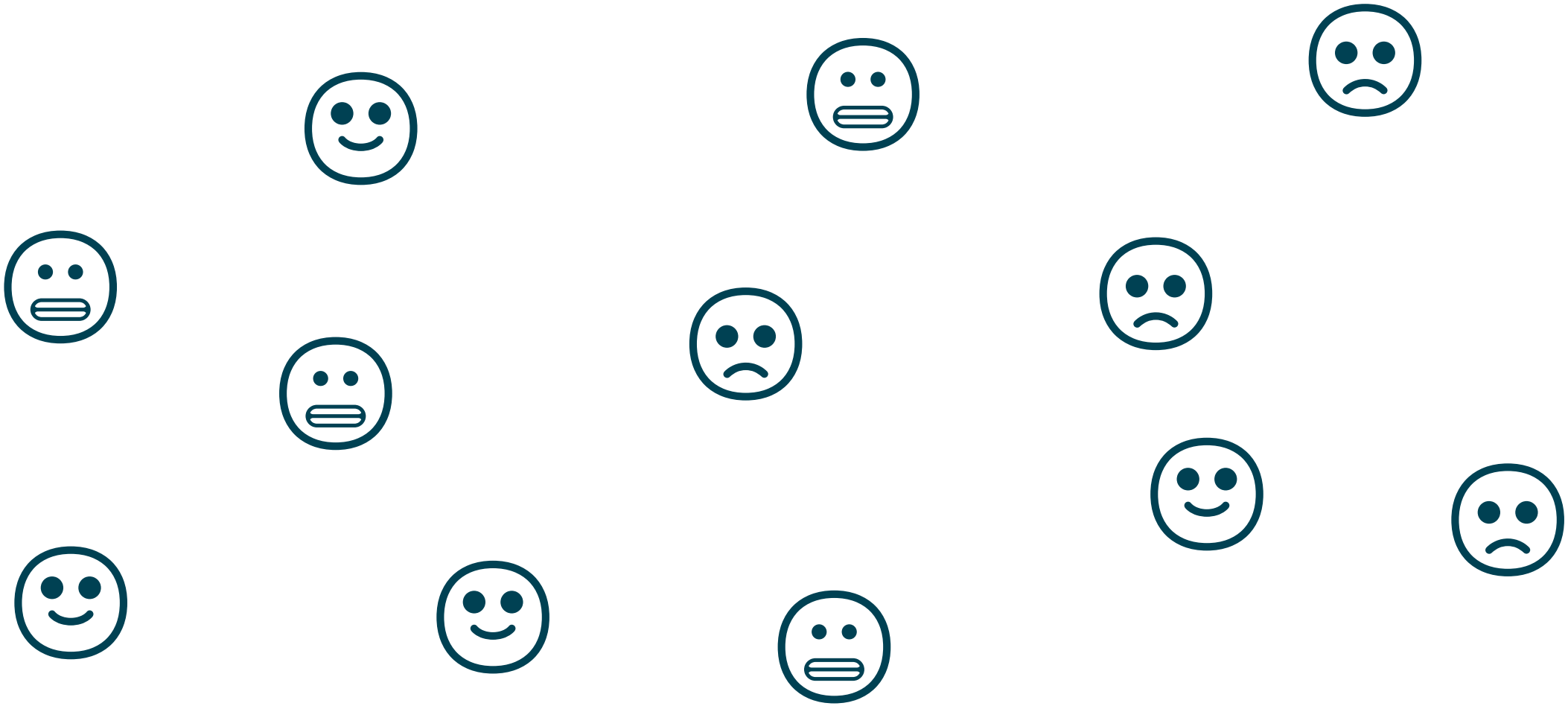


Inverse Conway

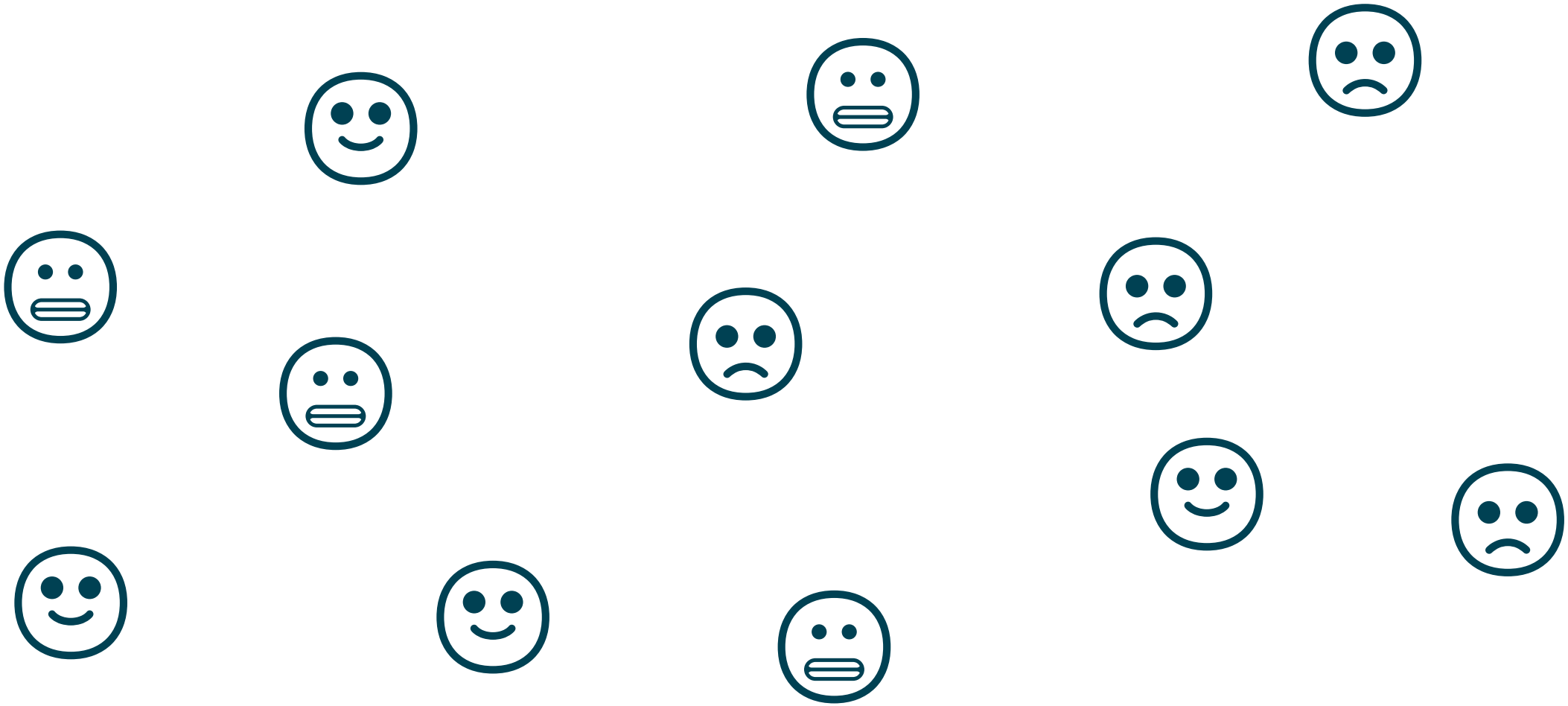
Inverse Conway Maneuver

- Architecture should drive organization
- I.e. set up the organization
- Architecture will follow

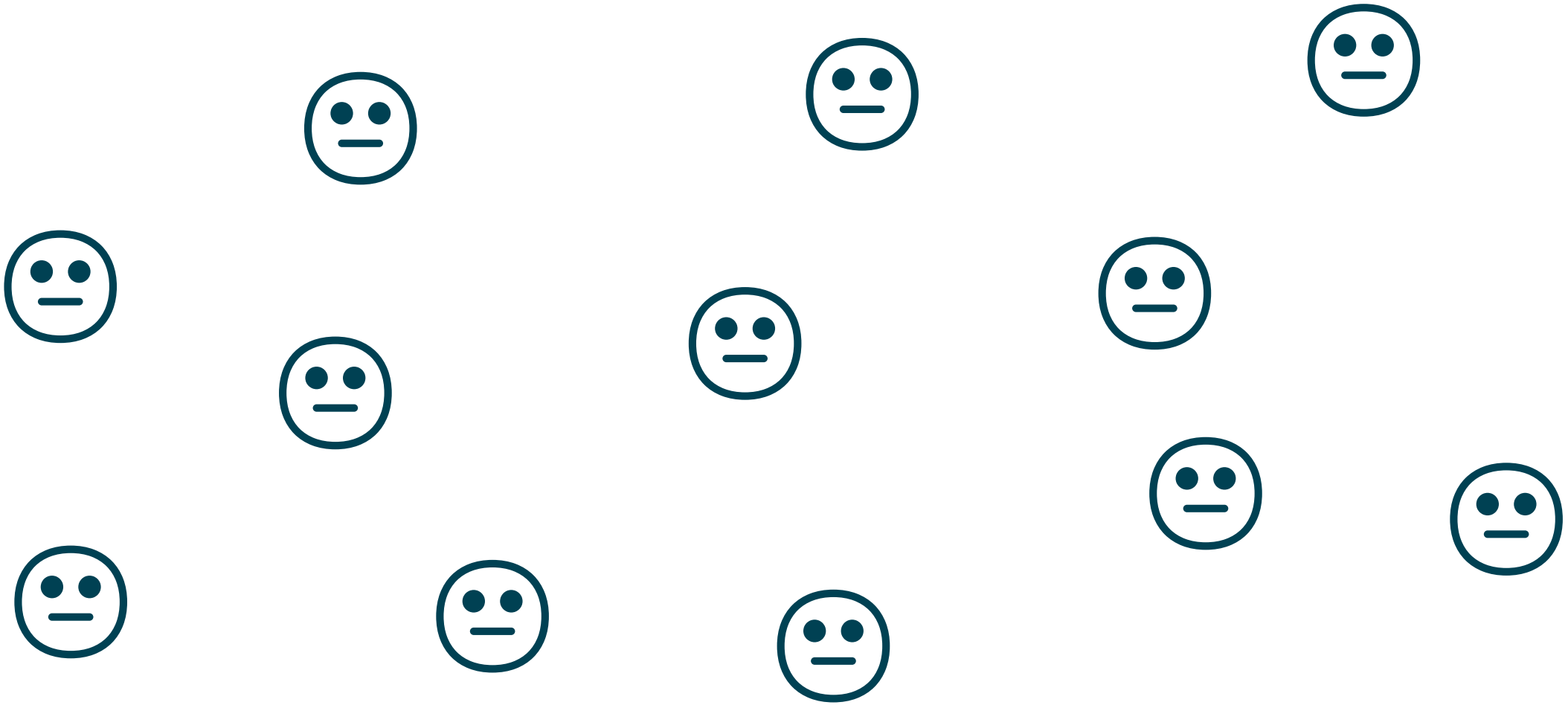
Developers, Designers ...



Chaos



Order



Order

Order Process 

Delivery 

Invoicing 

Order

Order Process

Modul

Delivery

Modul

Invoicing

Modul

Inverse Conway: Simplification

- Inverse Conway changes the org chart
- Org chart is not communication!
- Assumption: Org chart team will collaborate on module & communicate more internally
- Does it work that way?
- What if members of different teams sit in the same room?

Inverse Conway: Simplification

- Do you think people will just follow a reorg?
- Do you think people in the same room will work more closely together?
- Why I am doing the presentation? What is the news?
- We know but we don’t use the knowledge

Irritating the Organization

- Sociology: “irritating” organizations.
- New org chart: irritation
- Can lead to new communication structure
- Can lead to org chart teams working on modules.
- Might also be completely ignored.
- https://software-architektur.tv/2020/09/10/folge016.html

Inverse Conway: Assumptions

- People will follow the org chart.
- People will communicate according to the org chart.
- Too simplistic

What Now?

Conclusion

- Architecture is for people to better understand software.
- So: There is no absolute good / bad architecture.
- It depends on people.

Understand Your Problem!

- Software or Humans?
- Legacy because humans left?
- ...and maybe not even a big ball of mud

Fix the Organization?

- I want to develop software
- …not fix the organization
- Agile has the same problem

Live with It

- If you don't want to / can't fix the organization, you will have to live with it.
- You might need to adjust your architecture

Humans, not Robots

- Computers should be deterministic
- (Yes, I know it doesn't seem like it)
- Humans are not deterministic.
- Don't simplify like the inverse Conway Maneuver!
- Actually, we all know but we are not explicit about this.

Psychological Safety

- Without feedback no progress
- So: Need to create an environment where people feel safe to provide and receive feedback
- Psychological safety

Organisation

- Folge 163 - Kommunikation im Entwicklungsprozess mit Rebecca Temme
- Folge 147 - Wie reißt man den Elfenbeinturm ein? mit Anja Kammer
- Folge 141 - Auftragstaktik - Agilität beim Militär? mit Sönke Marahrens
- Folge 125 - Organisation und Architektur - ein Beispiel
- Folge 115 - Data Mesh - nur ein neuer Datenanalyse-Hype?
- Folge 110 - Conway's Law
- Folge 106 - Anne Herwanger, Alexandra Hoitz, Stefan Link - Resiliente Organisation und resiliente Software Architektur - live von der OOP
- Episode 101 - Kenny Baas-Schwegler, Gien Verschatse, Evelyn Van Kelle - Facilitating Collaborative Design Decisions - Live from OOP
- Folge 96 - Organisation, Architektur - Was ich im Stream gelernt habe
- Folge 91 - Sven Johann - Cross-funktionale Teams zielgerichtet in den Abgrund stürzen
- Episode 82 - Avraham Poupko & Kenny Baas-Schwegler - The Influence of Culture on Software Design
- Epsiode 80 - Microservices, Inverse Conway Maneuver, and Flow with James Lewis - Live from Software Architecture Gathering
- Folge 73 - Das Spotify-Modell gibt es gar nicht!
- Folge 63 - Kim Nena Duggen zu Soft Skills für Software-Architekt:innen
- Folge 16 - Gerrit Beine zu Sozialwissenschaften und Software-Architektur
- Folge 2 - Organisation und Architektur

https://software-architektur.tv/tags.html#Organisation

Send email to jax2023@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