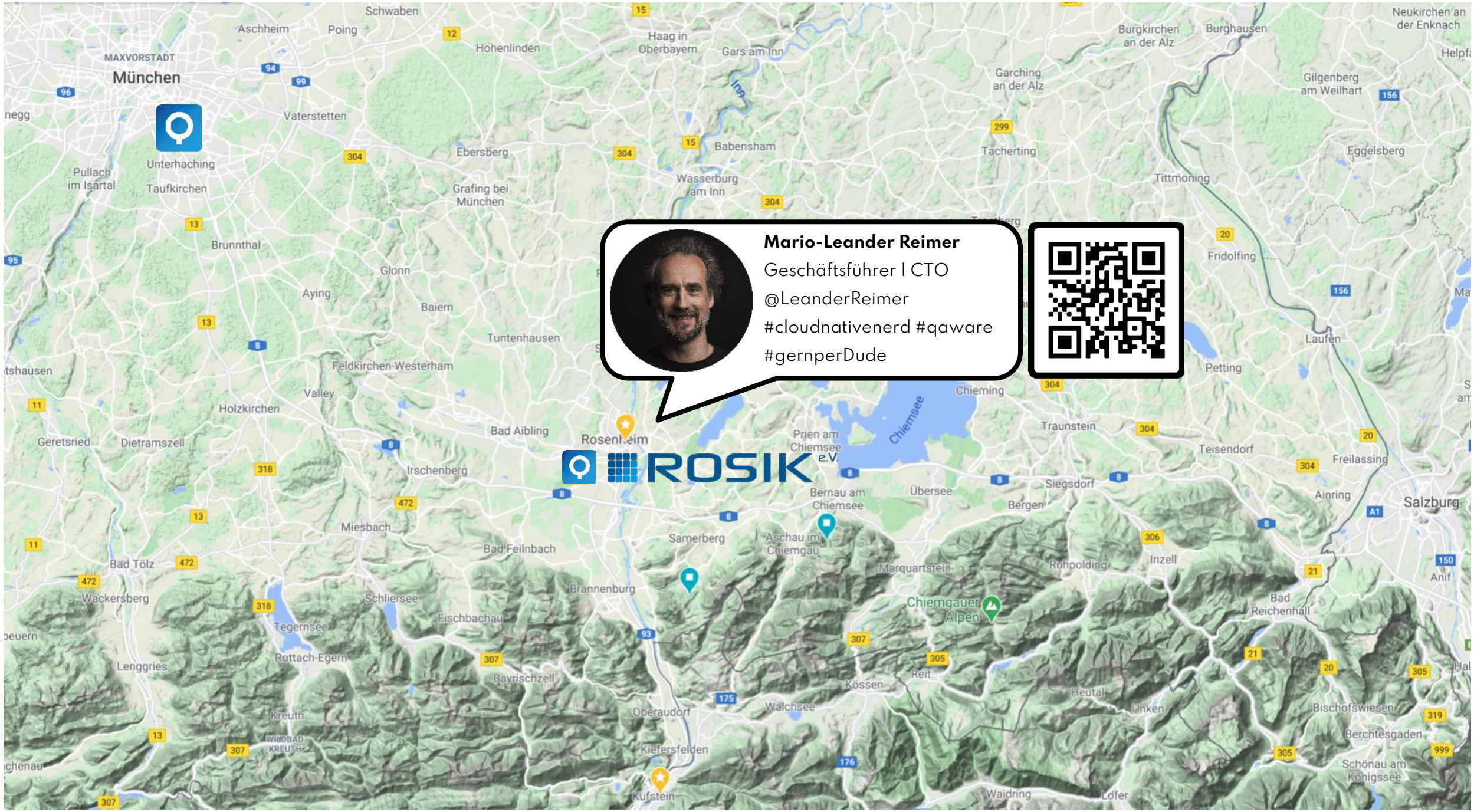
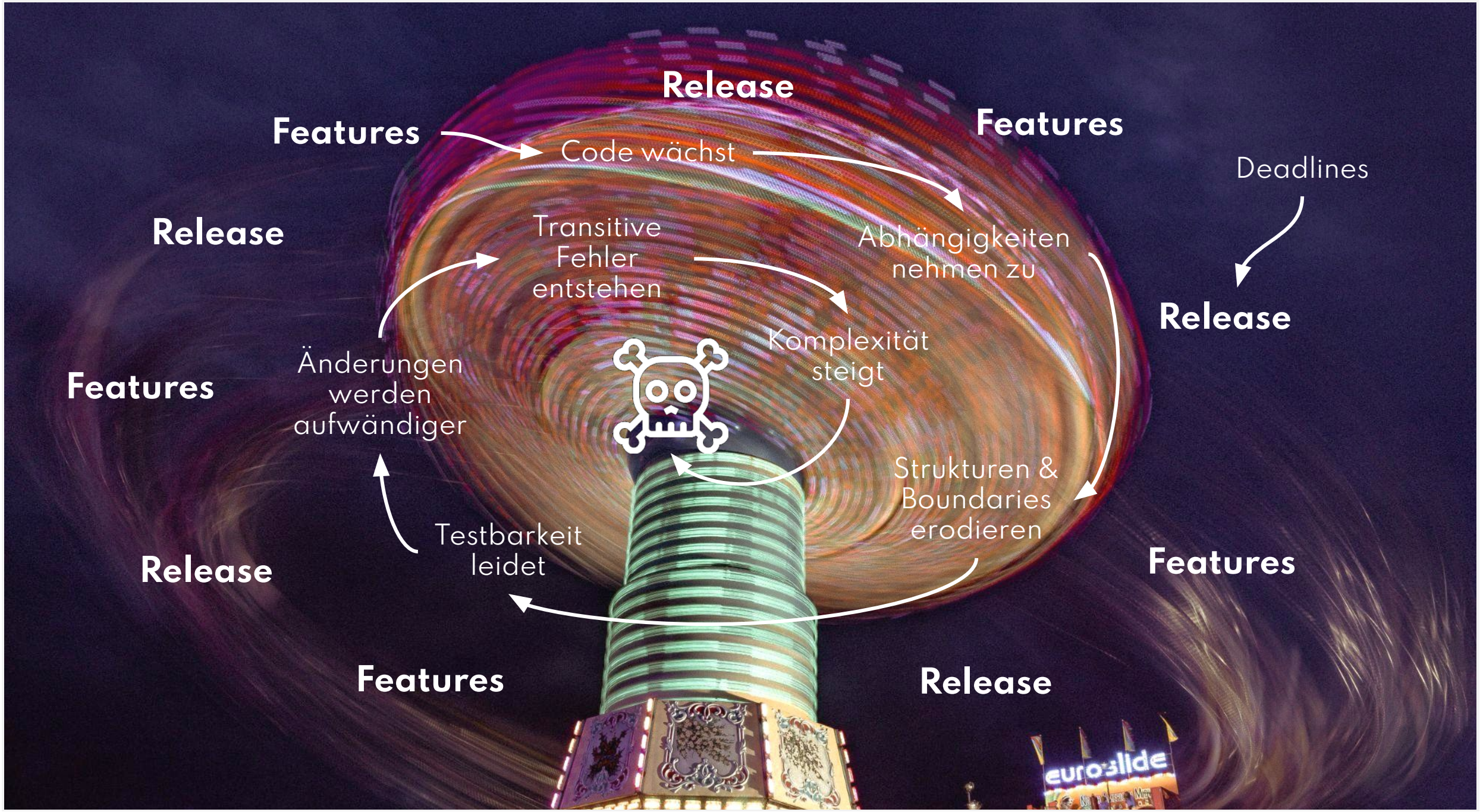
qaware.de

QA|WARE
SOFTWARE ENGINEERING

Methoden und Tools für eine nachhaltig saubere Architektur in agilen Projekten

Mario-Leander Reimer
mario-leander.reimer@qaware.de
@LeanderReimer


Mario-Leander Reimer
Geschäftsführer | CTO
@LeanderReimer
#cloudnativenerd #qaware
#gernperDude
ROSIK e.V.
Rosenheim
München
Unterhaching
Aschheim
Poing
Schwaben
Hohenlinden
Haag in Oberbayern
Gars am Inn
Burgkirchen an der Alz
Burghausen
Neukirchen an der Enknach
Garching an der Alz
Gilgenberg am Weilhart
Vaterstetten
Ebersberg
Babensham
Tacherting
Eggelsberg
Pullach im Isartal
Taufkirchen
Wasserburg am Inn
Tittmoning
Grafing bei München
Brunnthal
Fridolfing
Glonn
Aying
Baiern
Tuntenhausen
Laufen
Feldkirchen-Westerham
Petting
Valley
Holzkirchen
Chieming
Bad Aibling
Prien am Chiemsee
Chiemsee
Traunstein
Geretsried
Dietramszell
Teisendorf
Freilassing
Irschenberg
Bernau am Chiemsee
Übersee
Siegsdorf
Ainring
Salzburg
Bergen
Miesbach
Samerberg
Aschau im Chiemgau
Bad Feilnbach
Inzell
Bad Tölz
Marquartstein
Ruhpolding
Anif
Wackersberg
Brannenburg
Schliersee
Chiemgauer Alpen
Bad Reichenhall
Fischbachau
Tegernsee
Rottach-Egern
Lenggries
Reit
Kössen
Bayrischzell
Heutal
Unken
Bischofswiesen
Oberaudorf
Walchsee
Berchtesgaden
Kufstein
Kiefersfelden
Schönau am Königssee
Waidring
Lofer

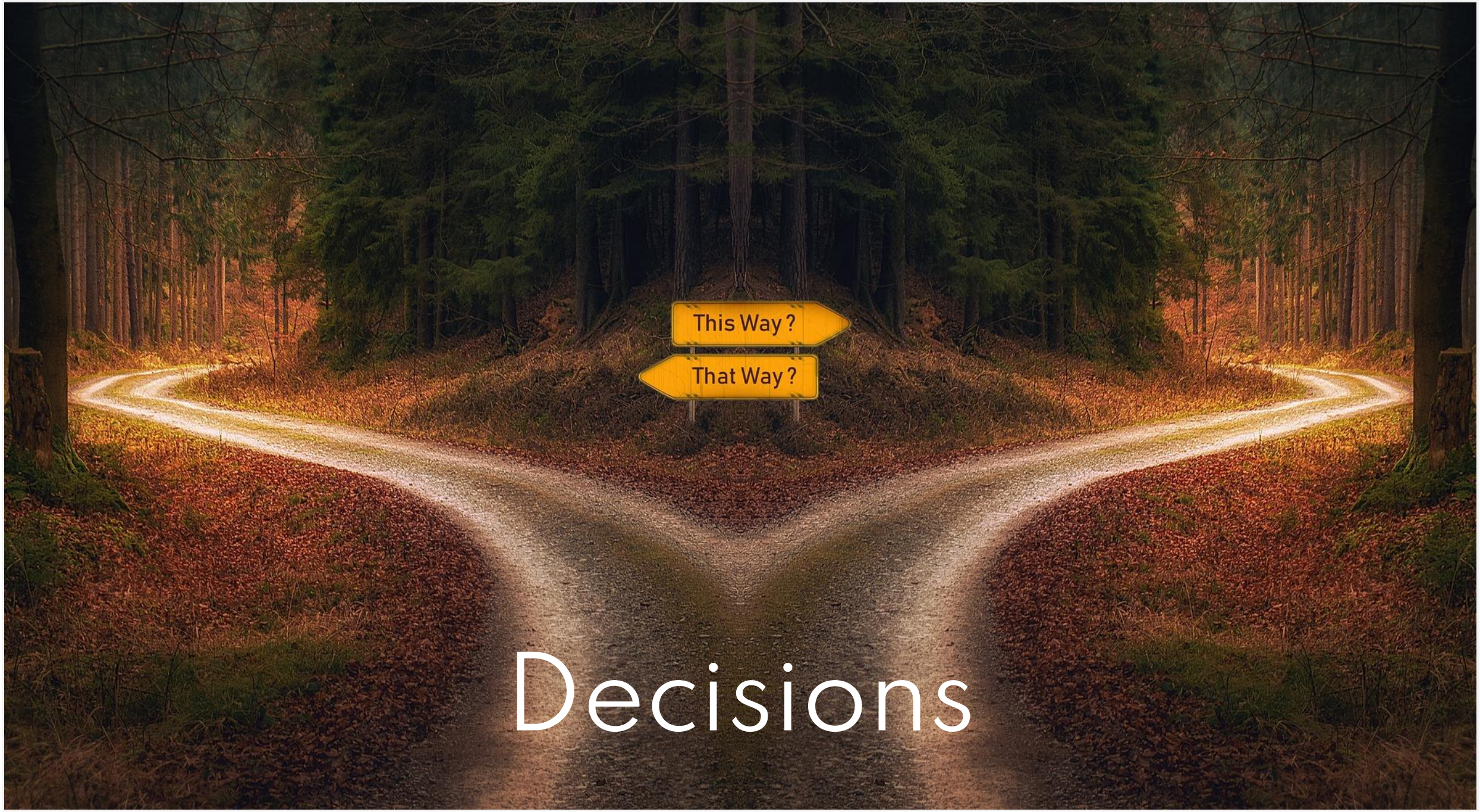

Release
Features
Code wächst
Features
Deadlines
Release
Transitive Fehler entstehen
Abhängigkeiten nehmen zu
Release
Komplexität steigt
Features
Änderungen werden aufwändiger
Strukturen & Boundaries erodieren
Testbarkeit leidet
Release
Features
Features
Release
eurosllide

slido

QA|WARE

Was sind die Gründe für eine Architektur Erosion in agilen Projekten?

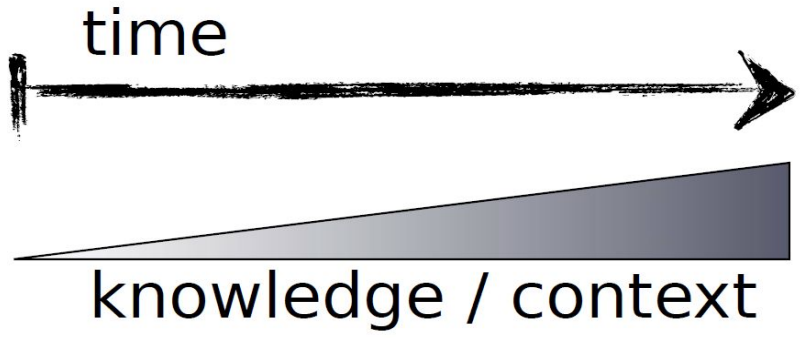
ⓘ Start presenting to display the poll results on this slide.



Decisions

The Last Possible Moment != The Last Responsible Moment

- Welche Fragen sind für verantwortungsvolle Entscheidungen relevant?
  - Muss ich die Entscheidung jetzt treffen?
  - Kann ich es noch aufschieben?
  - Was passiert, wenn ich es nicht mache?
  - Wann wird die Komplexität zu hoch?
  - Welche Alternativen gibt es?
  - Was sind die Trade-Offs der Entscheidung?
  - Kann die Entscheidung rückgängig gemacht werden?



- The Last Responsible Moment ist die bessere Wahl!
- Keine Angst vor suboptimalen Entscheidungen.
- Iterative Entwicklung, Tests, Refactoring, Continuous Integration und Architecture Fitness Functions helfen das Risiko beherrschbar zu machen

DOCUMENTING ARCHITECTURE DECISIONS

Michael Nygard - November 15, 2011

AGILITY ARCHITECTURE

CONTEXT

Architecture for agile projects has to be described and defined differently. Not all decisions will be made at once, nor will all of them be done when the project begins.

Agile methods are not opposed to documentation, only to valueless documentation. Documents that assist the team itself can have value, but only if they are kept up to date. Large documents are never kept up to date. Small, modular documents have at least a chance at being updated.

One of the hardest things to track during the life of a project is the motivation behind certain decisions. A new person coming on to a project may be perplexed, baffled, delighted, or infuriated by some past decision. Without understanding the rationale or consequences, this person has only two choices:

1. **Blindly accept the decision.**

   This response may be OK, if the decision is still valid. It may not be good, however, if the context has changed and the decision should really be revisited. If the project accumulates too many decisions accepted without understanding, then the development team becomes afraid to change anything and the project collapses under its own weight.

2. **Blindly change it.**

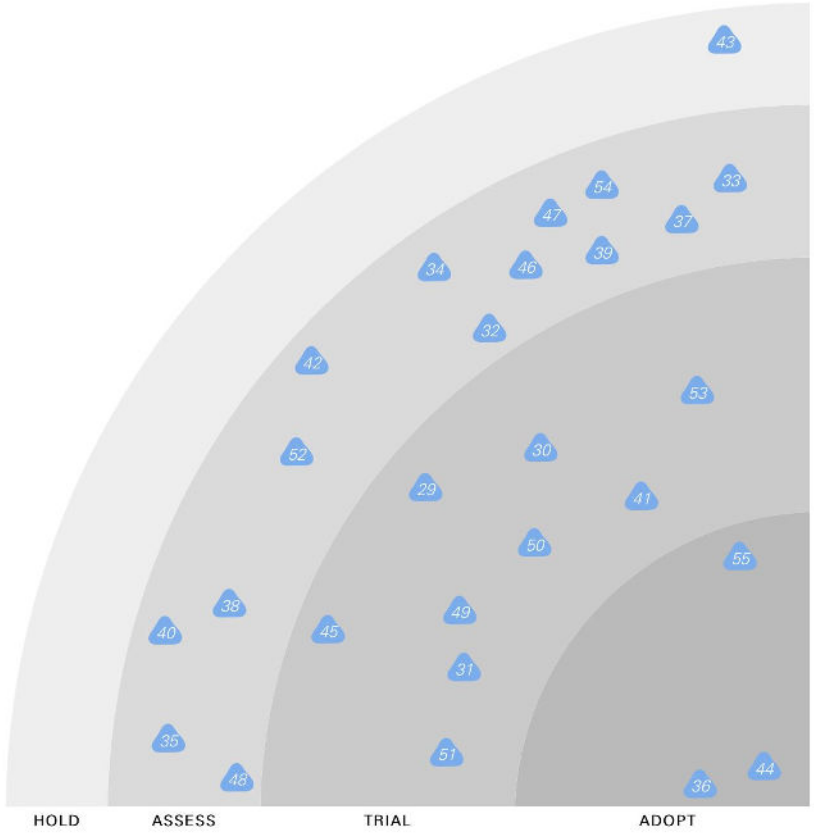
   Again, this may be OK if the decision needs to be reversed. On the other hand, changing the decision without understanding its motivation or consequences could mean damaging the project's overall value without realizing it. (E.g., the decision supported a non-functional requirement that hasn't been tested yet.)

It's better to avoid either blind acceptance or blind reversal.

https://cognitect.com/blog/2011/11/15/documenting-architecture-decisions

Architecture Decision Records (ADR)

QA|WARE



MAY 2018

Adopt

Much documentation can be replaced with highly readable code and tests. In a world of evolutionary architecture, however, it's important to record certain design decisions for the benefit of future team members as well as for external oversight. **Lightweight Architecture Decision Records** is a technique for capturing important architectural decisions along with their context and consequences. We recommend storing these details in source control, instead of a wiki or website, as then they can provide a record that remains in sync with the code itself. For most projects, we see no reason why you wouldn't want to use this technique.

https://www.thoughtworks.com/radar/techniques/lightweight-architecture-decision-records

The Anatomy of an Architecture Decision Records (ADR)

QA|WARE

- Short text file; 1-2 pages long, one file per decision
- Simple format like Markdown, Asciidoc, TXT, etc.

Short noun phrase and number,
e.g. ADR 5: AWS as preferred cloud provider

Describes the forces at play: technology, political, project local. Value neutral, simple facts.

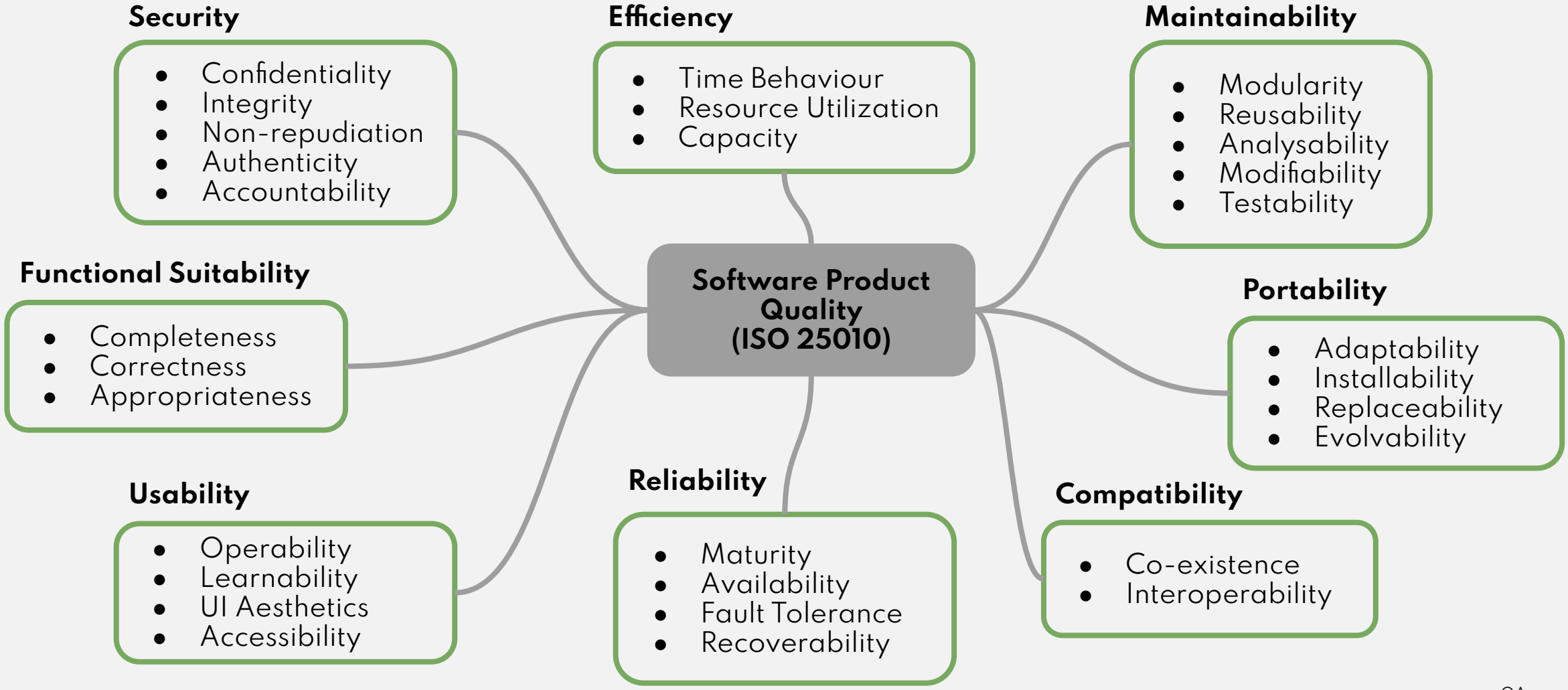
Response to the forces with justification. Stated in full sentences, with active voice. "We will …"

Proposed, Accepted, Deprecated, Superseded

Describe context, after applying the decision. All consequences should be listed here, not just the "positive" ones.

Controlling

Nicht nur Maintainability ist wichtig. Clean Architecture ist Enabler für ganzheitliche Qualität!

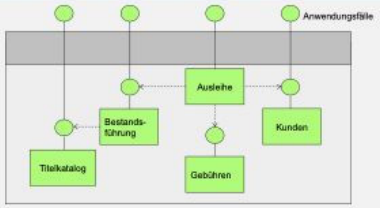


Die Clean Architecture Konzepte müssen auf allen Ebenen einer Software-Architektur angewendet werden.

QA|WARE

Conceptual

Goal: Structures the system from a business perspective

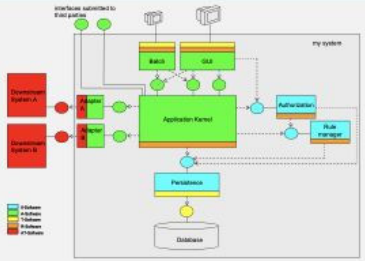


Typical Inputs:
- Data Models
- Use Cases
- Functional Requirements

Typical Outputs:
- Business Domain Components
- Business Domain Interfaces
- System Context (Embeddingt into the system landscape)

Technical

Goal: Defines how to map and run the conceptual view on the infrastructure

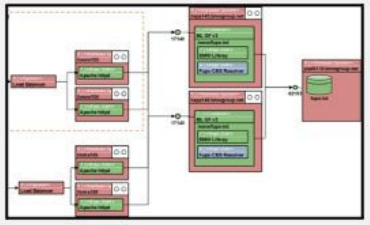


Typical Inputs:
- Non-functional Requirements
- Enterprise Standards
- Reference Architectures, Patterns and Blueprints

Typical Outputs:
- Layers, Onions, Hexagons, …
- Reusable common components (0-components) and technical components (T components)
- Selection of open source as building blocks for T/0-components (Software OEM)

Infrastructure

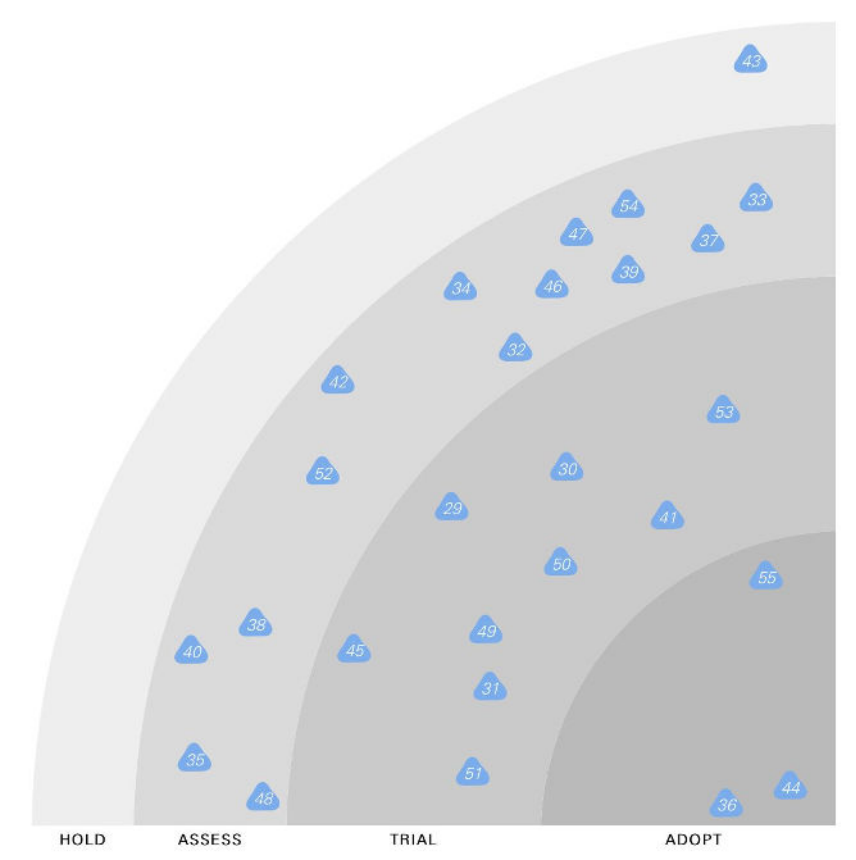
Gial: Defines the execution environment to run and operate our systems



Typical Inputs:
- Non functional requirements
- Enterprise Standards
- Existing corporate infrastructure environments

Typical Outputs:
- Hardware Infrastructure (Compute, Network, Storage, …)
- Software Infrastructure (OS, DB, App-Servers, …) and protocols
- Ops components (individual executable pieces of the application)

Architecture Fitness Functions validieren die geforderten (nicht)-funktionalen System-Eigenschaften kontinuierlich.



MAY 2018

TRIAL

Borrowed from evolutionary computing, a fitness function is used to summarize how close a given design solution is to achieving the set aims. When defining an evolutionary algorithm, the designer seeks a 'better' algorithm; the fitness function defines what 'better' means in this context. An **architectural fitness function** , as defined in **Building Evolutionary Architectures**, provides an objective integrity assessment of some architectural characteristics, which may encompass existing verification criteria, such as unit testing, metrics, monitors, and so on. We believe architects can communicate, validate and preserve architectural characteristics in an automated, continual manner, which is the key to building evolutionary architectures.

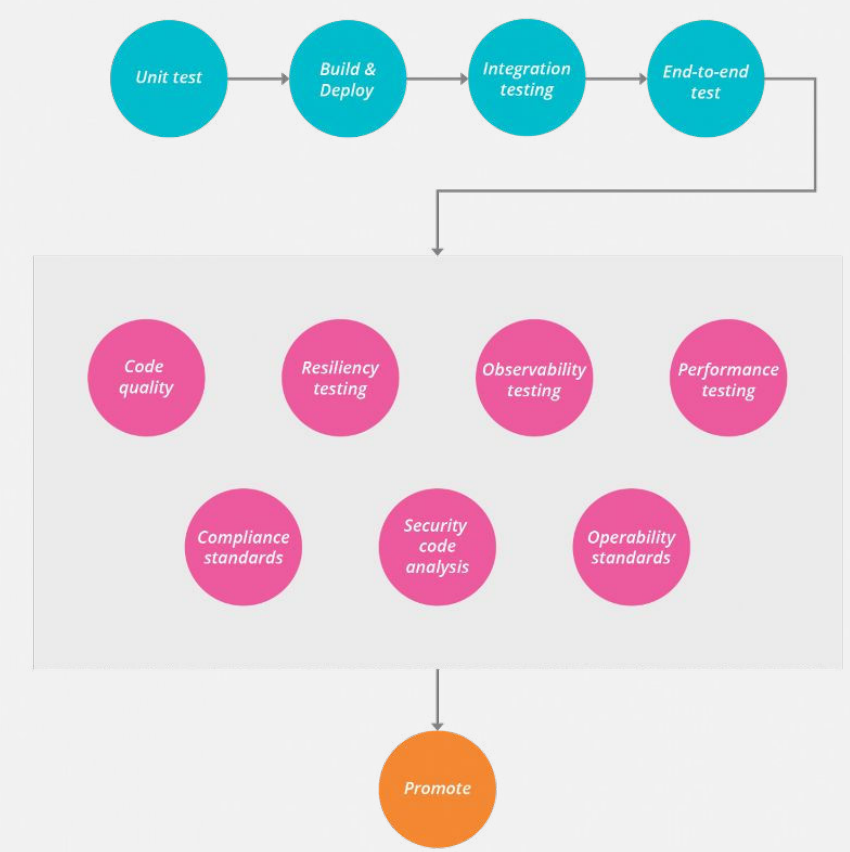
https://www.thoughtworks.com/de/radar/techniques/architectural-fitness-function

Fitness-function Driven Development

QA|WARE

- Architektur ist wie ein Produkt mit User Journeys
- Anforderungen werden von den Stakeholdern eingesammelt
  - Business
  - Compliance
  - Operations
  - Security
  - Infrastructure
- Das sind häufig unsere „-illities“ und Qualitätsmerkmale

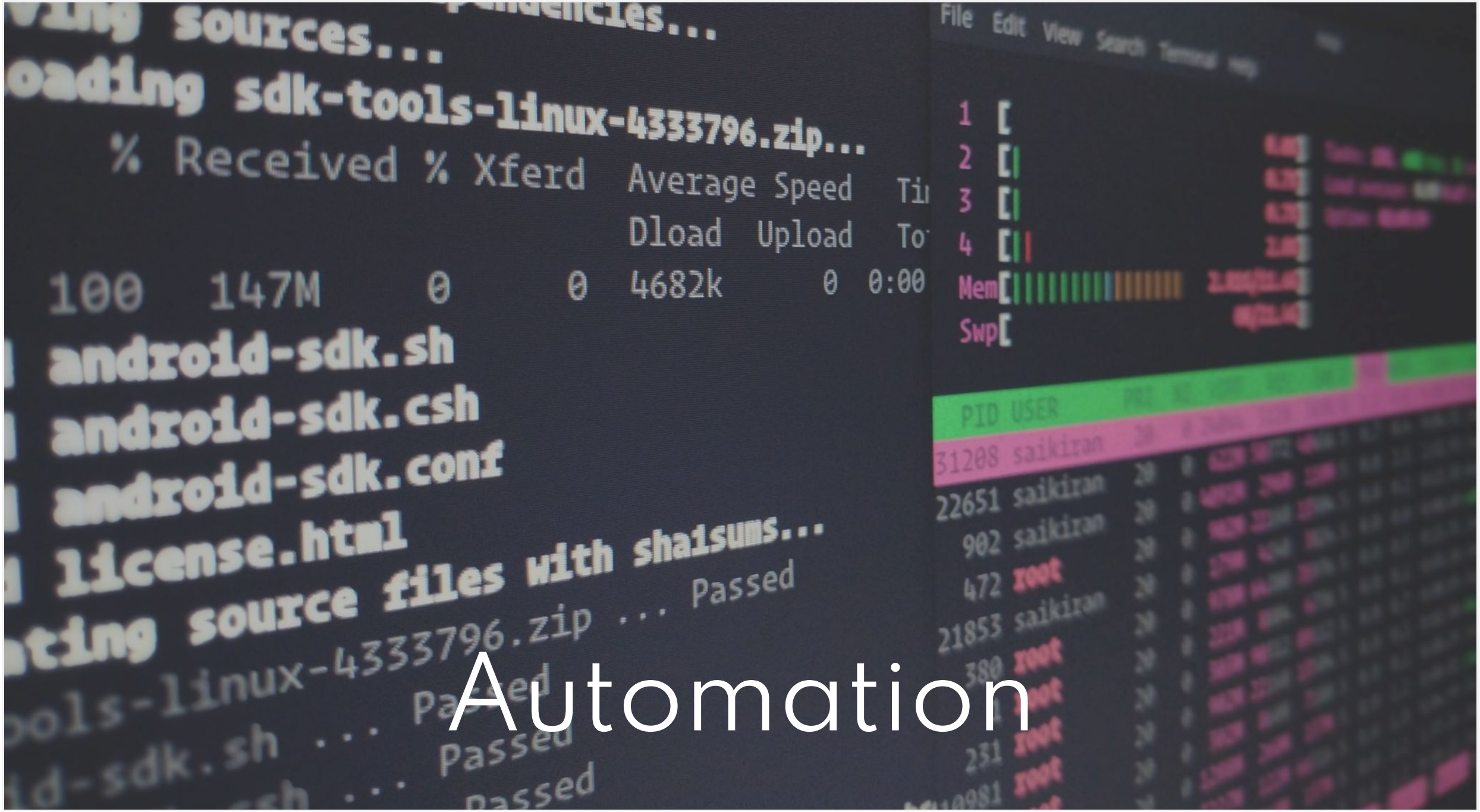
- Die Akzeptanz-Kriterien werden mit einem BDD Test Framework formuliert.
- Tests werden als Teil der CI/CD Pipeline ausgeführt, und anschließend verifiziert



https://www.thoughtworks.com/de/insights/articles/fitness-function-driven-development

Beispiele für Architectural Fitness Functions

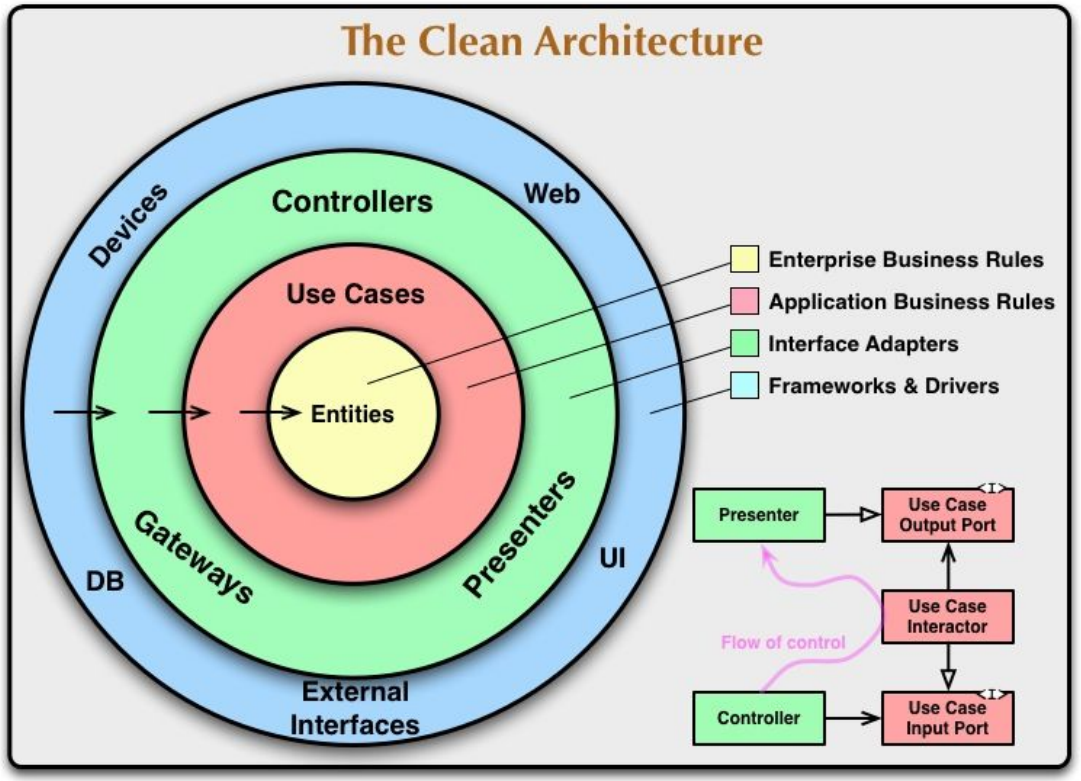

```
describe "Performance" do
     it "completes a transaction in under 2 seconds" do
          expect(k6.check_transaction_round_trip_time()).to < 2
     end

     it "has less than 10% error rate for 10000 transactions" do
          expect(k6.check_error_rate_for_transactions(10000)).to < .1
     end
end

describe "Security" do
     describe “Static Analysis” do
          it "should not have plaintext secrets in codebase" do
               expect(code.has_secrets_in_codebase()).to_not be(true)
          end
     end

     describe “Dynamic Analysis” do
          it "should not have any of the OWASP Top 10" do
               expect(zap.has_owasp_top_10_vulnerabilities()).to be(false)
          end
     end
end
```

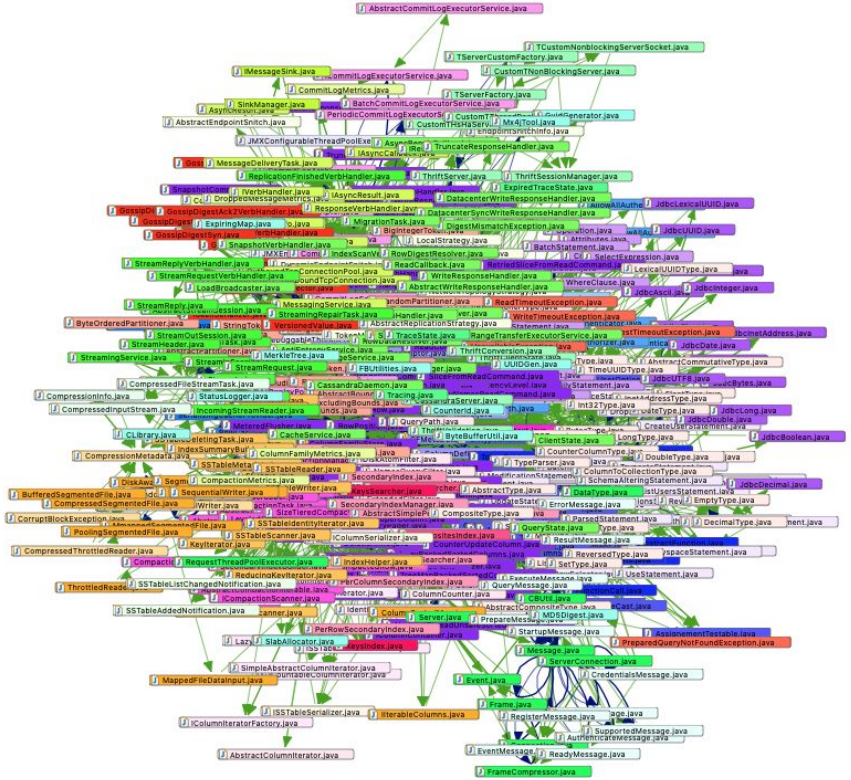

sources...
oading sdk-tools-linux-4333796.zip...
% Received % Xferd Average Speed
Dload Upload
100 147M 0 0 4682k 0 0:00
android-sdk.sh
android-sdk.csh
android-sdk.conf
license.html
ting source files with shasums...
ools-linux-4333796.zip ... Passed
d-sdk.sh ... Passed
sh ... Passed
Passed
File Edit View Search
PID USER
31208 saikiran
22651 saikiran
902 saikiran
472 root
21853 saikiran
380 root
231 root

Automation

Menschen machen Fehler. Von der Clean Architecture zum Big Ball of Mud geht's schneller als man glaubt!

QA|WARE





ArchUnit ermöglicht die einfache automatisierte Prüfung einer Software-Architektur in Form von Unit Tests.

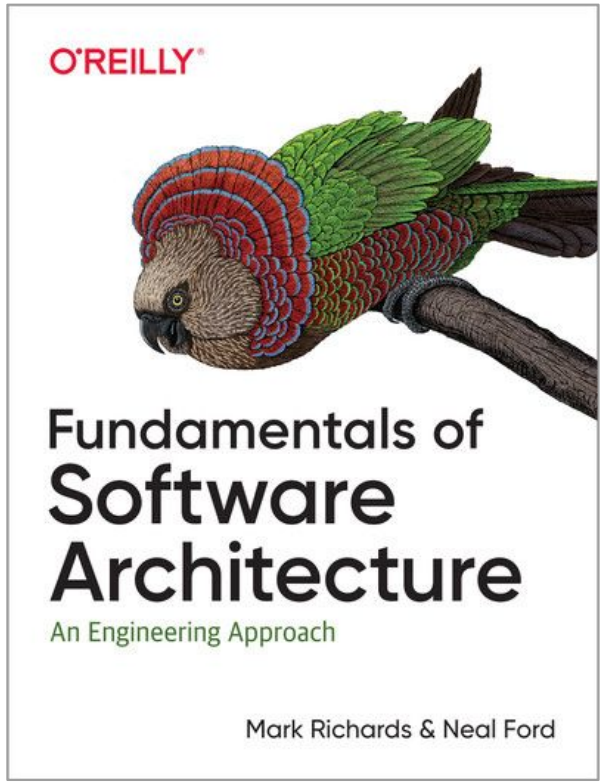
- https://www.archunit.org/
- Freie (Apache v2), einfache und erweiterbare Bibliothek zur Prüfung der Architektur von Java Code. Gibt es auch für .NET/C#.
- Alle gängigen Build Tools und Unit Test Frameworks werden unterstützt
- Das ArchUnit Library API bietet eine Sammlung an vordefinierten Regeln für wiederkehrende Architektur-Prüfungen
  - *Architectures*: Regeln zur Überprüfung von Layered und Onion Architectures
  - *Slices*: Erkennung von "Cycles" auf unterschiedlichen Ebenen
  - *General*: Sammlung von Regeln für Good Coding Practices (z.B: Logging, Exceptions, …)
  - *PlantUML*: Regeln zum Abgleich der Codebase mit einem PlantUML Modell
  - *Freezing Arch Rules*: Erlaubt die Definition einer Baseline für Violations, besonders nützlich um Technische Schulden in Altprojekten zu managen

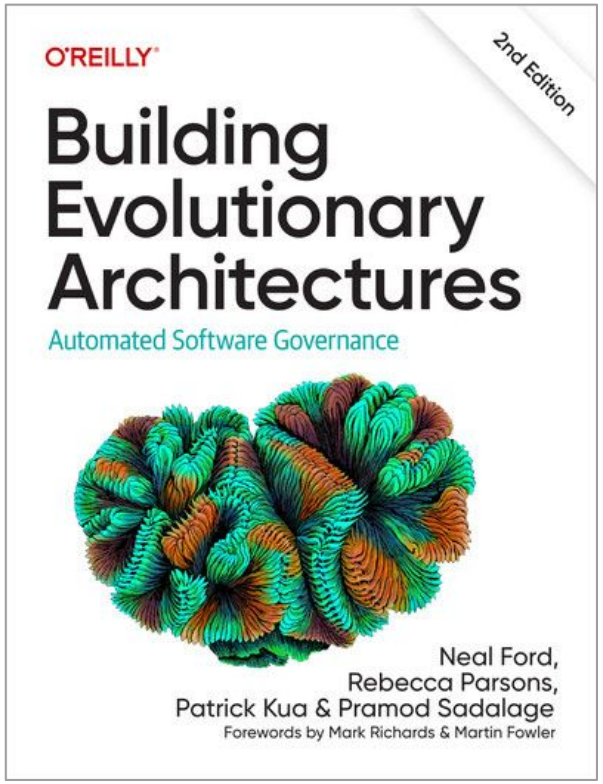
Nur wenige Zeilen Code validieren unsere Clean Architecture kontinuierlich und wiederholbar bei jedem Build.

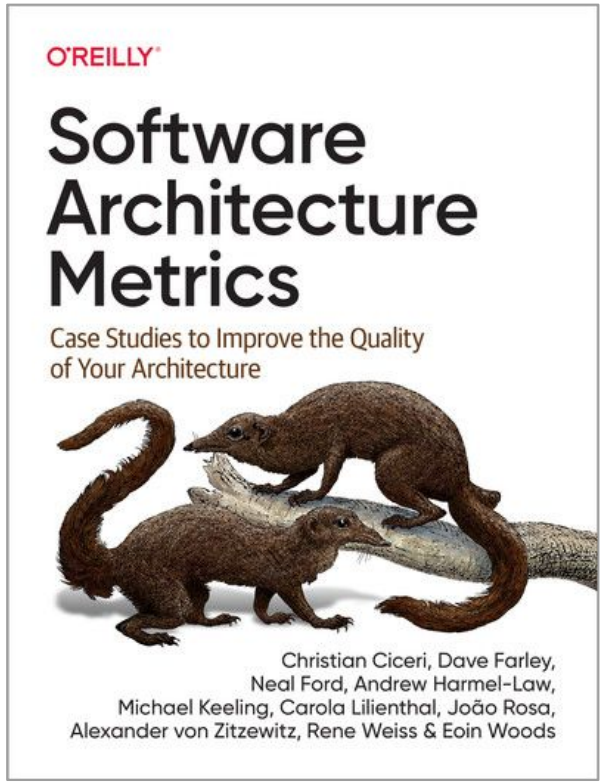

```
@AnalyzeClasses(packages = {"de.qaware.archunit.example.onion"})
public class OnionArchitectureFitnessTest {
    @ArchTest
    static final ArchRule onion_architecture_is_respected = onionArchitecture()
            .domainModels("..domain.model..")
            .domainServices("..domain.service..")
            .applicationServices("..application..")
            .adapter("cli", "..adapter.cli..")
            .adapter("persistence", "..adapter.persistence..")
            .adapter("rest", "..adapter.rest..");
}
```

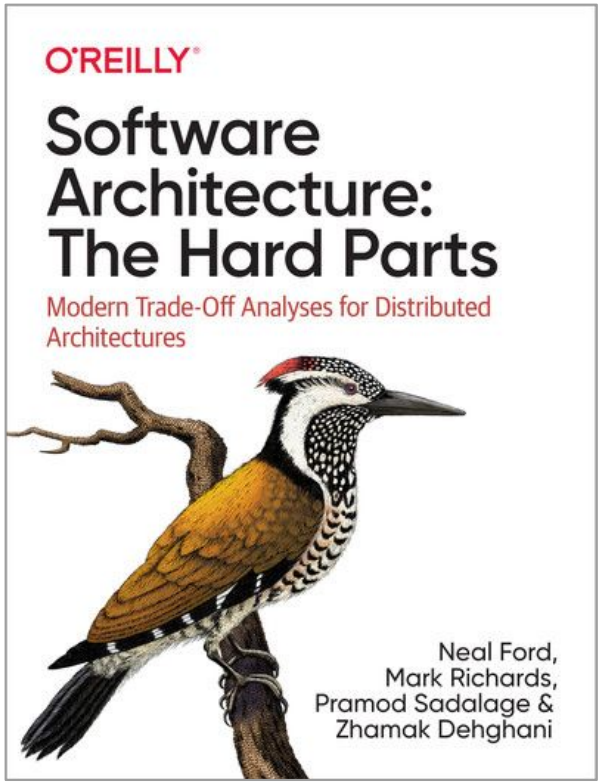
Beyond Clean Architecture.
Persönliche Leseliste für eine nachhaltig saubere Architektur.

QA|WARE









Upcoming Events

Meetup | Cloud Native Night

Couchbase @ CNN

München | 2.2.

Meetup | Clean Infrastructure as Code (IaC)

IaC without tests is like a broken window...

Online | 2.2.2023

Jfokus 2023

Quality in a Square.
K8s-native Quality Assurance of Microservices with Testkube

Stockholm | 6.2.2022

OOP 2023

Your APIs on Steroids: Retrofitting GraphQL by Code, Cloud-native or Serverless.

Online | 6.2.2022 | 11.00

Meetup | Codineers Rosenheim

Themenabend: Moderne APIs

Rosenheim | 9.3.

JavaLand 2023

Talks, Schulung & Aussteller in der Expo

Brühl | 21.-23.3.2023

Meetup | Shevelopers 3/23

SAVE THE DATE!

Details noch nicht bekannt

München | 29.3.2023

Meetup | Cloud Native Night

Terraform is not enough!
#pluginFramework #k8s #operator

München | 30.3.

Cloud Native Night Meetups

Weitere Meetups

Messen & Konferenzen

qaware.de

QA|WARE
SOFTWARE ENGINEERING

Meine Kontaktdaten ...

QAware GmbH
Aschauer Straße 32
81549 München
Tel. +49 89 232315-0
info@qaware.de

twitter.com/qaware
linkedin.com/company/qaware-gmbh
xing.com/companies/qawaregmbh
slideshare.net/qaware
github.com/qaware