Modern Software Development

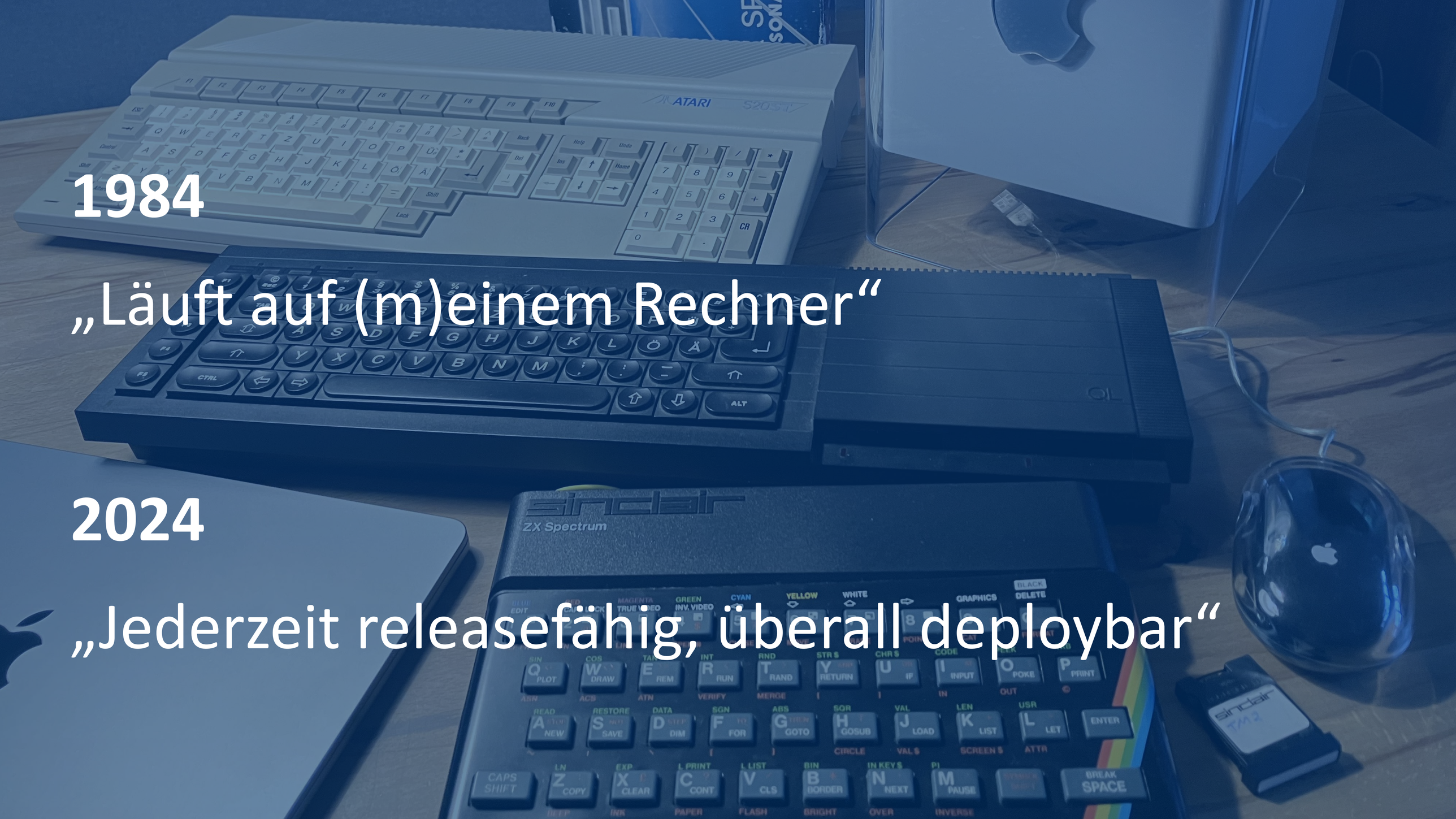
Stand der Dinge

Thomas Much

@thmuch

10. September 2024, Hannover


@thmuch
Technical
Agile
Coach
ATARI
ZX Spectrum
Die
Techniker
www.tk.de/IT



1984

„Läuft auf (m)einem Rechner“

2024

„Jederzeit releasefähig, überall deploybar“

„Wir müssen schneller werden“

Jason Gorman
3 Tage

Here's how I see it: if dev teams have the capability to reliably release working software at any time, and can sustain that for years, then being agile becomes entirely a business decision.

But if the dev team can't, then it makes little difference if the business decides to be agile, because it ain't gonna happen.

https://www.linkedin.com/posts/jasongorman_heres-how-i-see-it-if-dev-teams-have-the-activity-7123230332316758016-ZmfG

Wie

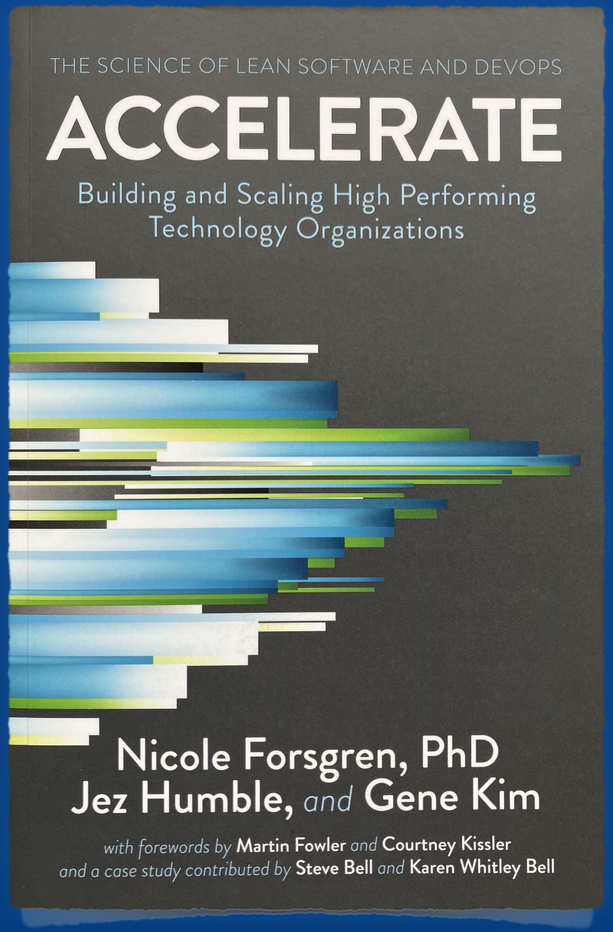
kann **nützliche** Software

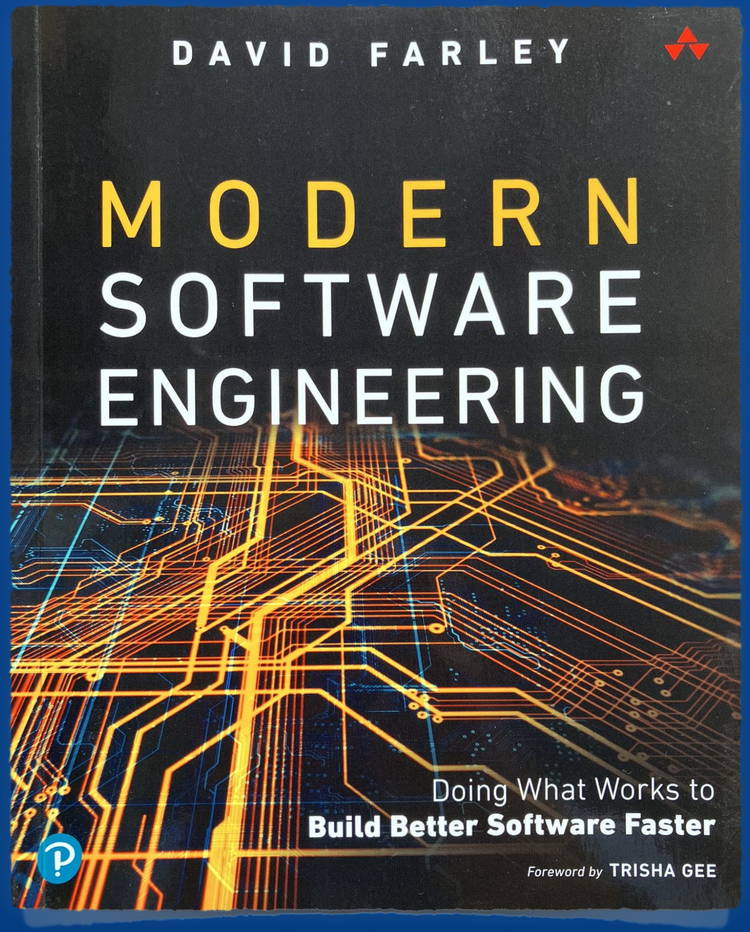
mit **guter** Qualität

zügiger

zum **Kunden**

ausgeliefert werden?


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

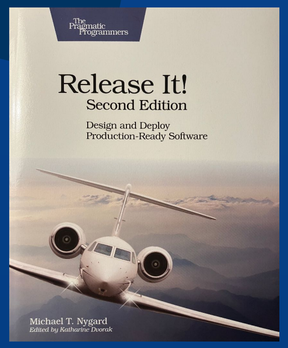

DAVID FARLEY
MODERN SOFTWARE ENGINEERING
Doing What Works to
Build Better Software Faster
Foreword by TRISHA GEE

2001

Manifest für agile Softwareentwicklung

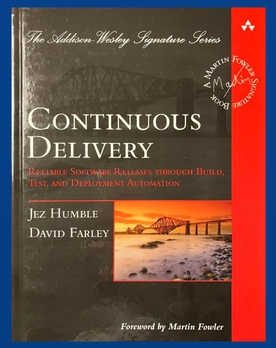
1999-2008

Diverse Bücher über Refactoring, Pragmatic Programming, Extreme Programming, Clean Code, Unit-Testing, Software Craftsmanship etc.



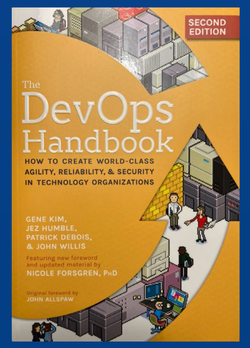
2007 (& 2018)

Lose gekoppelte Teilsysteme, Resilienz, Deploy-Automatisierung, Feature-Toggles, Trunk-Based Development

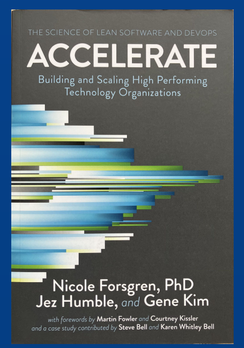


2010

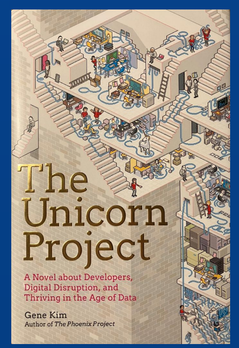
2013



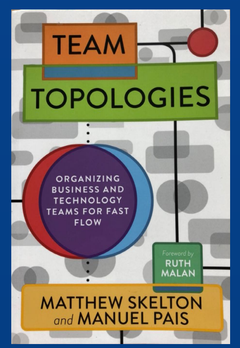
2016 (& 2021)



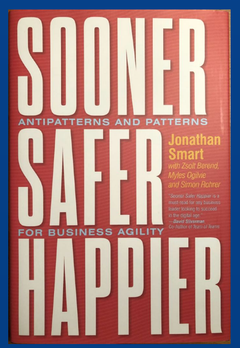
2018



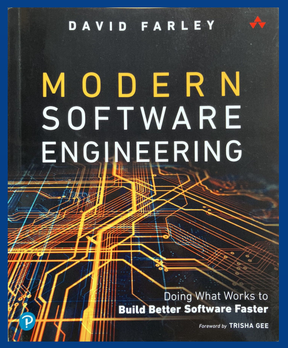
2019



2019



2020



2022

State of DevOps Report (seit 2012)

2010:

- Unternehmen releasen im Schnitt **4x pro Jahr**

2019:

- High-Performer releasen **täglich**
- Elite-Performer releasen **mehrfach täglich**

DORA Metriken

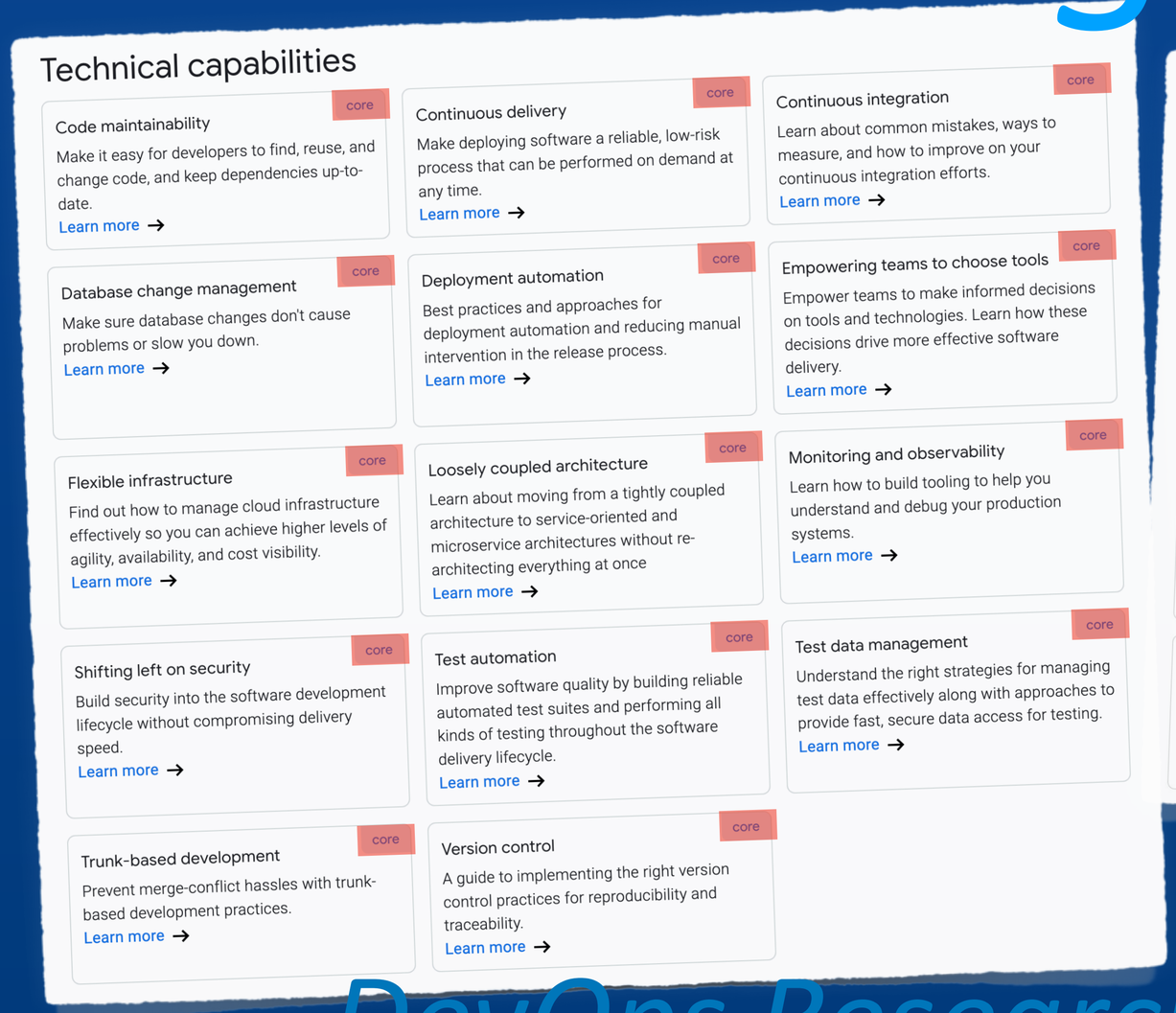
Korrelation ✅
Kausalität?? 🤔

Change Lead Time
Deployment Frequency
Change Failure Rate
Mean Time to Recovery

State of DevOps Report (seit 2012)

DevOps Research and Assessment

DORA Fähigkeiten



Technical capabilities

Code maintainability
core

Make it easy for developers to find, reuse, and change code, and keep dependencies up-to-date.

Learn more →

Continuous delivery
core

Make deploying software a reliable, low-risk process that can be performed on demand at any time.

Learn more →

Continuous integration
core

Learn about common mistakes, ways to measure, and how to improve on your continuous integration efforts.

Learn more →

Database change management
core

Make sure database changes don't cause problems or slow you down.

Learn more →

Deployment automation
core

Best practices and approaches for deployment automation and reducing manual intervention in the release process.

Learn more →

Empowering teams to choose tools
core

Empower teams to make informed decisions on tools and technologies. Learn how these decisions drive more effective software delivery.

Learn more →

Flexible infrastructure
core

Find out how to manage cloud infrastructure effectively so you can achieve higher levels of agility, availability, and cost visibility.

Learn more →

Loosely coupled architecture
core

Learn about moving from a tightly coupled architecture to service-oriented and microservice architectures without re-architecting everything at once

Learn more →

Monitoring and observability
core

Learn how to build tooling to help you understand and debug your production systems.

Learn more →

Shifting left on security
core

Build security into the software development lifecycle without compromising delivery speed.

Learn more →

Test automation
core

Improve software quality by building reliable automated test suites and performing all kinds of testing throughout the software delivery lifecycle.

Learn more →

Test data management
core

Understand the right strategies for managing test data effectively along with approaches to provide fast, secure data access for testing.

Learn more →

Trunk-based development
core

Prevent merge-conflict hassles with trunk-based development practices.

Learn more →

Version control
core

A guide to implementing the right version control practices for reproducibility and traceability.

Learn more →

Process capabilities

Customer feedback

Drive better organizational outcomes by gathering customer feedback and incorporating it into product and feature design.

Learn more →

Documentation quality
core

Maintain accurate, well-organized, user-centric internal documentation to empower teams throughout the software development process.

Learn more →

Monitoring systems to inform business decisions

Improve monitoring across infrastructure platforms, middleware, and the application tier, so you can provide fast feedback to developers.

Learn more →

Proactive failure notification

Set proactive failure notifications to identify critical issues and act on problems before they arise.

Learn more →

Streamlining change approval
core

Replace heavyweight change-approval processes with peer review, to get th

Team experimentation

Innovate faster by building emp

Visibility of work in the valu

Understand and visualize the f from idea to customer outcom drive higher performance.

Learn more →

Working in small batches

Create shorter lead times and fa loops by working in small batche common obstacles to this critica and how to overcome them.

Learn more →

Cultural capabilities

Generative organizational culture
core

Discover how growing a generative, high-trust culture drives better organizational and software delivery performance.

Learn more →

How to empower software delivery teams as a business leader

Measure and enable performance to help teams deliver value.

Learn more →

How to transform

Find out about the importance of ensuring your people have the tools and resources to do their job, and of making good use of their skills and abilities.

Learn more →

Job satisfaction

Find out about the importance of ensuring your people have the tools and resources to do their job, and of making good use of their skills and abilities.

Learn more →

Learning culture

Grow a learning culture and understand its impact on your organizational performance.

Learn more →

Transformational leadership

Learn how effective leaders influence software delivery performance by driving the adoption of technical and product management capabilities.

Learn more →

Well-being
core

A focus on employee happiness and work environment can improve organizational performance while helping retain talent.

Learn more →

DevOps Research and Assessment

https://dora.dev

Heute: **Auswahl**

Versionsverwaltung

Ziele:

Reproduzierbarkeit

Nachvollziehbarkeit

https://pixabay.com/de/photos/buchwand-b%C3%BCcher-mauer-dekorativ-1151405/

Versionsverwaltung

Dezentral (Blockaden & Wartezeit vermeiden) ✅

Alles versionieren!

Programm-Code

Infrastruktur-Code

Datenbank-(Struktur-)Änderungen

ALLES 🔥🎖️

https://pixabay.com/de/photos/buchwand-b%C3%BCcher-mauer-dekorativ-1151405/

Versionsverwaltung

Wie lange dauert es auf einem **neuen Rechner**,

mit den versionierten Dateien

das **Entwicklungstooling** aufzusetzen 🔥🏅

und das **Produkt** zu bauen & produktiv bereitzustellen?

Inkl. der ***exakten*** Versionen & Konfigurationsstände? 🔥🏅

Nicht Ergebnisse (Builds) aufbewahren, sondern neu bauen können!

Continuous Integration (CI)

„Wir haben einen CI-Server“ ❌

Ja, auch, aber …

https://pixabay.com/de/photos/f%C3%A4den-stillleben-n%C3%A4hen-garn-dekor-5296409/

Continuous Integration (CI)

Automatisierter Build-Prozess. Vollständig.

Jeder Commit triggert einen Build

Selbst-testender Code & Build

Jeder Commit triggert automatisierte, verlässliche, stabile Tests mit schnellem Feedback (wenige Minuten) 🔥🏅

Transparenz über den Zustand & Fortschritt

https://pixabay.com/de/photos/f%C3%A4den-stillleben-n%C3%A4hen-garn-dekor-5296409/

Continuous Integration (CI)

Problem beim Build & Test?

Fix bekommt **Priorität über jede andere Aufgabe** 🔥🥇

Mind. 1x pro Tag Änderungen *auf* Hauptentwicklungslinie (trunk/main/master) integrieren 🔥🔥🥇

Kleine, kohärente Commits 🔥🥇

😮

„Und unsere langlebigen Feature-Branches?“

Branching-Strategien

Nein. 🔥🔥🎖️🎖️

Nicht so lang. 🔥🎖️

Kurzlebig(st)e Branches

Trunk-Based Development?!

😮

„Wie verstecken wir dann unfertige Arbeit?“

Deployment und Release entkoppeln

Dark Launches

Features Toggles

Canary Releases

A/B Testing

https://www.flickr.com/photos/nasahqphoto/52503339196/in/album-72177720297400430/

Deployment und Release entkoppeln

Deployment jederzeit möglich:

Tagsüber

Häufig(er) – zumindest
in Vor-/Nicht-Prod-Systeme

Keine Release-Wochenenden nötig 🔥

https://www.flickr.com/photos/nasahqphoto/52503339196/in/album-72177720297400430/

Continuous Delivery (CD)

Ständig „releasen“ (deployen) *können*

Alle Schritte automatisiert (bis inkl. Prod) 🔥🥇

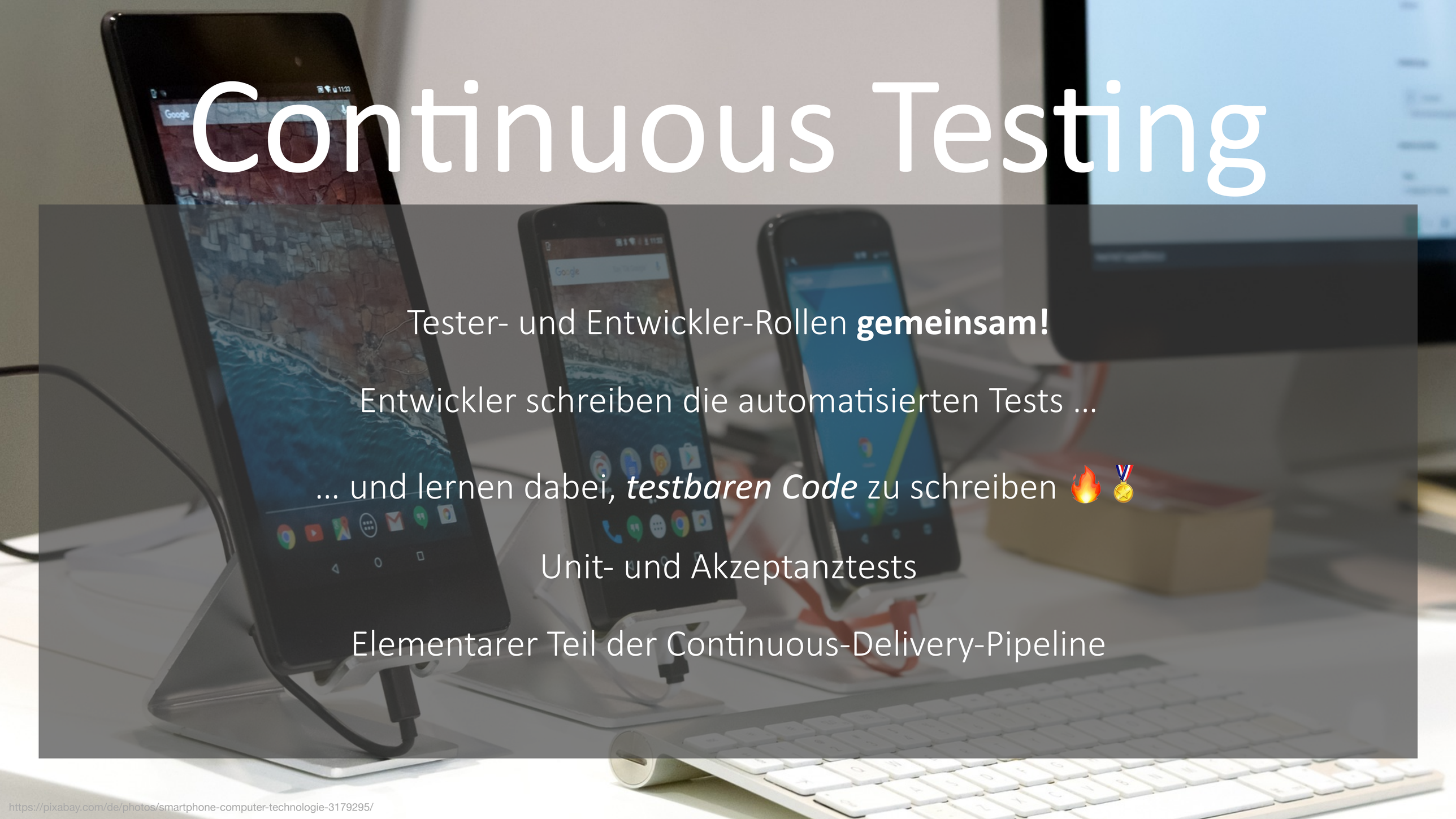
Nicht zwingend alles sofort automatisch auf Prod!

Baut auf vielen, vielen DORA-Capabilities auf

Viele Möglichkeiten zum Lernen & besser werden

DORA-Metriken geben Hinweis auf Qualität der Umsetzung

Continuous Testing



Tester- und Entwickler-Rollen **gemeinsam!**

Entwickler schreiben die automatisierten Tests …

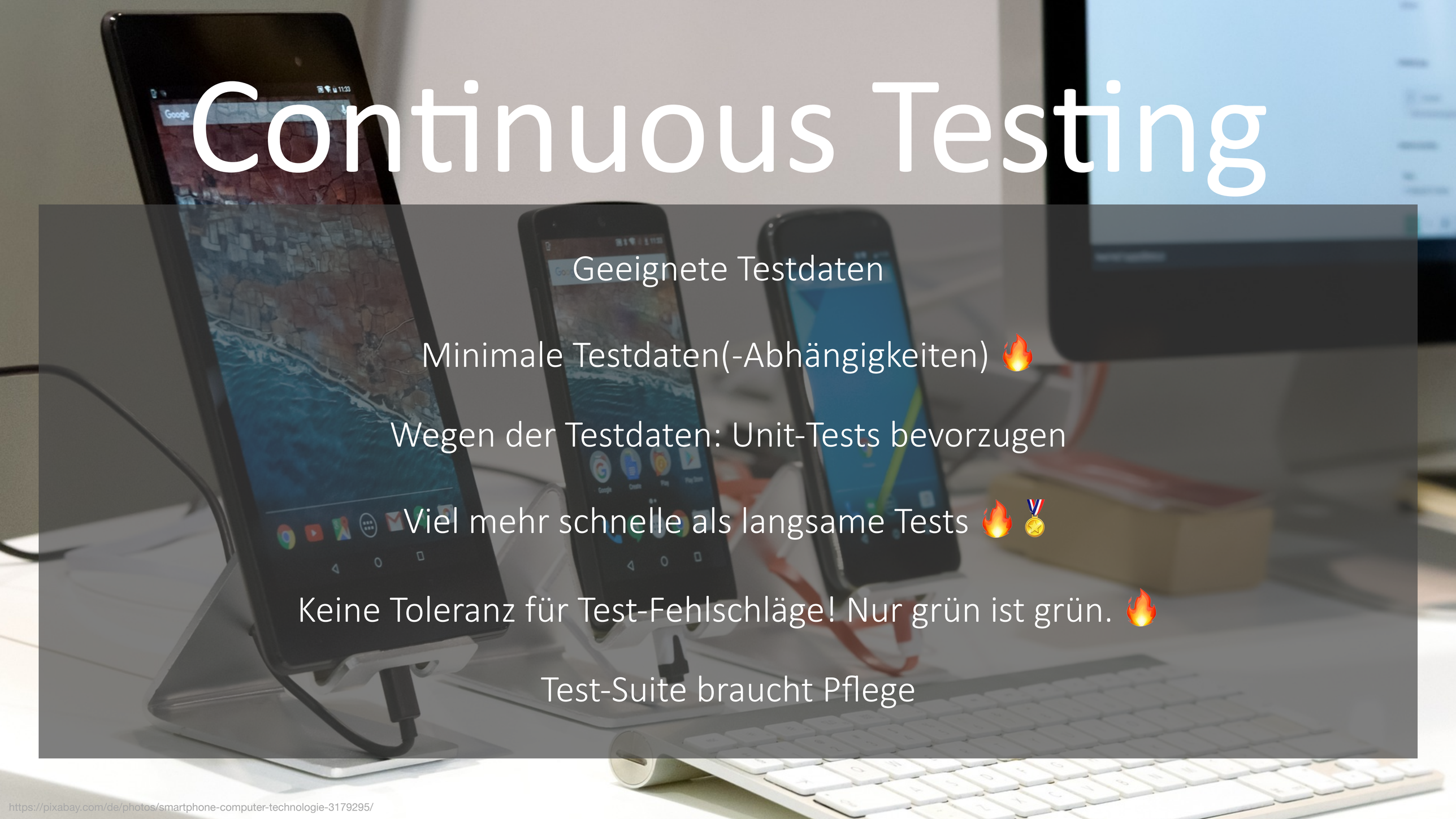
… und lernen dabei, ***testbaren Code*** zu schreiben 🔥🎖️

Unit- und Akzeptanztests

Elementarer Teil der Continuous-Delivery-Pipeline

https://pixabay.com/de/photos/smartphone-computer-technologie-3179295/

Continuous Testing



Geeignete Testdaten

Minimale Testdaten(-Abhängigkeiten) 🔥

Wegen der Testdaten: Unit-Tests bevorzugen

Viel mehr schnelle als langsame Tests 🔥🎖️

Keine Toleranz für Test-Fehlschläge! Nur grün ist grün. 🔥

Test-Suite braucht Pflege

https://pixabay.com/de/photos/smartphone-computer-technologie-3179295/

Build & Deployment

Architektur & Design

Modularisierung

Brought to you by the 1960s!

Kopplung

Kohäsion

Kontext!

https://pixabay.com/de/photos/steine-felsen-gro%C3%9F-und-klein-4628967/

Microservices

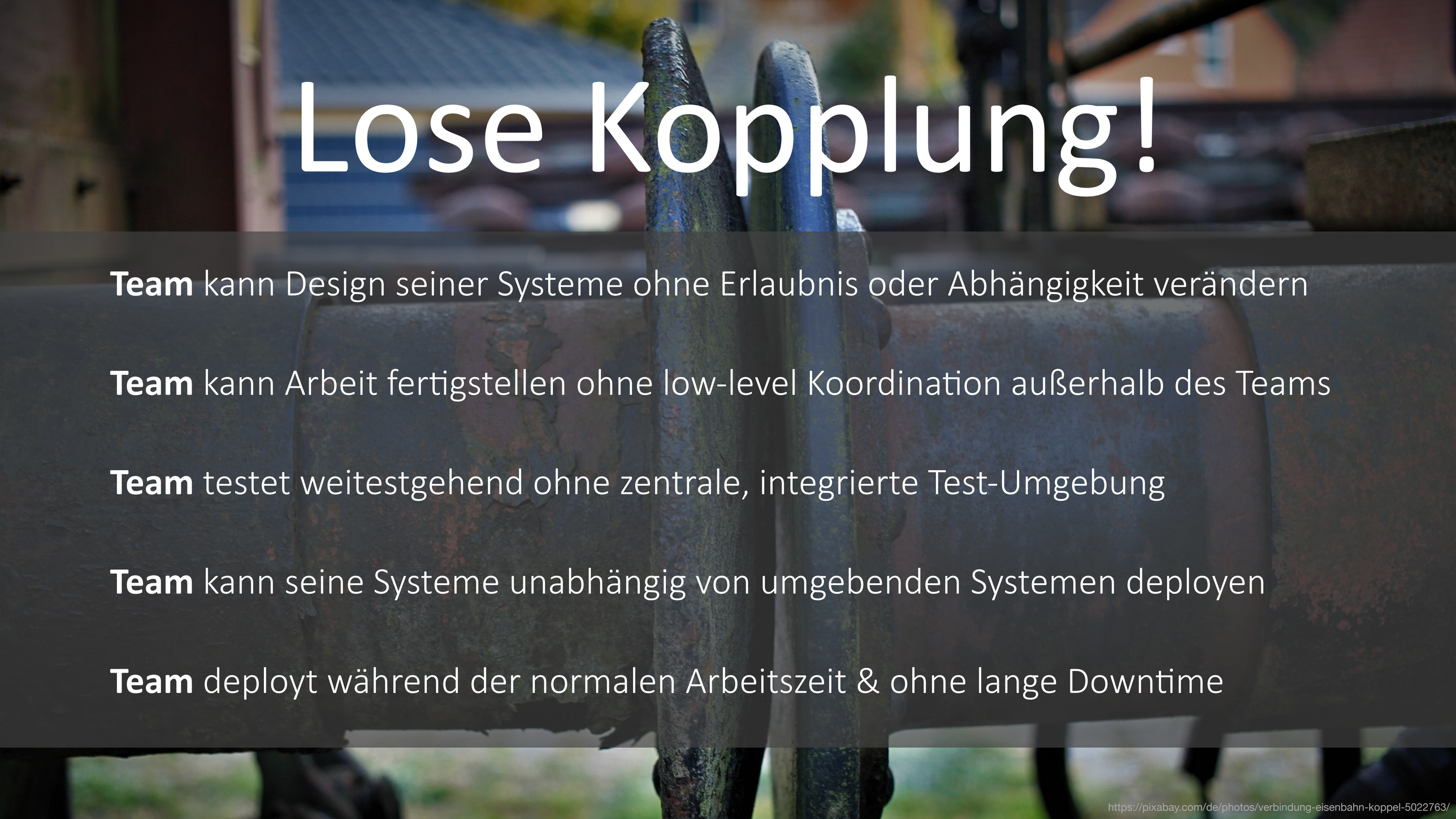
… sind nur ein Randthema.

Monolith (zu groß & schlecht strukturiert) ❌

Microservices?

Self-contained Systems (SCS)?

Modulith?



Lose Kopplung!

Team kann Design seiner Systeme ohne Erlaubnis oder Abhängigkeit verändern

Team kann Arbeit fertigstellen ohne low-level Koordination außerhalb des Teams

Team testet weitestgehend ohne zentrale, integrierte Test-Umgebung

Team kann seine Systeme unabhängig von umgebenden Systemen deployen

Team deployt während der normalen Arbeitszeit & ohne lange Downtime

https://pixabay.com/de/photos/verbindung-eisenbahn-koppel-5022763/

Lose Kopplung!

Autarke, resiliente (Teil-)Systeme

Kohäsion – (fachlicher) Kontext!

Keine Fehler-Kaskaden – kein „SOA“

Keine gleichzeitig notwendigen Deployments

Keine gleichzeitig notwendigen Changes vieler Entwickler in diversen Systemen

Dafür bewusste Dopplungen (Code, Daten)

(Un)Learn

SOLID (2000+)

Prinzipien

Single Responsibility

Open / Closed

Liskov Substitution

Interface Segregation

Dependency Inversion

OOD/OOP

CUPID (2020+)

Eigenschaften

Composable

Unix Philosophy

Predictable

Idiomatic

Domain-based

OO
FP
Daten-orientierung
Domänen-orientierung
…

Grund für diverse „solide“ Monolithen 😬

https://pixabay.com/de/photos/b%C3%BCcher-bibliothek-zimmer-schule-2596809/

(Un)Learn

Clean-Prinzipien?

Beispiel: **DRY** („Don’t Repeat Yourself“)

Wiederverwendung gerechtfertigt? Wiederholung zufällig?

DRY nur innerhalb eines Moduls!

Gezielte (Code- & Daten-)Dopplung für losere Kopplung!

https://pixabay.com/de/photos/b%C3%BCcher-bibliothek-zimmer-schule-2596809/

Wartbarkeit

Wieviel Code ist doppelt? Wieviel unbenutzt?

Wieviel % eurer Code-Basis können durchsucht werden?

Wie lange dauert eine Code-Änderung,
die ich nicht selber durchführen kann?

Wie lange dauert ein Änderung? Ein Emergency-Fix?

Wie aktuell sind die verwendeten Bibliotheken?

Wie häufig werden die aktualisiert? Wie lange dauert das?

Passende Werkzeuge

Team entscheidet

Passend zur Arbeitsweise & Aufgabe

Freiheit vs. totale Anarchie

Freiheit vs. Security

Basisangebot mit ausreichend Möglichkeiten

Aktuell halten & Ausnahmen ermöglichen

Freiheit & Sicherheit zum Ausprobieren (Lernen!)

https://pixabay.com/de/photos/schl%C3%BCssel-werkzeug-ausr%C3%BCstung-3013129/

Cloud?

Mittel zum Zweck

Ziel: flexible Infrastruktur

On-demand

Beliebig oft & reproduzierbar

Dev **& Prod**

Skalierbarkeit (falls nötig)

Handlungsbedarf? *Jetzt* anfangen!

„Die beste Zeit, einen Baum zu pflanzen, war vor 20 Jahren.

Die zweitbeste Zeit ist **jetzt**.“

https://pixabay.com/de/photos/eiche-natur-gr%C3%BCn-wald-setzling-4542800/

Kleine Schritte

Experimente

Schnelles Feedback

Lernen & Anpassen

Safe Space

Kontrolliertes (geringes) Risiko

Geringer „Blast Radius“

https://pixabay.com/de/photos/klettersteig-alpspitze-leiter-824376/

www.infoq.com/articles/replace-process-dogma-engineering/

www.puppet.com/resources/history-of-devops-reports

minimumcd.org

dora.dev

dannorth.net/cupid-for-joyful-coding/

www.continuous-delivery.co.uk

www.youtube.com/c/ContinuousDelivery

martinfowler.com/articles/continuousIntegration.html

scs-architecture.org

www.soonersaferhappier.com

trunkbaseddevelopment.com

tsvallender.co.uk/blog_posts/a-vision-of-continuous-integration

Vielen Dank 😊

Die Techniker

www.tk.de/IT

@thmuch