Ohne technische Skills kein Wandel:

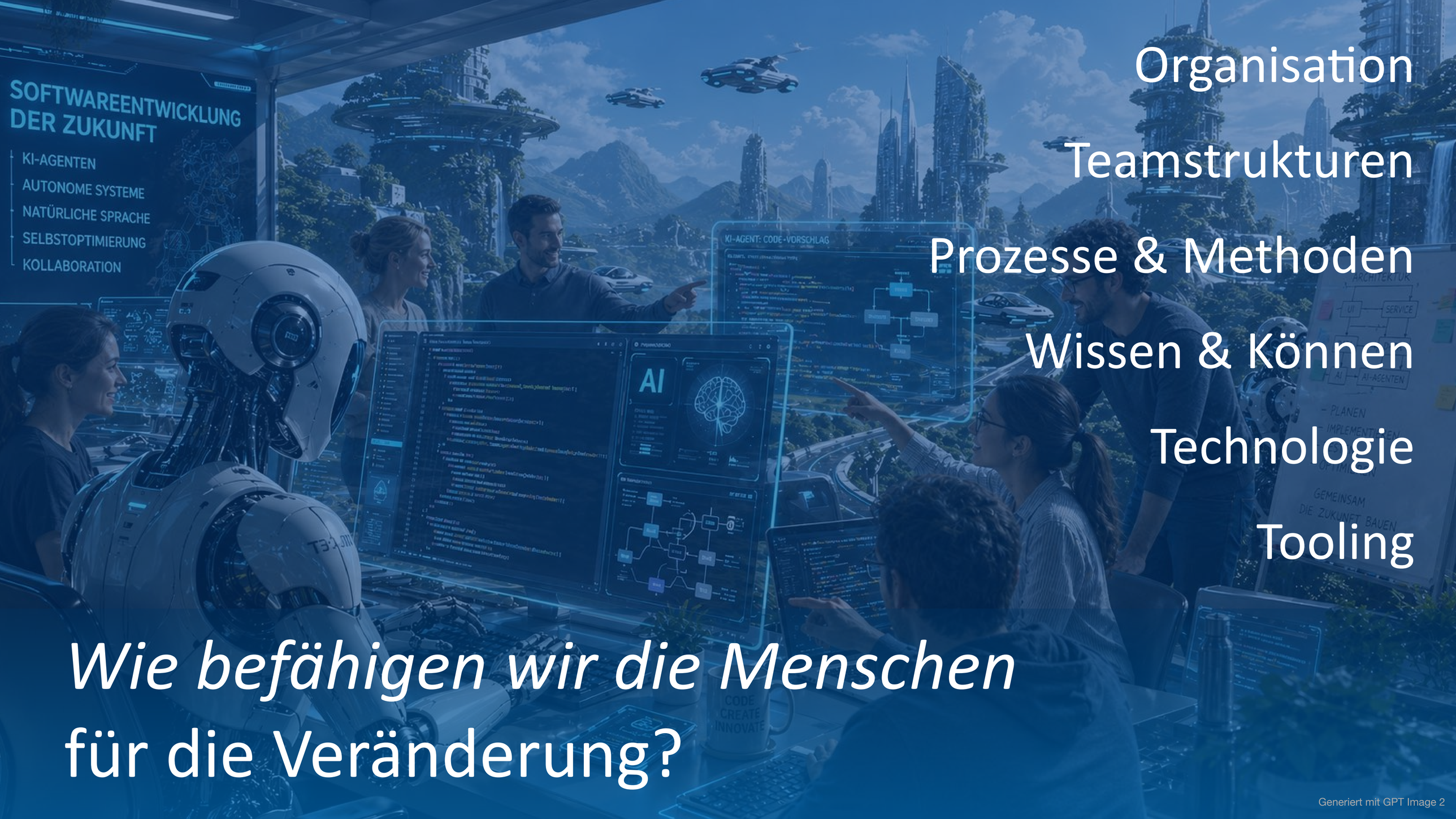
Technisches Coaching als Motor für Veränderung

Thomas Much, Techniker Krankenkasse

Scrum Masters HH XXL 2nd ed.

21. Mai 2026

@thmuch



Organisation

Teamstrukturen

Prozesse & Methoden

Wissen & Können

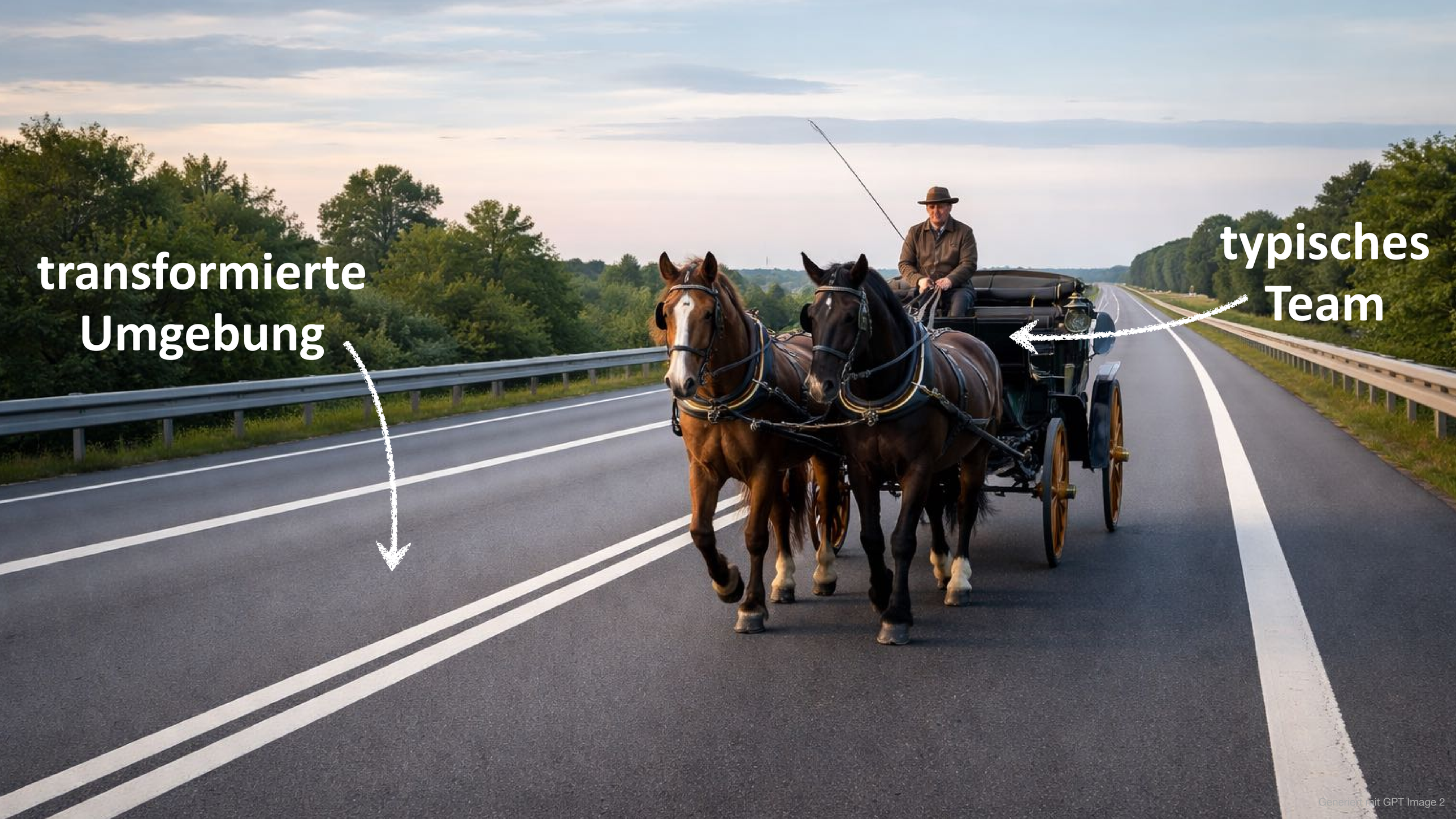
Technologie

Tooling

Wie befähigen wir die Menschen
für die Veränderung?

Generiert mit GPT Image 2

Generiert mit GPT Image 2


transformierte Umgebung
typisches Team
Generiert mit GPT Image 2

Generiert mit GPT Image 2

Was hat das mit uns zu tun?

Wo setzen Scrum & Co. an?

James Grenning • Follower:in

Wingman Software - Coaching and training in Agile technical pra...

3 Wochen •••

When we started doing XP immersion in late 1999, we started with the planning part of XP, the outer loop. People were confused and struggled with how to develop in small pieces.

Someone, probably Ron, Kent, or Bob suggested, let's start with the technical practices.

Mid week we'd introduced the planning practices... it was no big deal as developers could now envision small deliveries and incremental development.

Then Scrum took over the agile world and the struggle continued.

https://www.linkedin.com/feed/update/urn:li:activity:7443812568441212928/

Jason Gorman · 1.

Dev Trainer & Mentor | accelerating software delivery with and without A...

3 Wochen ·

I've watched a lot of Agile transformations fail over the last 25 years. The cause is commonly that these efforts make the mistake of thinking that the outer feedback loops of the development process drive the inner feedback loops.

Sorry folks. It's the other way around.

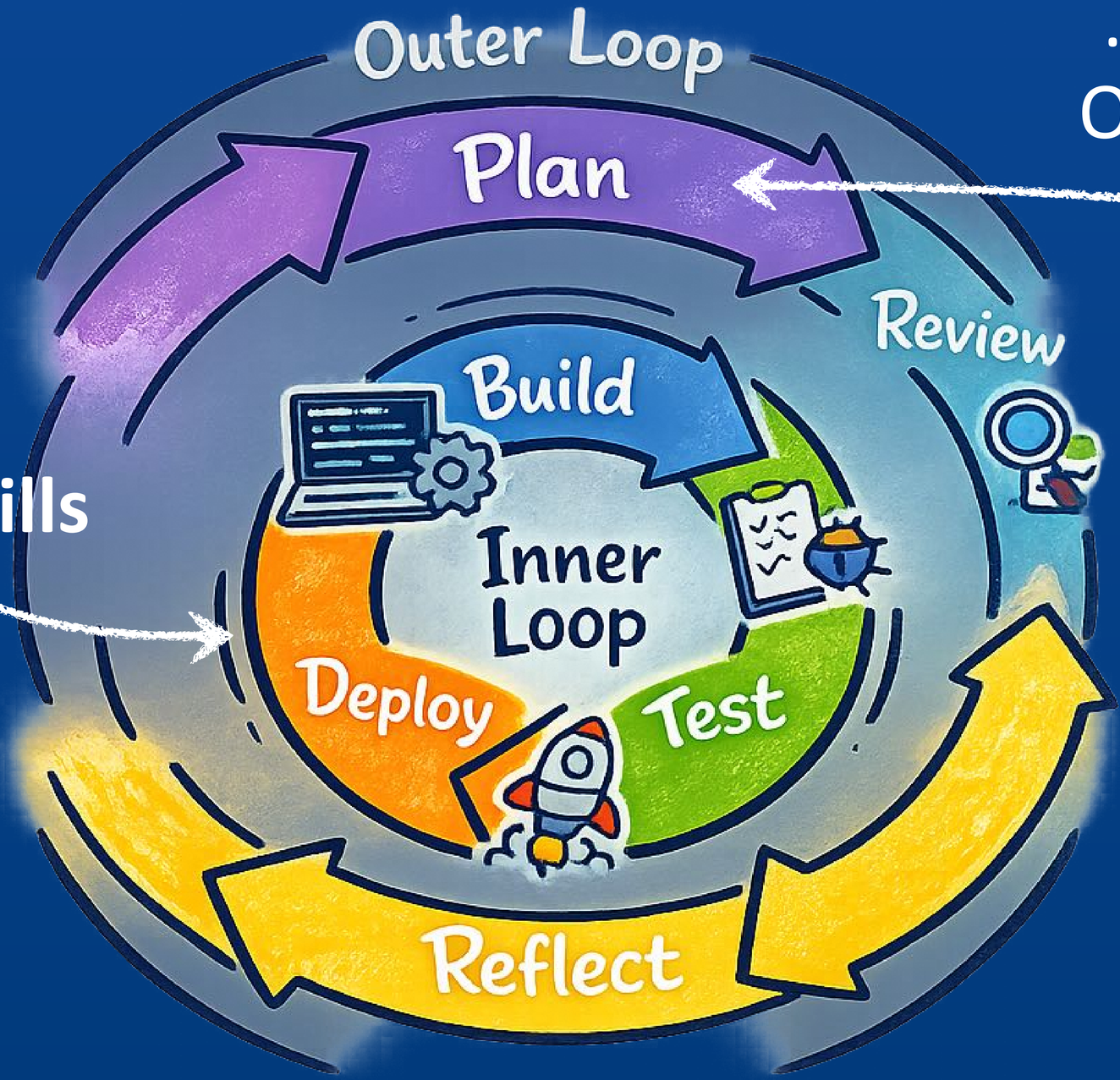
Plan as fast as you like. It won't make your build-test-release cycles any faster.

But make build-test-deploy faster, and hey presto! You can accelerate your plans.

Agility comes from the inside out.

https://www.linkedin.com/feed/update/urn:li:activity:7443812568441212928/

Das Team benötigt hierfür **technische Skills**



... sonst haben Optimierungen hier **wenig Wirkung**

Generiert mit GPT Image 2

Wenn Teams **nicht**
kontinuierlich Software entwickeln
und ausliefern können,

könnt ihr methodisch („agil“)
coachen, was ihr wollt.

Es wird nicht wirksam sein.

Technisches Coaching

holt Softwareentwickler:innen
in ihrer **Lebenswirklichkeit** ab:

Softwareentwicklung. Technik. Tools.

Technisches Coaching
fördert

Sorgfalt (eingebaute Qualität) & Disziplin

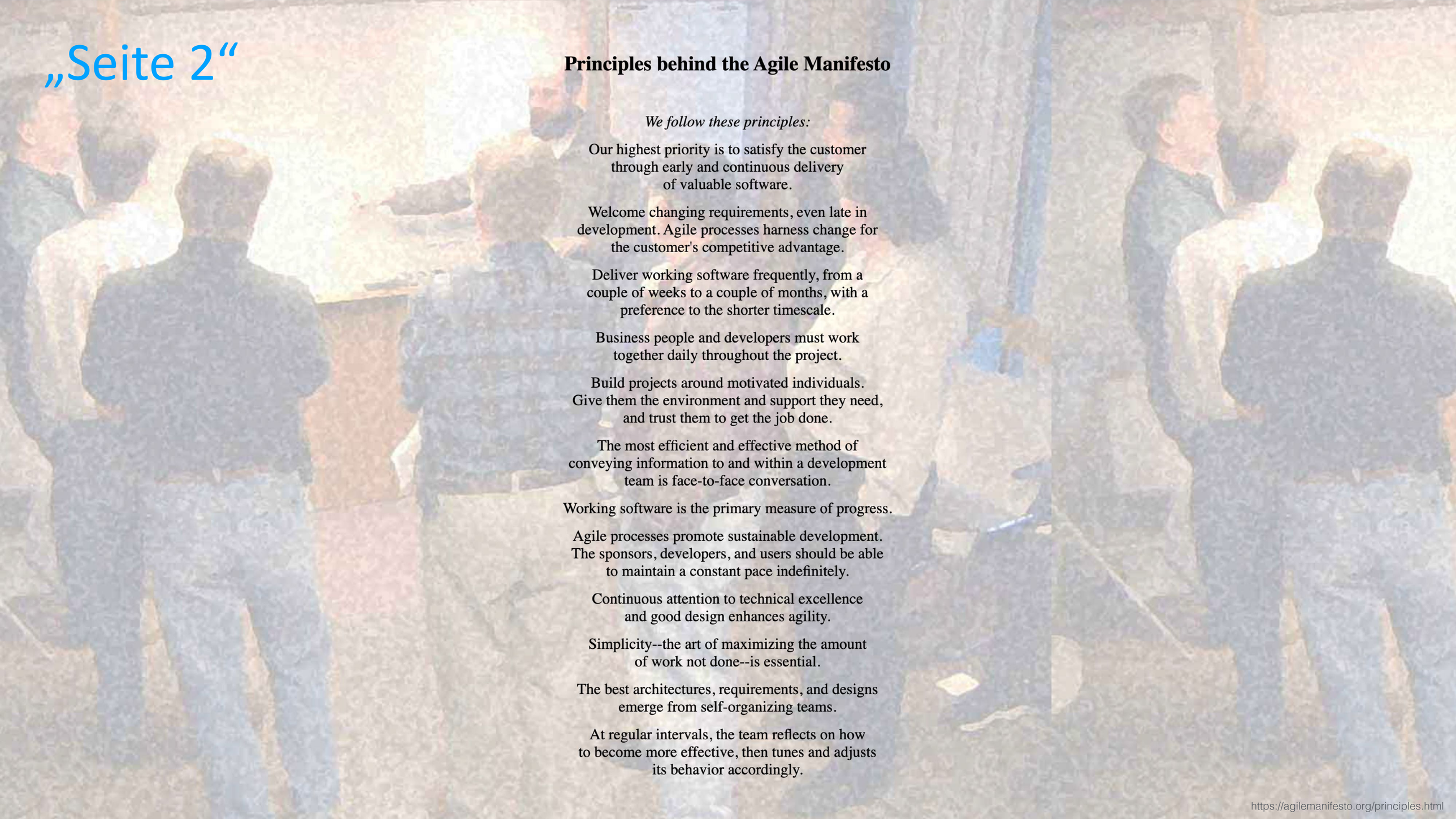
Schnelle (techn.) Feedback-Loops

Kontinuierlich Software entwickeln („Flow“)

Neugier & Lernen

… **Technische Exzellenz**

„Seite 2“



Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

https://agilemanifesto.org/principles.html

team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development.
The sponsors, developers, and users should be able
to maintain a constant pace indefinitely.

Continuous attention to technical excellence
and good design enhances agility.

Simplicity--the art of maximizing the amount
of work not done--is essential.

The best architectures, requirements, and designs
emerge from self-organizing teams.

At regular intervals, the team reflects on how

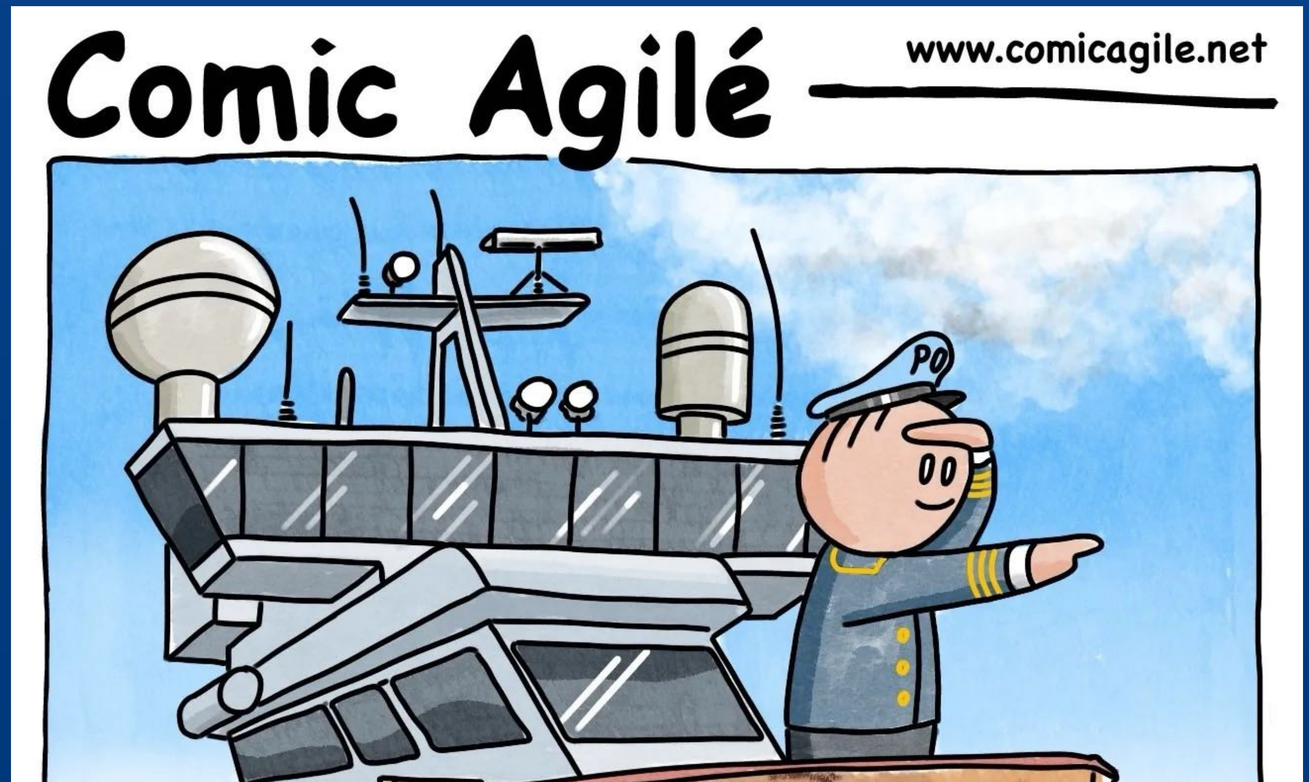
Technisches Coaching

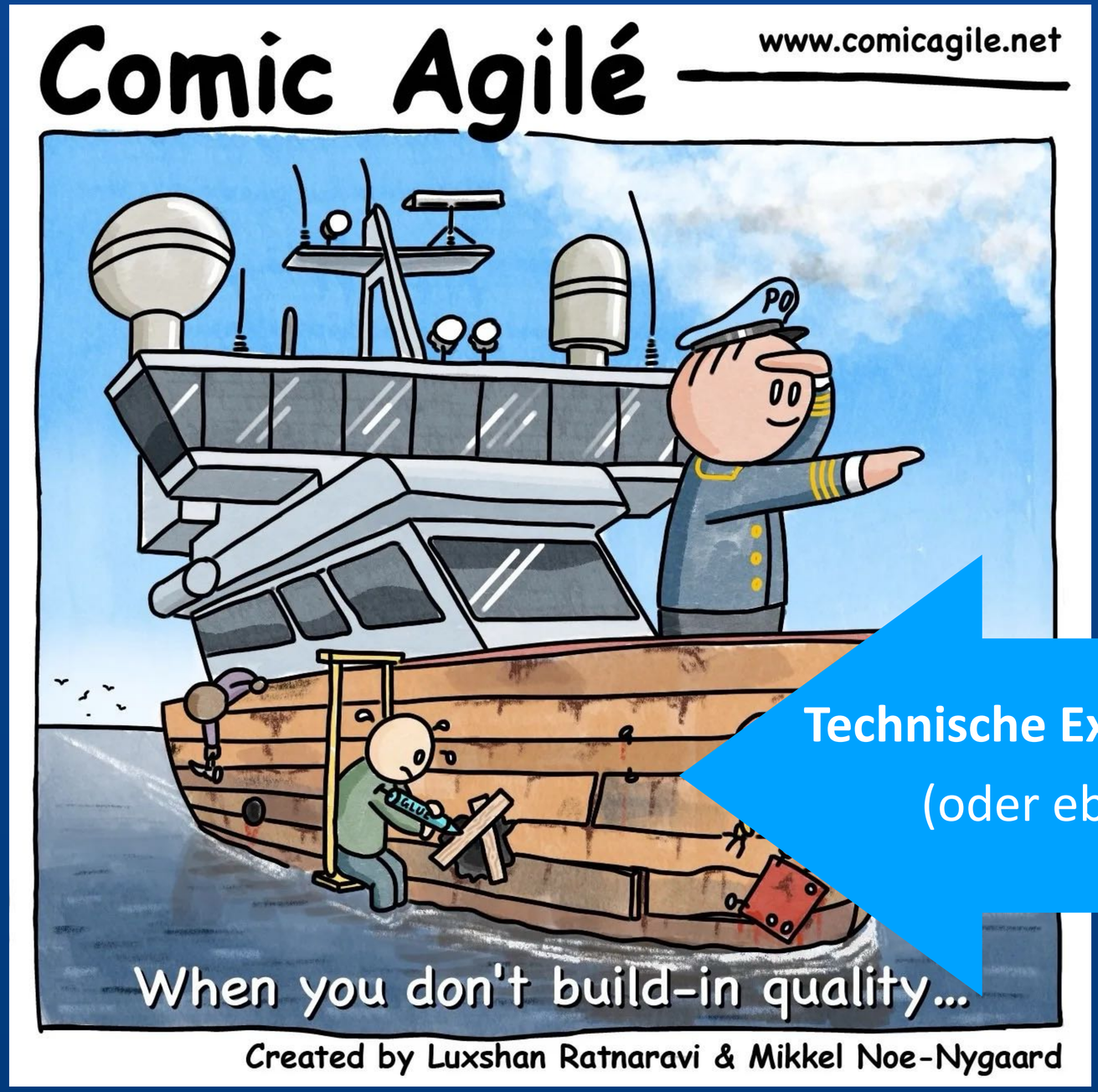
Coaching für **Technische Exzellenz***

***) Haltung / Mindset / Praktiken**

Technische Exzellenz?

https://www.comicagile.net/comic/features-vs-enablers/





Technische Exzellenz wird hier gelebt

(oder eben auch nicht 🙄)

https://www.comicagile.net/comic/features-vs-enablers/


Balance?!
Coding
Version Control
Tests
Design & Architektur
Performance
Security
CI/CD
Refactoring
TECHNISCHE SKILLS & PRAKTIKEN
Strategie & Ziele
Anforderungen verstehen
Stakeholder managen
Planung
Priorisierung
Risikomanagement
Prozesse
Kommunikation
Zusammenarbeit
METHODIK & ORGANISATION
Generiert mit GPT Image 2

Technical Coaching – Historie

- Pair-Programming
- Test-Driven Development
- Continuous Integration
- Planning & User Stories
- …
- **XP-Coach (technisch stark)**

Rollen & Fokus verschieben sich
auf **Prozess & Organisation**

Gegenbewegung zu
„Agile ohne Technik“:
Betonung von **Professionalität**
und **technischem Können**

frühe Wurzeln

1990er
Extreme Programming (XP)

2000er
Agiles Manifest & Skalierung

2010er
Software Crafting & DevOps

seit ca. 2015
Technical (Agile) **Coaching**

Agilität
+
Collaborative *

Architektur
+
Continuous *

Technical
Coach

Coden, Testen
+
Craft

Die Techniker

www.tk.de/IT

[ˈtoːmas] 👋

@thmuch

Viele Teams **scheitern nicht**
an fehlenden Meetings oder falschen Boards,

sondern an:

🤯 *schwer wartbarem Code*

⛈️ *fehlenden Tests*

🌋 *„technischen Schulden“*

😬 *Unsicherheit bei neuen Technologien*

Was macht ein Technical Coach?

-  **Codequalität** verbessern
-  **Engineering Practices** etablieren
-  **Architektur**entscheidungen begleiten
-  **Moderne Technologien** sinnvoll einführen
-  Hands-on unterstützen – oft **selbst mitentwickeln**

Die Rollen ergänzen sich!

Staff Engineer

⚠ Löst alles selber? Team lernt wenig

- Fokus: Technische Führung durch eigene Expertise
- Arbeitet *für* ein Team
- Skaliert als Multiplikator für Wissen & durch teamübergreifendes Arbeiten

Technical Coach

⚠ Agile Coaching ohne Technikbezug?

- Fokus: Menschen & Fähigkeiten & Arbeitsweise im Code
- **Arbeitet *mit* dem Team**, auch selber im Code
- Skaliert über *Lernen im Team*

oft Teil eines „Enabling Team“

Architekt

⚠ Elfenbeintumarchitektur? Passt nicht zur Realität

- Fokus: Struktur und Systemdesign
- Arbeitet oft vorausdenkend & langfristig
- Skaliert über Leitplanken & Entscheidungen

Manifesto for Agile Software Development

Präambel

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Twelve Principles of Agile Software

View Signatories

About the Authors
About the Manifesto

in einer
coachenden
Haltung

Manifesto for Agile Software Development

We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to valu

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on
the right, we value the items on the left more.

Coaching

Professionelle **Begleitung**

zur persönlichen oder beruflichen **Weiterentwicklung**

durch gezielte **Fragen**, **Feedback** und **Unterstützung**

Nicht „alles selber machen“ –
das müssen Technical Coaches
lernen & beachten!

Welche **Skills** coachen wir?

Was hilft uns beim Coaching?

Wie coachen wir?

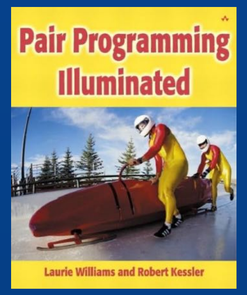
Skills

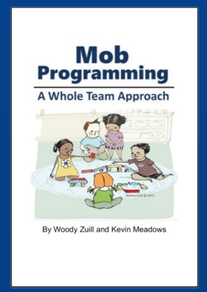
Fertigkeiten & Praktiken

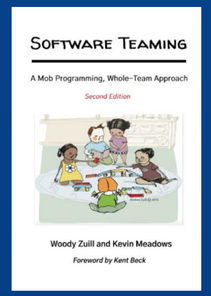
Skills

Klassische (XP-)Praktiken

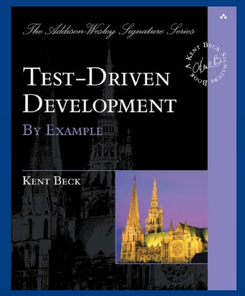
Dokumentiert zw. 1999-2002, kontinuierlich weiterentwickelt

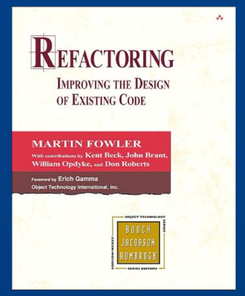




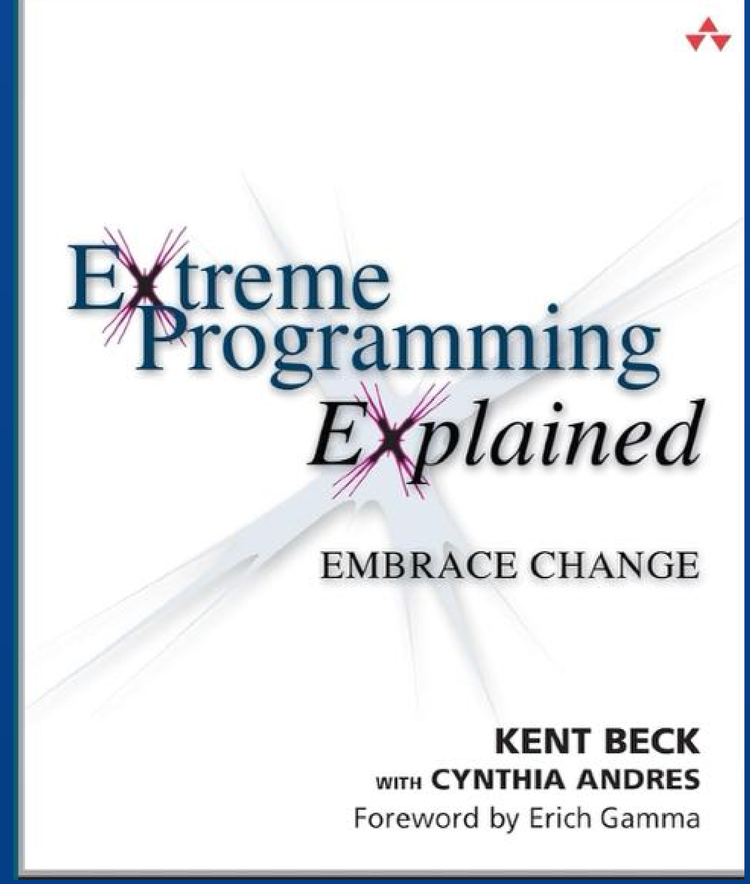


Pair- und Mob-/Team-Programming;
Zusammenarbeit mit dem Kunden

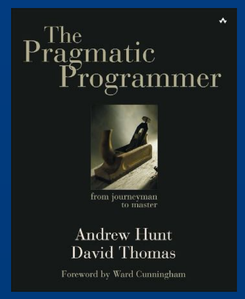


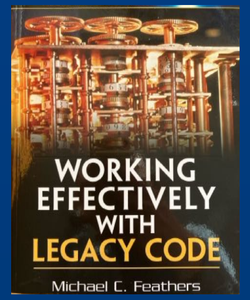


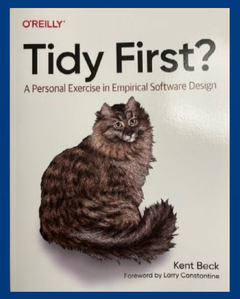
TDD inkl. Refactoring;
Arbeit in kleinen Schritten,
iterativ & inkrementell



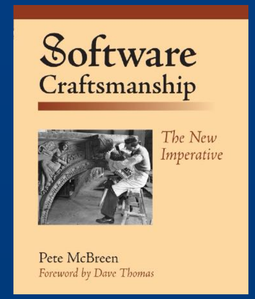
Berufsethos, Sorgfalt, Disziplin

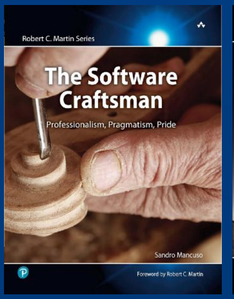


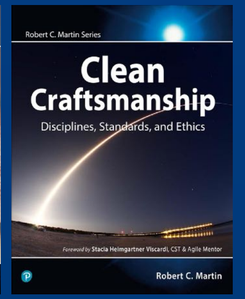




Pragmatic Programming
(„Good Enough Software“);
Aufgeräumter,
testbarer Code



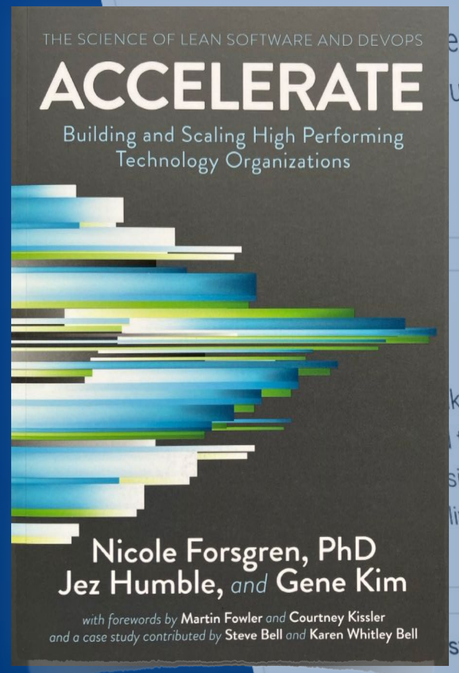


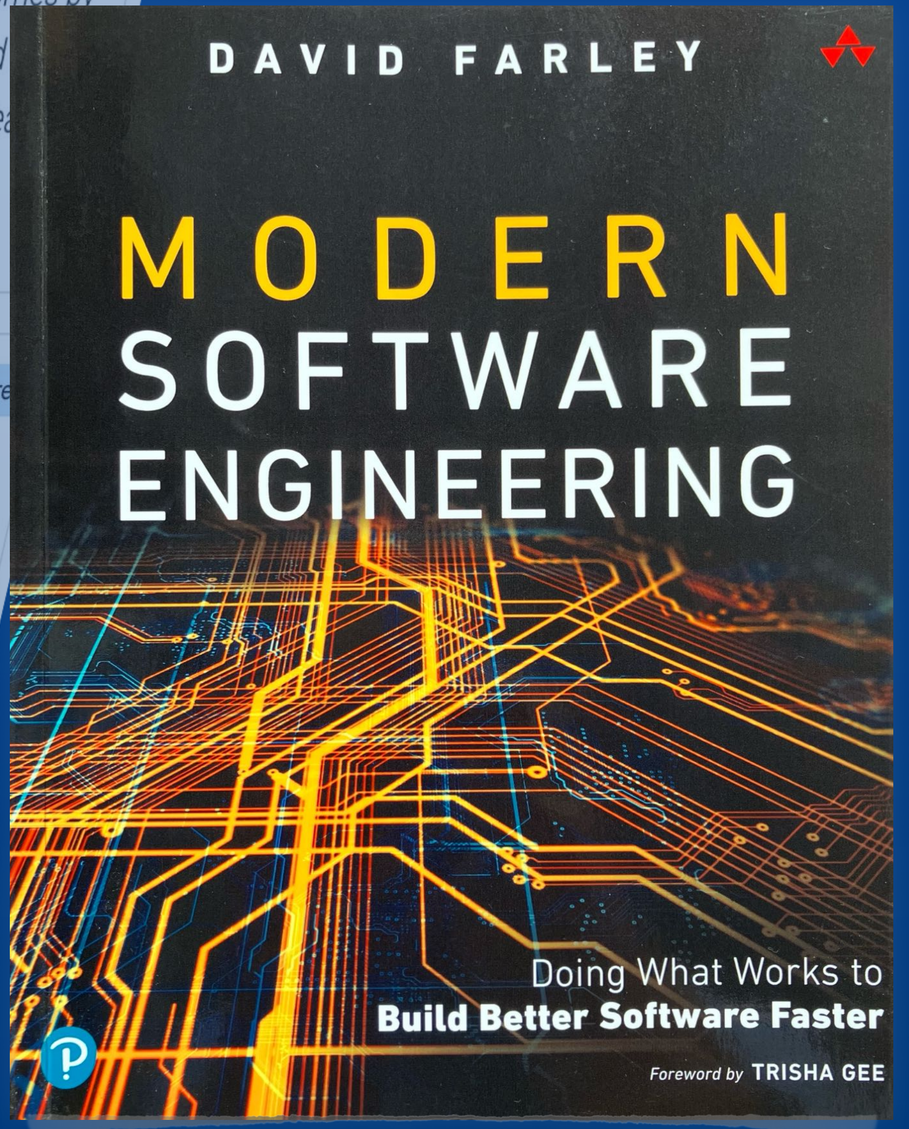



Noch viel, viel mehr ...
worauf Fokus legen?
ACCELERATE
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim
Release It!
Design and Deploy Production-Ready Software
Release It!
Second Edition
Design and Deploy Production-Ready Software
Michael T. Nygard
Continuous Delivery
Jez Humble
David Farley
Foreword by Martin Fowler
DAVID FARLEY
MODERN SOFTWARE ENGINEERING
Doing What Works to Build Better Software Faster
The Pragmatic Programmer
your journey to mastery
David Thomas
Andrew Hunt
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers
O'REILLY
Software Architecture: The Hard Parts
Modern Trade-Off Analyses for Distributed Architectures
Neal Ford,
Mark Richards,
Pramod Sadalage &
Zhamak Dehghani
Balancing Coupling in Software Design
Universal Design Principles for Architecting Modular Software Systems
Vlad Khononov
Extreme Programming Explained
EMBRACE CHANGE
KENT BECK
WITH CYNTHIA ANDRES
Foreword by Erich Gamma
Mob Programming
A Whole Team Approach
Software Teaming
A Mob Programming, Whole-Team Approach
Second Edition
Woody Zuill and Kevin Meadows
Foreword by Kent Beck
Building Microservices
Designing Fine-Grained Systems
Second Edition
Sam Newman
Technical Agile Coaching with the Samman method
Agile Software Development
Alistair Cockburn
Refactoring
Improving the Design of Existing Code
Second Edition
Martin Fowler
Kent Beck
Crafting Great APIs with Domain-Driven Design
Collaborative Craftsmanship of Asynchronous and Synchronous APIs
Annegret Junker
Fabrizio Lazzaretti
Apress
Collaborative Software Design
How to facilitate domain modeling decisions
Evelyn van Kelle
Gien Verschatse
Kenny Baas-Schwegler
Pair Programming Illuminated
Laurie Williams and Robert Kessler
Software Craftsmanship
The New Imperative
Pete McBreen
Foreword by Dave Thomas
Specification by Example
How successful teams deliver the right software
BDD in Action
Behavior-Driven Development for the whole software lifecycle
John Ferguson Smart
The Software Craftsman
Professionalism, Pragmatism, Pride
Sandro Mancuso
The Unicorn Project
A Novel about Developers, Digital Disruption, and Thriving in the Age of Data
Gene Kim
Author of The Phoenix Project
Tidy First?
A Personal Exercise in Empirical Software Design
Kent Beck
Foreword by Larry Constantine
"Looks Good To Me"
Adrienne Braganza
Refactoring
Improving the Design of Existing Code
Martin Fowler
With contributions by Kent Beck, John Brant, William Opdyke, and Don Roberts
Foreword by Erich Gamma
Object Technology International, Inc.
Booch Jacobson Rumbaugh
Test-Driven Development
By Example
Kent Beck
The Pragmatic Programmer
from journeyman to master
Andrew Hunt
David Thomas
Foreword by Ward Cunningham
Contract Testing IN ACTION
With Pact, PactFlow, and GitHub Actions
Team Topologies
Matthew Skelton and Manuel Pais
Foreword by Ruth Malan
The DevOps Handbook
How to Create World-Class Agility, Reliability, & Security in Technology Organizations
Foreword by Robert C. Martin
SECOND EDITION
The Phoenix Project
A Novel About IT, DevOps, and Helping Your Business Win
REVISED WITH NEW RESOURCE GUIDE
SOONER SAFER HAPPIER
ANTIPATTERNS AND PATTERNS
Jonathan Smart
EXTRAORDINARILY BADASS AGILE COACHING
THE JOURNEY FROM BEGINNER TO MASTERY AND BEYOND
ROBERT L. GALEN

DORA-Capabilities

dora.dev/capabilities/





Modern Software Engineering

https://dora.dev/capabilities/

DORA-Capabilities ❤️ AI

7 von 34 Capabilities (2026)

Platform engineering
AI

A platform provides the automated, secure pathways that allow AI's benefits to scale across the organization.

Learn more →

AI-accessible internal data
AI

Connecting AI to your internal documentation and codebases moves it from a generic assistant to a specialized expert.

Learn more →

Clear and communicated AI stance
AI

Ambiguity creates risk. A clear policy provides the psychological safety needed for effective experimentation.

Learn more →

Working in small batches
core AI

Create shorter lead times and faster feedback loops by working in small batches, counteracting the risk of instability as AI accelerates development.

Learn more →

Healthy data ecosystems
AI

The benefits of AI are significantly amplified by high-quality, accessible, and unified internal data.

Learn more →

User-centric focus
AI

A focus on user needs is essential to ensure that AI-accelerated teams are moving quickly in the right direction.

Learn more →

Version control
core AI

Version control ensures reproducibility and traceability, serving as a critical safety net as AI accelerates the velocity of change.

Learn more →

https://dora.dev/capabilities/

Technisches Coaching

Coaching für **Continuous Delivery***

***) Haltung / Mindset / Praktiken**

Wie lernen Menschen Neues?

aka „Auf dem Weg zu Wissen & Können“

Shu Ha Ri

Ab wo ist welche **Haltung** sinnvoll?

Coaching

Mentoring

Training

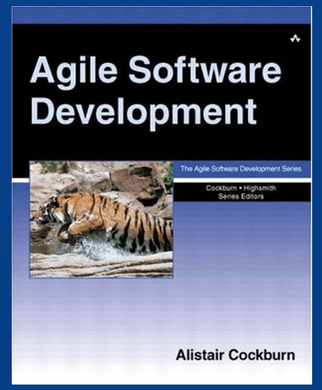
守 破 離

Regeln befolgen

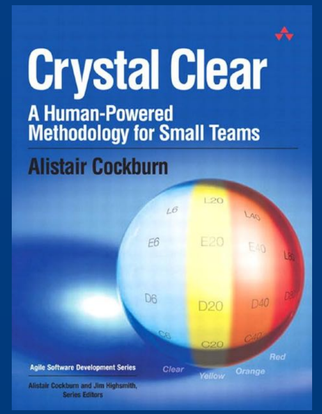
Regeln bewusst brechen

keine Regeln mehr benötigen

2002



2005



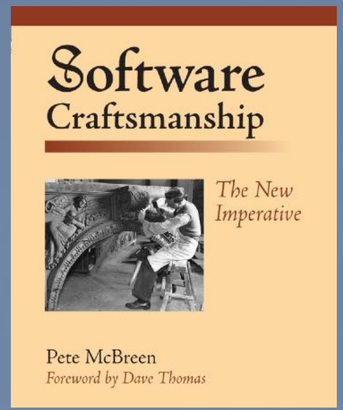
Apprentice – Journeyman – Master

Lehrling – Geselle – Meister

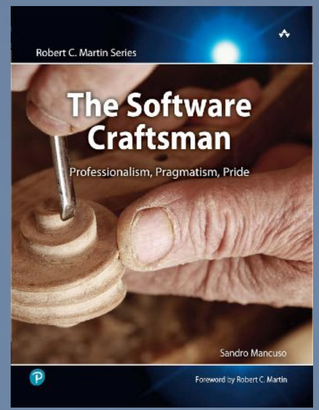
Brauchen Ausbildung.
Dürfen nicht allein gelassen werden!

Welche **Skills** müssen gelernt werden?
Z.B. Code-Smells erkennen & beseitigen,
Refactoring …
Welche **Übungen (Katas)** passen?

2001



2014



https://unsplash.com/photos/brown-wooden-cat-figurine-on-white-surface-XHXOafBIOhc

Technisches Coaching

Coaching für
Lernen & Können *im Dev-Team*

Stufen des Lernens & Könnens

Nur ein **Beispiel** von ganz vielen Modellen

Auf welcher Stufe befindet sich jede:r Einzelne:r?

Bzgl. welcher Themen?

„Vier Stufen der Kompetenzentwicklung“ von Noel Burch

Was hilft?

Coaching-Modelle

Modelle

Vereinfachte Darstellungen oder Konzepte,

die helfen, **komplexe Zusammenhänge**

zu **verstehen**, zu **erklären** oder zu **steuern**

„All models are wrong

but some are useful“

– angelehnt an George E.P. Box

https://en.wikipedia.org/wiki/All_models_are_wrong

Vereinfachungen!

„Remember that
all models are wrong;

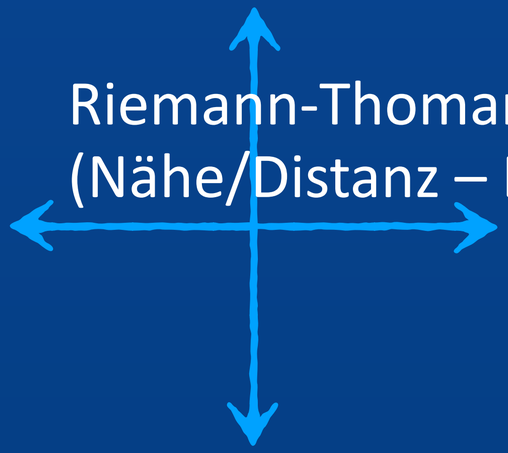
the practical question is
how wrong do they have to be
to not be useful.“

– George E.P. Box

Wie fehlertolerant können/wollen wir in einem bestimmten Kontext sein?

https://medium.com/swlh/all-models-are-wrong-does-not-mean-what-you-think-it-means-610390c40c9c

Bekannte Coaching-Modelle (Auswahl)



Riemann-Thomann
(Nähe/Distanz – Dauer/Wechsel)

SCARF
(Status,
Certainty,
Autonomy,
Relatedness,
Fairness)

Tuckman
(Forming, Storming, Norming,
Performing, Adjourning)

Hackmans
Autoritätsmatrix

Burch
(4 Stufen der
Kompetenzentwicklung)

Lencioni
(5 Dysfunctions
of a Team)

Immer dran denken:

Vereinfachungen. **Schubladen**. Modelle 😉

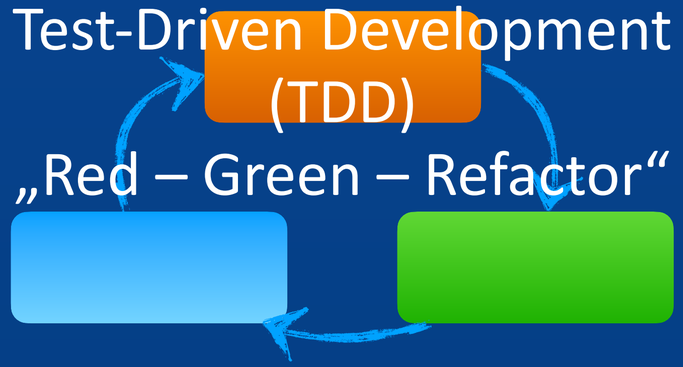
Aber evtl. nützlich?!

Generiert mit GPT Image 2

Modelle für Technical Coaching (Auswahl)

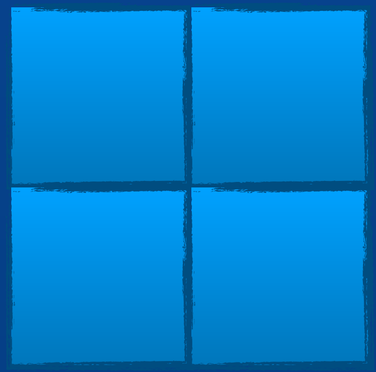
Test-Driven Development (TDD)

„Red – Green – Refactor“

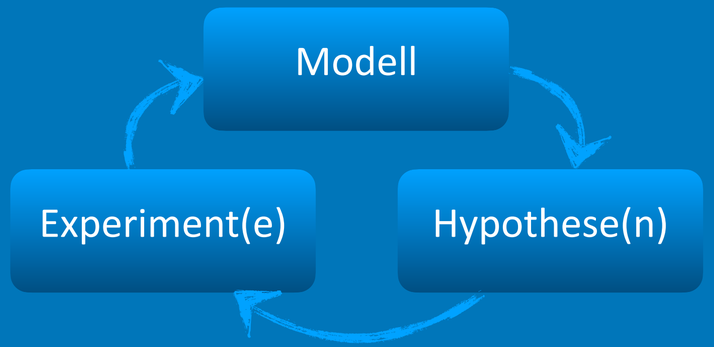


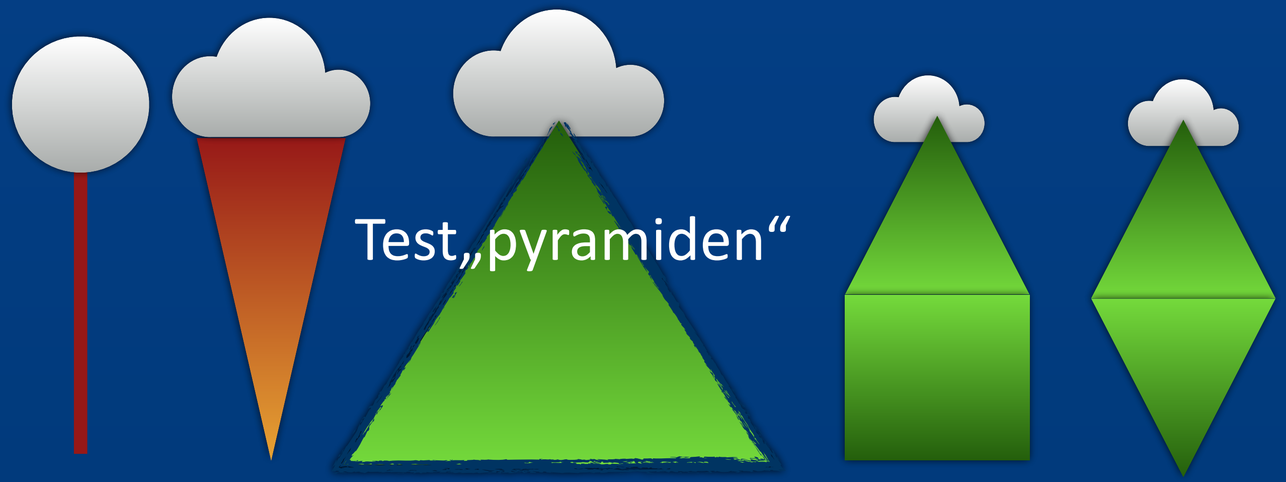
… auch in Legacy Code

Technische Schulden



Auch technische Modelle lassen sich validieren!

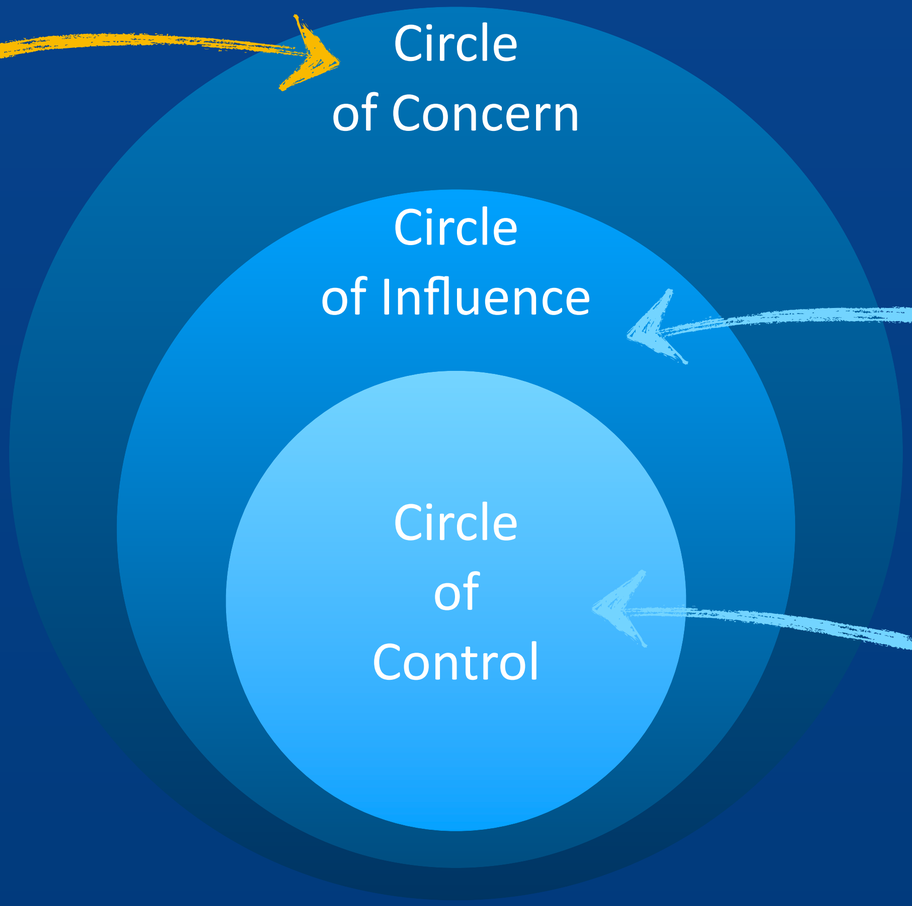




Test„pyramiden“

Grenzen für Veränderung

Quelle für **Frust** –
bei Veränderungsversuchen
ohne Wirksamkeit



Wen muss das Team
für sein Veränderungsanliegen
gewinnen?

Was kann vom Team
überhaupt
verändert werden?

angelehnt an Stephen Coveys „Circle of Concern & Circle of Influence“

Wie

Vorgehensmodelle

“Structured Approaches“

coachen?

Modelle

(Team-)**Dynamik**

Phasen

Reife(grad)

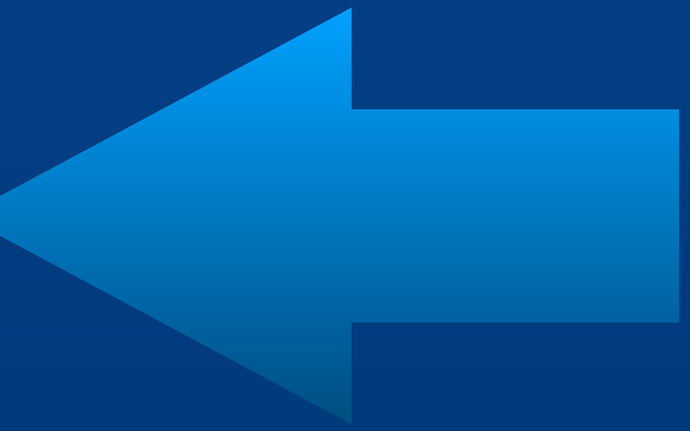
Diagnose

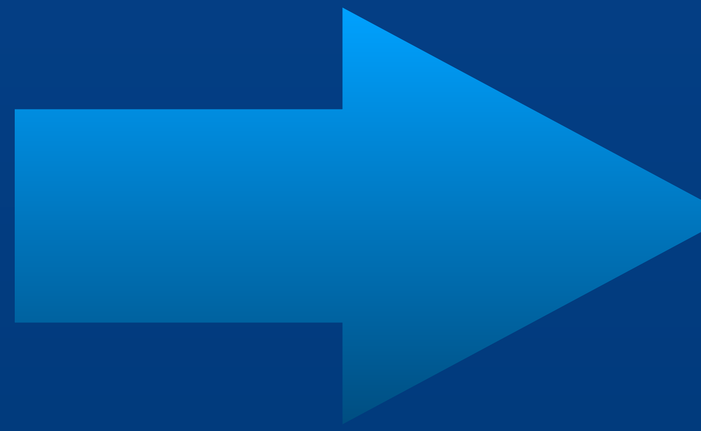
Qualität

Zustand

Coaching-**Prozess**

Methodischer **Ablauf**





(Technische) Team-Begleitungen

Auftragsklärung

Wer ist **Auftraggeber?**

Wann ist die Begleitung ein **Erfolg?**

Reflexion mit Auftraggeber

Modus i.d.R. **Pair- und Team-Programming**

Coach **arbeitet mit,** beobachtet und reflektiert

Kick-off

1-2 Wochen Team-Begleitung

optional

Zwischenstand

1-2 Wochen Team-Begleitung

Abschluss

Begleitung bei **normalen Team-Aufgaben.**

Code-Katas nur bei Bedarf.

Workshops mit Team-Programming

Auftragsklärung

„Team-Programming“

optional verteilt auf zwei halbe Tage

i.d.R. in **Präsenz**

z.B. 1x **Remote-Tools** üben

Erwartungshaltung & Grundlagen

2h Team-Programming mit **Code-Kata**

Mini-Retro

2,5h Team-Programming mit **Code-Kata**

Abschluss

„Coding-Skills“

i.d.R. 1 Tag

Nach Möglich-keit in Präsenz

Etwas **Theorie** & gemeinsames **Üben** (im Team-Programming mit **Code-Katas**)

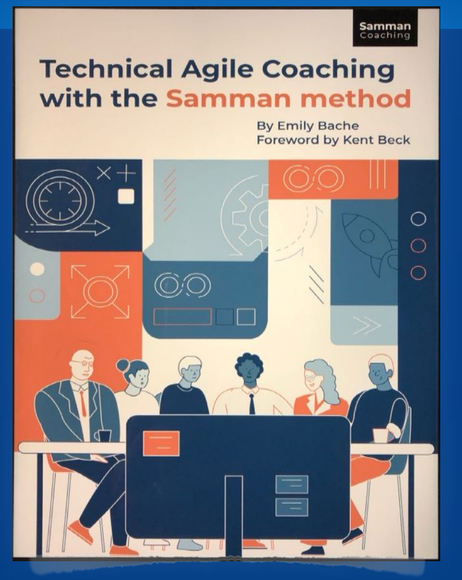
Mini-Retro

Gelerntes am **Produktivcode** umsetzen (Pair- & Team-Programming)

Abschluss

Samman Coaching – Lernen skalieren

🇸🇪 „zusammen“



Auftragsklärung

Kickoff-Workshop:
Team kennenlernen

Chartering-Workshop:
Entscheidung für/gegen
Coaching & **Themen**

Auswahl aus Learning-Hour-**Sammlung**

evtl. für 2-3 Teams gemeinsam

1h **Learning Hour** mit Code-Kata

2h **Ensemble Working** am eigenen Code

9x

10.

1h **Inspirational Demo** (evtl. Video)

https://sammancoaching.org/

Code Retreats etc.

Evtl. gemeinsames Frühstück

1 Tag

In Präsenz

Immer **dieselbe Übung!** (Kata)

Immer 2-4 zusam-men!

Kurze Einleitung

Session 1

Session 2

Session 3

Gemeinsames Mittagessen

Session 4

Session 5

Session 6

Abschluss

Insgesamt 4-6 Sessions

à 45 Minuten

10 Min. Retro

5 Min. Pause

Immer neue Gruppe!

Immer andere „Constraints“

Evtl. gemeinsamer Ausklang

ℹ️ Jedes Jahr Anfang November: „Global Day of Coderetreat“

https://www.coderetreat.org/

Und sonst noch?

… **Communities of Practice** (CoP) fördern

z.B. Testen, Code-Regeln, Architektur

… **Impulsvorträge** & Diskussionen

z.B. zu CoMo, Architektur, Code-Design, Tools/Libs/…

… ab und zu techn. **Mitarbeit**,

z.B. beim Bereitstellen neuer (Test-)Werkzeuge

… **Multiplikatoren** aufbauen

z.B. mit Mentoring und via Meetup- und Konferenz-Speaking

Fazit

Fazit

Technisches Coaching

💪 Ist wirksam, weil es die Entwickler:innen in ihrer Lebenswirklichkeit abholt: **Software. Technik. Tools.**

🏆⭐ Fördert *Technische Exzellenz*

🛠️🚚 Übt die Skills für *Continuous Delivery*

👩‍🎓👨‍🎓 Bringt *technisches Wissen & Können* ins Team

Technisches Coaching

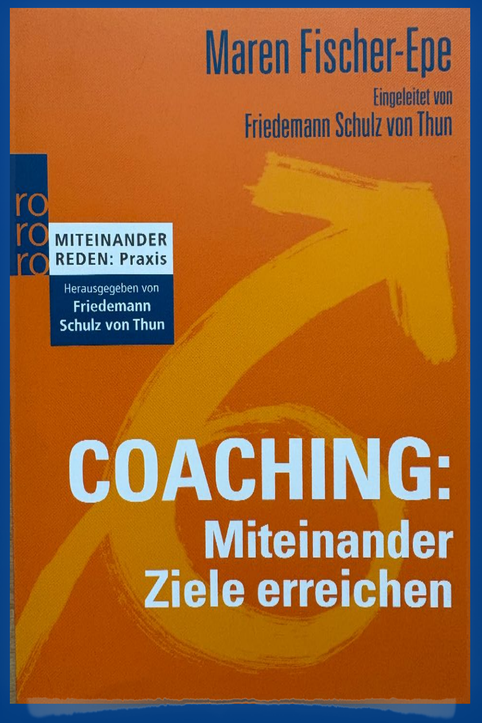
💪 *Nachhaltig* wirksam, weil es Dev-Teams in ihrer Kernaufgabe befähigt: **Software kontinuierlich** & mit **Qualität von Anfang** an entwickeln

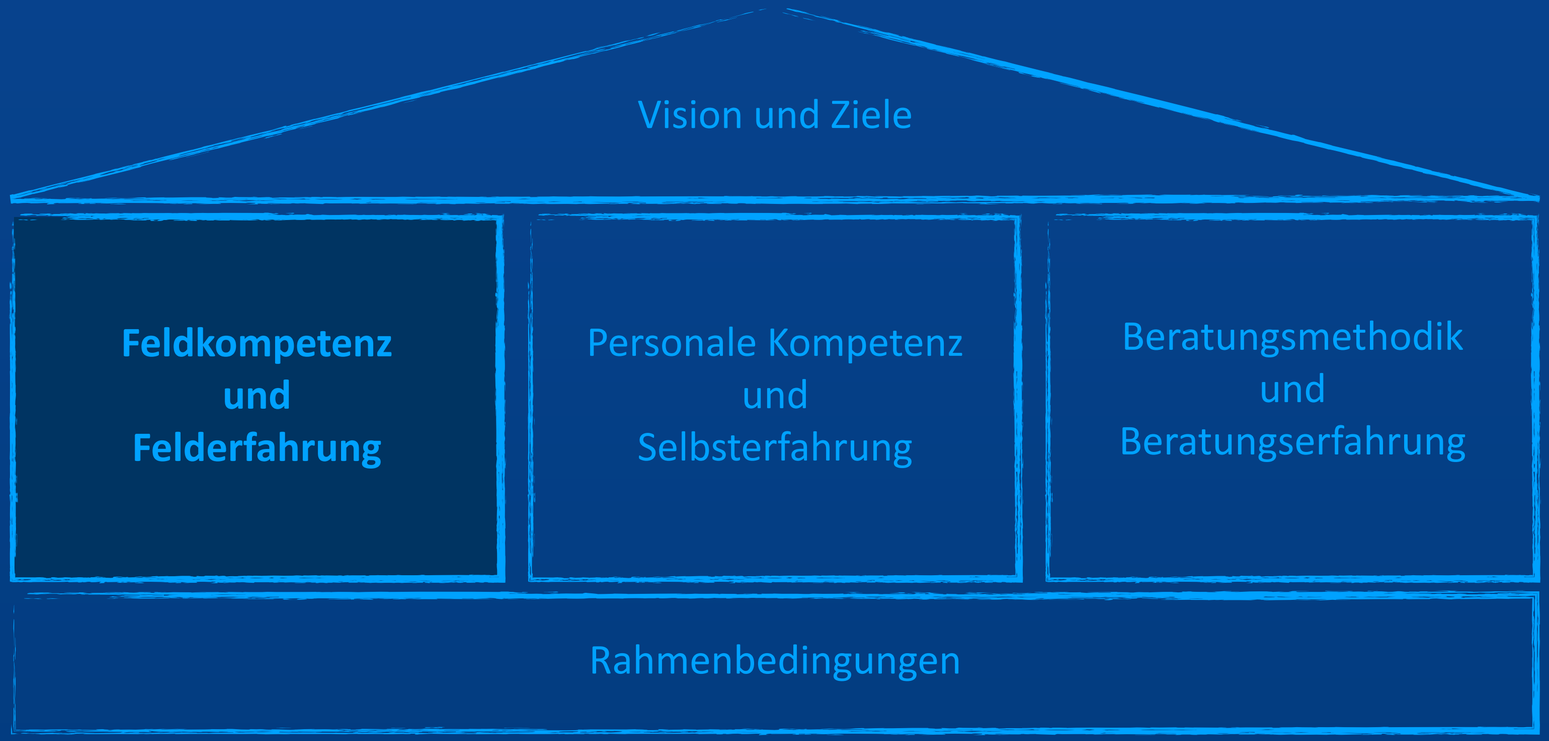
👩‍🎓👨‍🎓 Schafft ein **Umfeld** zum Lernen lernen

👨‍🔬🧪 Ermutigt & befähigt, **Neues** auszuprobieren

🤗 Kein Widerspruch oder Konkurrenz zu Agile Coaching, sondern ergänzt sich und ist **zusammen wirksam.**

Technische Kompetenz nötig?





„Wenn man als Coach aber **keinerlei Feldkompetenz** mitbringt,
sollten Coachee und Coach zumindest im Auge behalten,
wann die **Grenzen der Beratung** erreicht sind.“

Scrum Masters HH XXL
Second Edition

Die Techniker

www.tk.de/IT

Danke!

[ˈtoːmas] 👋

@thmuch