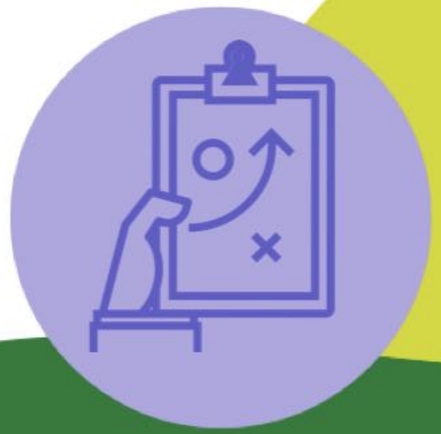
Scrum Tisch Darmstadt

THOMAS MUCH

COACHING-MODELLE FÜR TECHNICAL AGILE COACHING



@thmuch

19. November 2025

Technical Agile Coaching

Für wen ist das relevant?

Wer begleitet (Softwareentwicklungs-)Teams als

- ✅ Senior Developer
- ✅ Tech Lead
- ✅ Architekt
- ✅ Scrum Master
- ✅ Agile Coach

Wer beauftragt Teambegleitungen als

- ✅ Team Lead
- ✅ Projektleiter

TL’DR

Vorhandenen Zustand &
Konsequenzen von Änderungen **besser verstehen.**

Um nichts „kaputt zu coachen“.

Auch beim **Technical** (Agile) Coaching.

Modelle helfen dabei.

➡️ Was ist Coaching? Was sind Coaching-Modelle?

➡️ Agile Coaching vs. Technical Agile Coaching

➡️ **Coaching-Modelle für Technical (Agile) Coaching**

Die Techniker

www.tk.de/IT

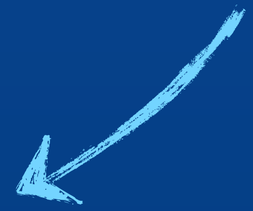
Softwareentwickler
Workshop-Trainer
Coding Architect
Agile Developer Coach
Technical Coach

@thmuch

Generated with GPT-4o

Coaching

Kann komplex werden –
menschlich wie technisch



Professionelle **Begleitung**

zur persönlichen oder beruflichen **Weiterentwicklung**

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

Modelle

Vereinfachte Darstellungen oder Konzepte,

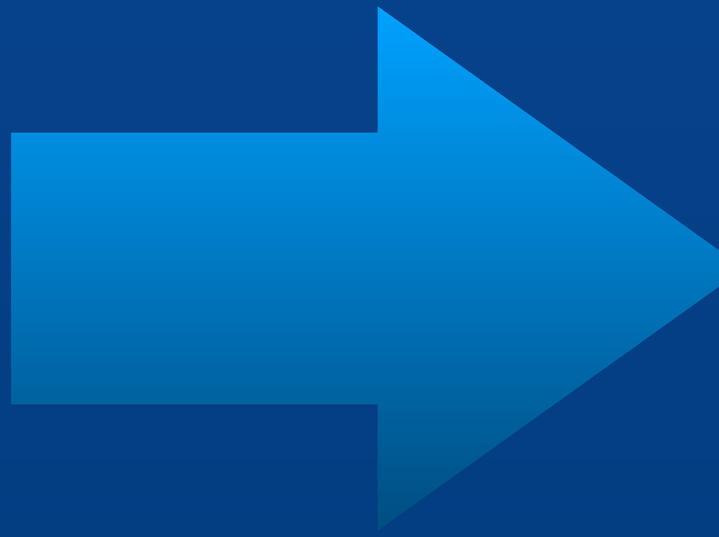
die helfen, **komplexe Zusammenhänge**

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.*“

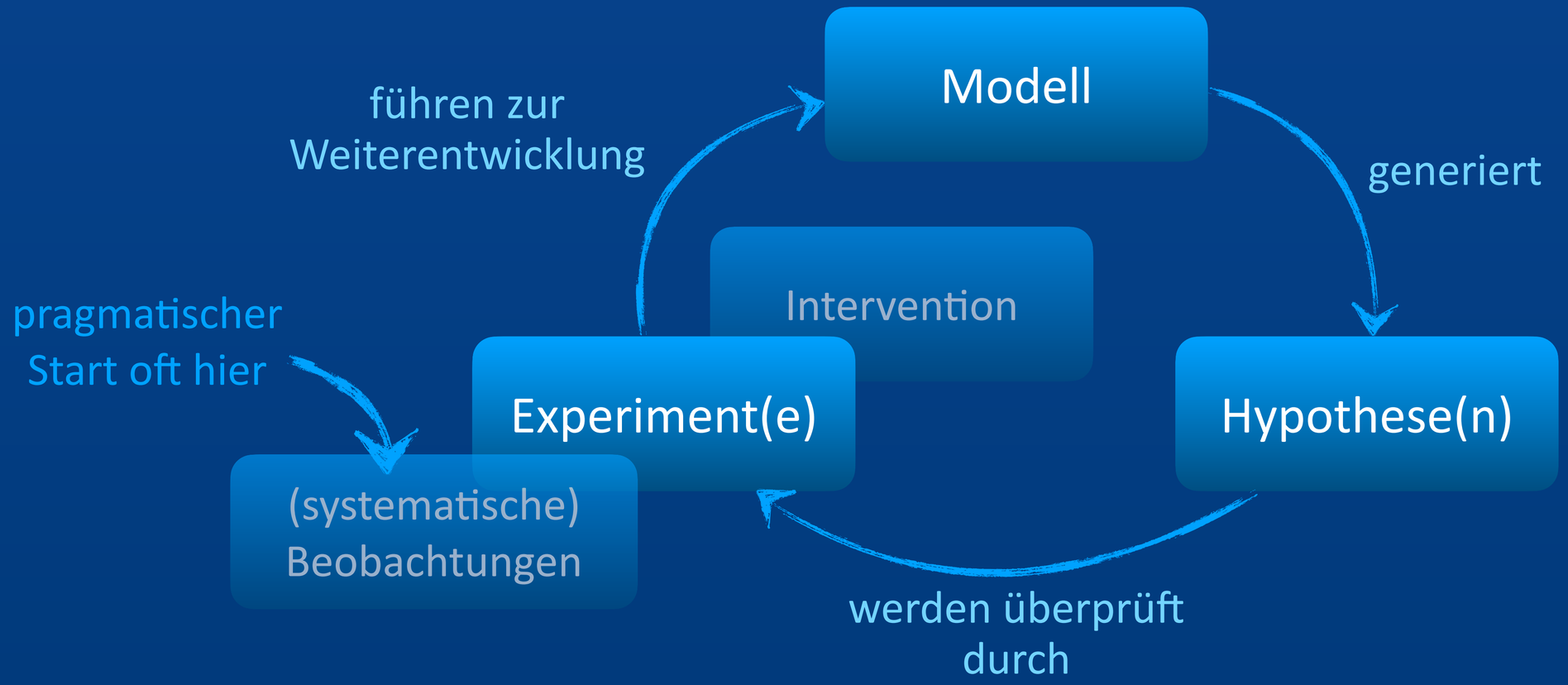
– George E.P. Box

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

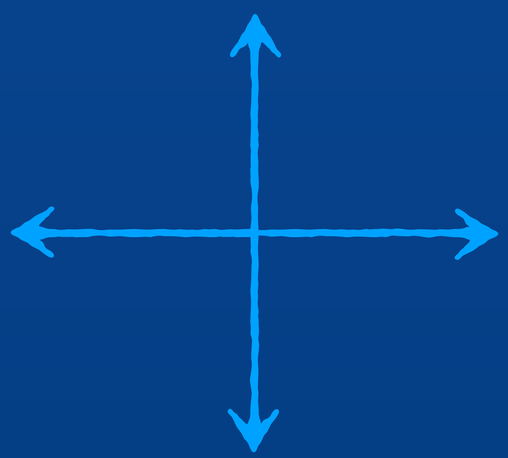
https://medium.com/swlh/all-models-are-wrong-does-not-mean-what-you-think-it-means-610390c40c9c

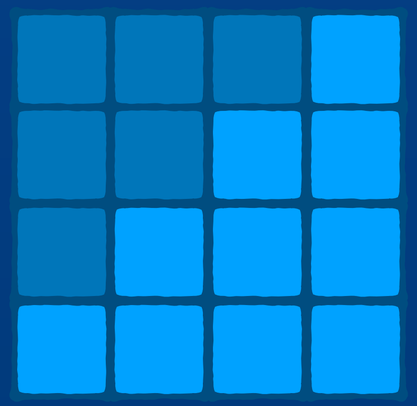
Modelle, Hypothesen, Experimente

ermöglichen **gezieltes**, reflektiertes **Handeln** – statt reinem Bauchgefühl



Bekannte Coaching-Modelle

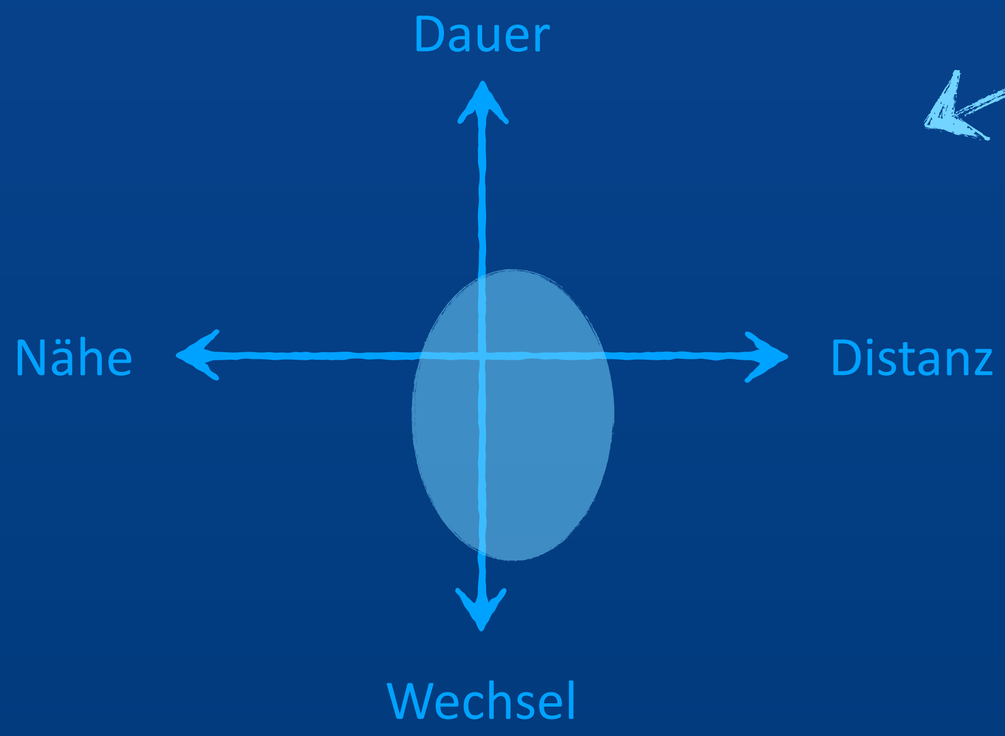




unvollständige
Auswahl

Wie „funktionieren“
unterschiedliche
Modelle?

Riemann-Thomann



Welche **Wünsche** und **Ängste** hat ein Individuum?

Lassen sich damit Konflikte im Team besser **verstehen**?

Wie balanciert ist ein Team?

Immer dran denken:
Vereinfachung. **Schubladen**. Modell 😉

Aber: **Evtl. nützlich?**

SCARF

Status

Certainty (Gewissheit)

Autonomie

Relatedness (Verbundenheit)

Fairness

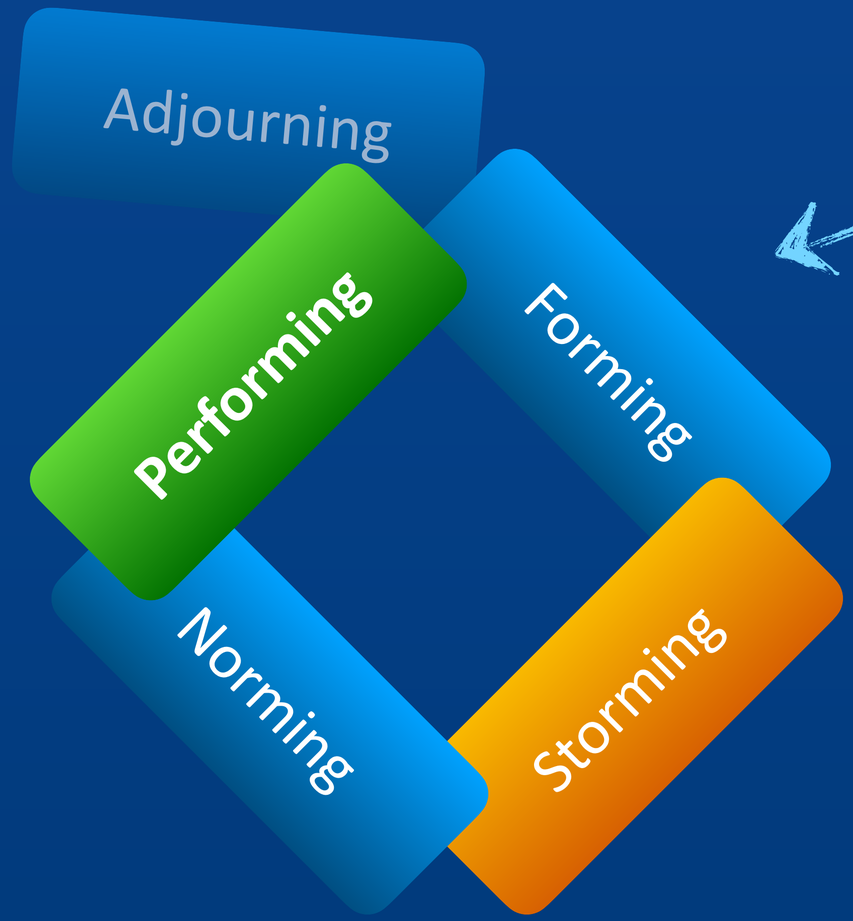
Welche **Belohnungen** und **Bedrohungen** erleben Menschen in diesen Bereichen (Dimensionen)?

Beispiel: Was bedeutet es für einen Java- und JSF-Spezialisten, wenn das Frontend auf TypeScript und React umgestellt wird?

Lässt sich damit menschliches **Verhalten** in **Veränderungsprozessen** besser **verstehen**?

SCARF-Modell von David Rock

Teamphasen nach Tuckman



Wo steht das Team gerade?

Kann das Team
„**Performing**“ erreichen?
Was benötigt es dafür?

Hilft, das Verhalten eines Teams zu verstehen, wenn man als (Technical) Coach dazukommt.

Five Dysfunctions of a Team (Lencioni)

Was hindert ein Team daran, wirklich effektiv zu sein?

Warum funktioniert das Team (noch) nicht „richtig“?

Wo scheitert Zusammenarbeit im Team?

Hackmans Autoritätsmatrix

Wer darf was **entscheiden**?

Was bedeutet das für
Teamautonomie und **Reife**?

Verstehen alle unter einem
„autonomen Team“ das gleiche?

… Modelle für allgemeines (Team-)Coaching.

Nützlich und oft Teil von **Agile Coaching.**

Aber was ist speziell mit **Technical Agile Coaching?**

Agile Coaching

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

Twelve Principles of Agile Software

Für Softwareentwicklung?
Hier gibt’s die
notwendigen „Zutaten“

https://agilemanifesto.org

Agile Coaching

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 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

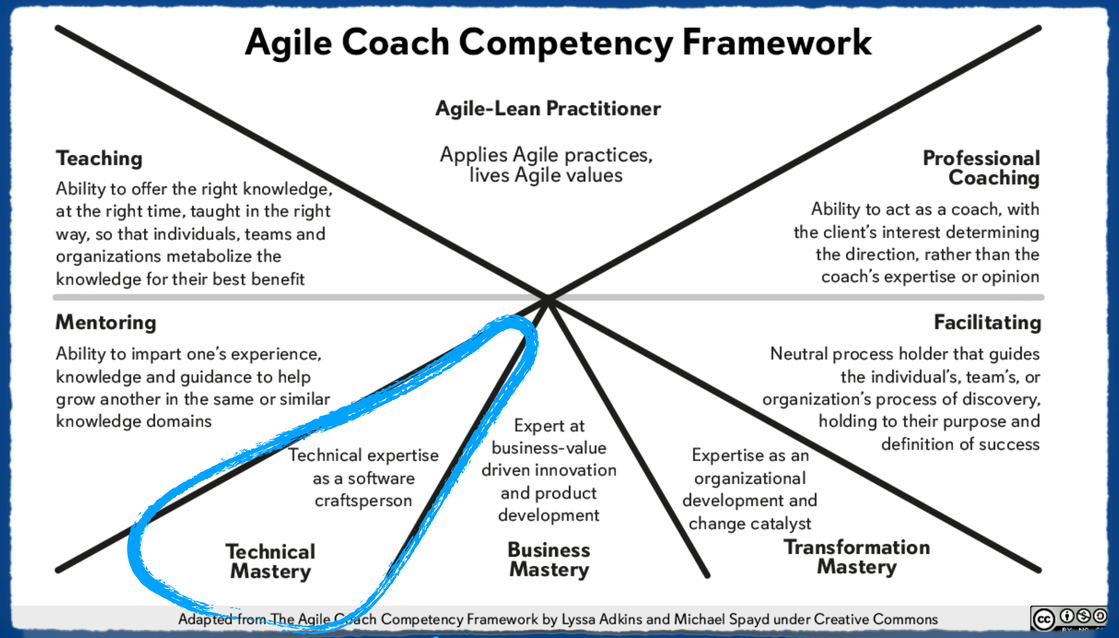
That is, while there is value in the items on
the right, we value the items on the left more.

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

Twelve Principles of Agile Software

Welche **Haltungen (Stances)** sollte ein Agile Coach einnehmen können?

Welche **Kompetenzen** sind wichtig?



„**Technical expertise** as a software craftsperson“

https://agilecoachcompetencyframework.com/

Agile Coaching

https://agilecoachinggrowthwheel.org/
https://resources.scrumalliance.org/Article/agile-coaching-growth-wheel

Welche **Haltungen (Stances)** sollte ein Agile Coach einnehmen können?

Welche **Kompetenzen** sind wichtig?

manchmal sogar nur hierauf

leider oft hierauf beschränkt

Manifesto for Agile Software Development

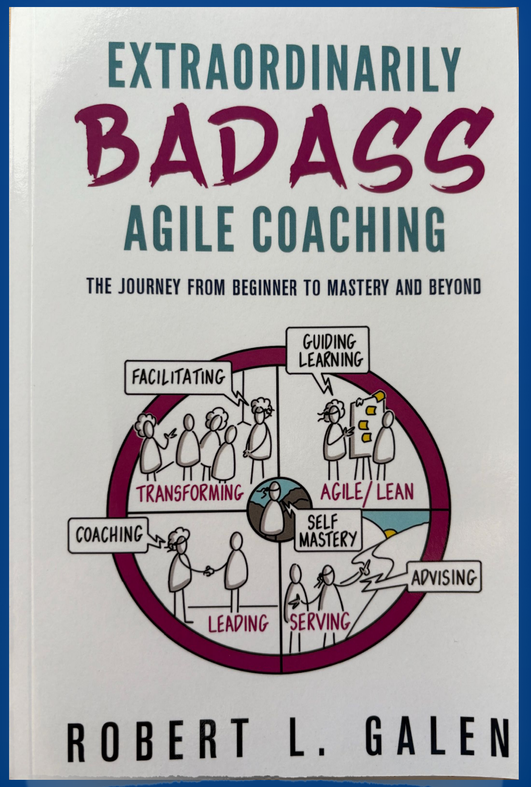
We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

Twelve Principles of Agile Software



Technical Agile Coaching

https://agilecoachinggrowthwheel.org/
https://resources.scrumalliance.org/Article/agile-coaching-growth-wheel

versucht, diese Kompetenzen zu **stärken**

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

Twelve Principles of Agile Software

Welche **Kompetenzen (Skills)** sind wichtig?

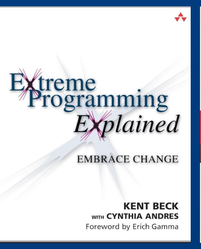
problematisch, falls hierauf beschränkt

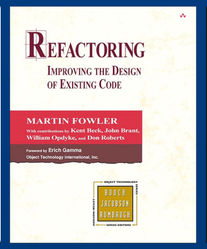
Technical Agile Coaching

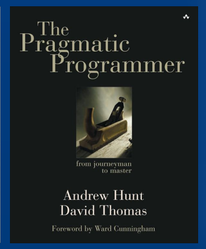
Technical Excellence

We are uncovering better ways of developing software by doing it and helping others do it.

1999

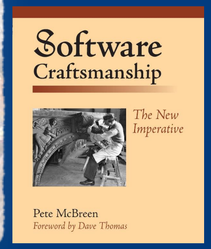


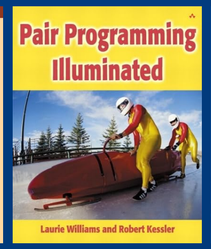


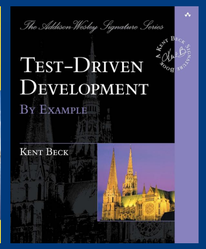


2001

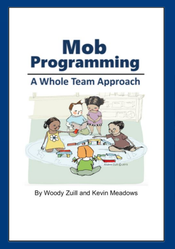
2002



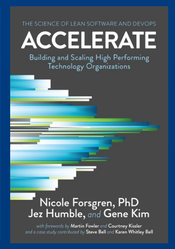




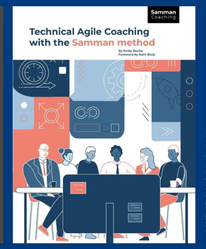
2013



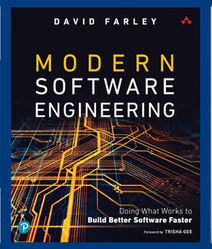
2018



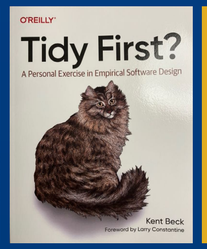
2019



2022

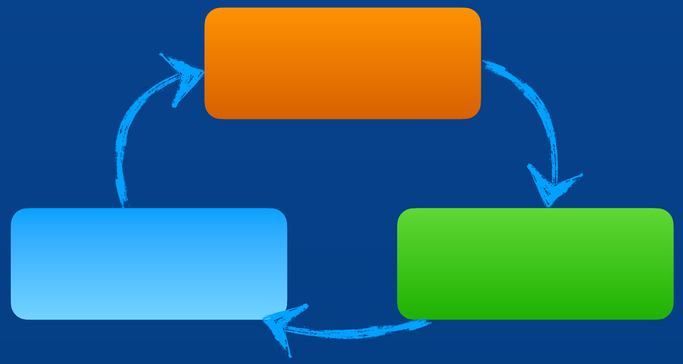


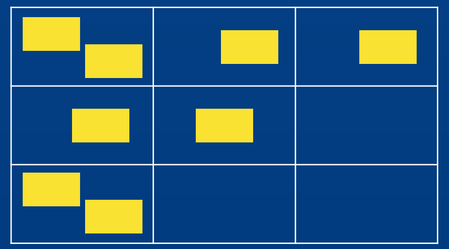
2023

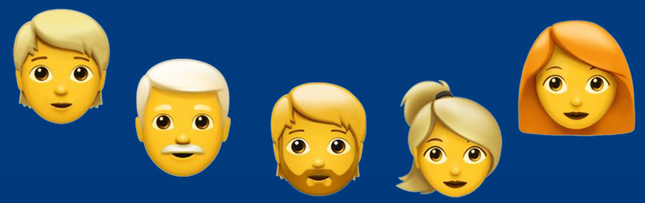


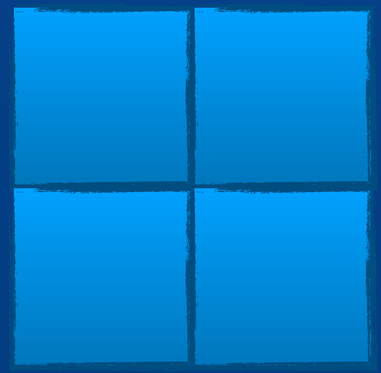
2025

Modelle für Technical Agile Coaching



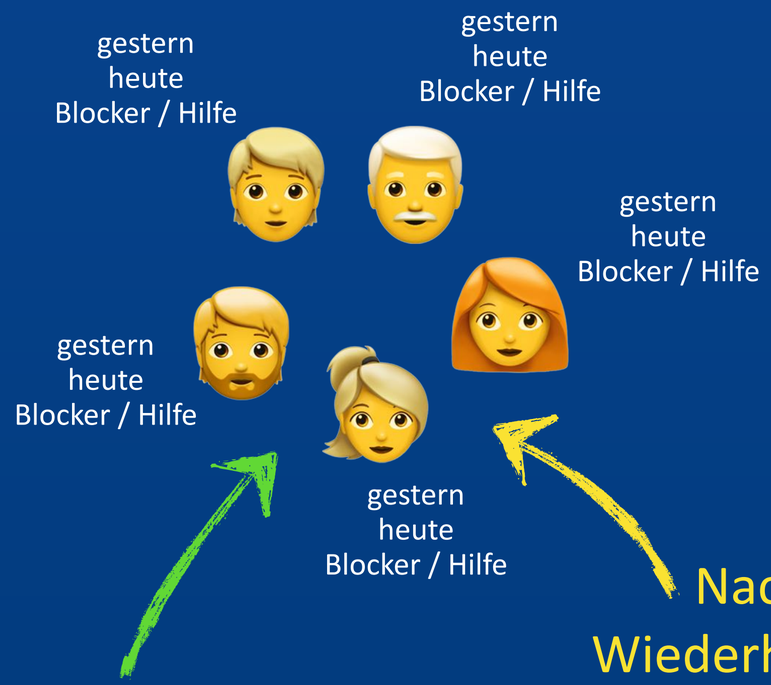






Kommunikation: Daily Standup

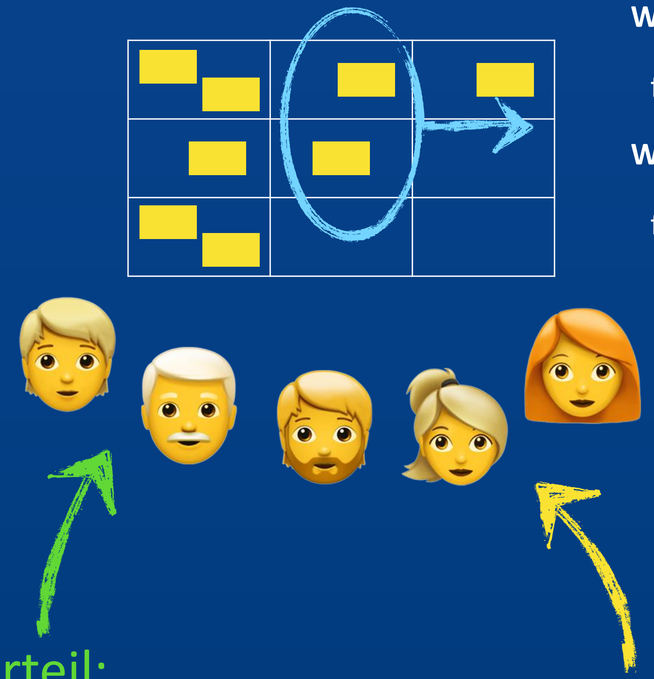
„Walk the People“



Vorteil:
Jeder kommt dran!

Nachteil:
Wiederholungen?
Zusammenhang?
Kontext-Ping-Pong?

„Walk the Work“



Wer & was ist nötig, um diese Aufgabe zügig fertig zu bekommen?

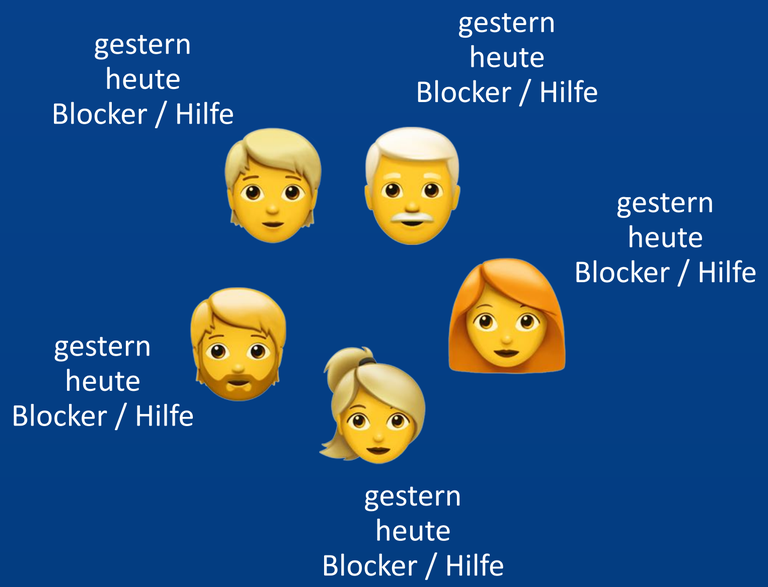
Wer & was ist nötig, um diese Aufgabe zügig fertig zu bekommen?

Vorteil:
Fokus auf
„angefangene Arbeit
fertig bekommen“

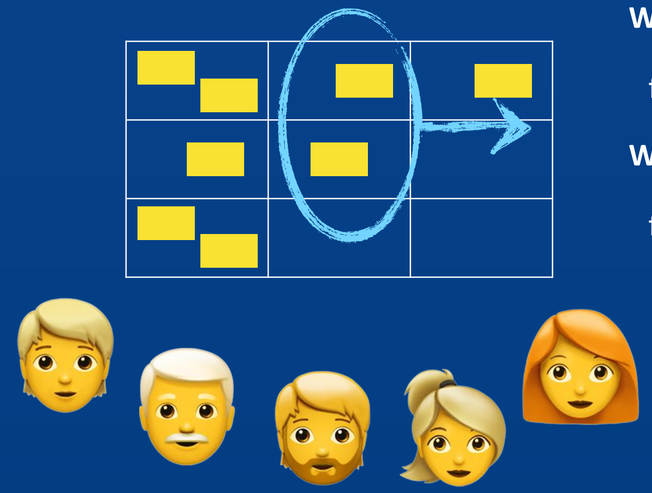
Nachteil:
Was ist mit den
eher Stillen?

Kommunikation: Daily Standup

„Walk the People“



„Walk the Work“



Wer & was ist nötig, um diese Aufgabe zügig fertig zu bekommen?

Wer & was ist nötig, um diese Aufgabe zügig fertig zu bekommen?

Was ist **besser?**

Was braucht das Team gerade?

Welche Zusammenarbeit ist gewünscht?

… bzw. passt zu den Aufgaben?

„Das ist ***technisches*** Coaching???“

Auch. Die Basis.

„Bitte technischer“

*„Macht **TDD**, dann wird euer Code besser“*

Test-Driven Development (TDD)

erst Code-**Nutzung** (ausführbar) beschreiben

passenderer und besser wartbarer (testbarer) Code entsteht

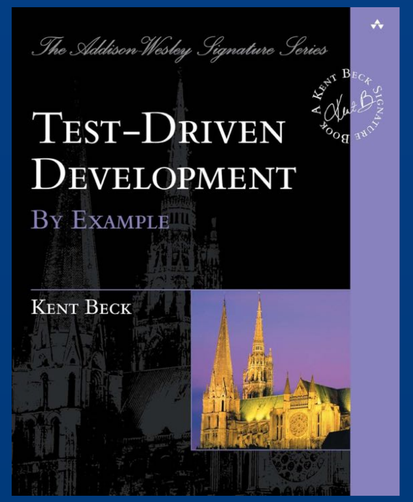
Red
(neuer Test
für neues Verhalten)

Green
(Implementation)

Refactor
(Struktur verbessern)

Ist das auch ein gutes Modell für **Legacy Code?**

2002



Test-Driven Development (TDD)

… in Legacy Code

bestehendes Verhalten als „**korrekt**“ unter Test bringen (Sicherheitsnetz)

sichere Veränderung & Verbesserung möglich

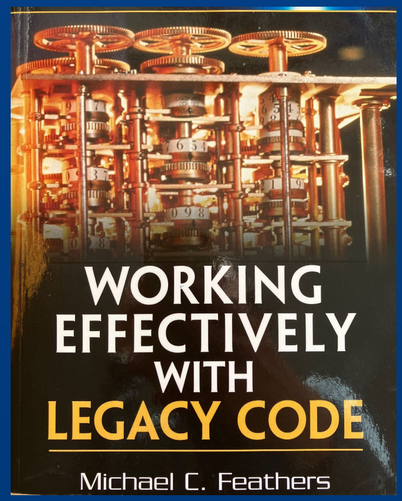
Green
(ungetestet = „korrekt“)

Red
(neuer Approval Test)

Change & Refactor
(Implementation)

Green
(Approval Test)

2005

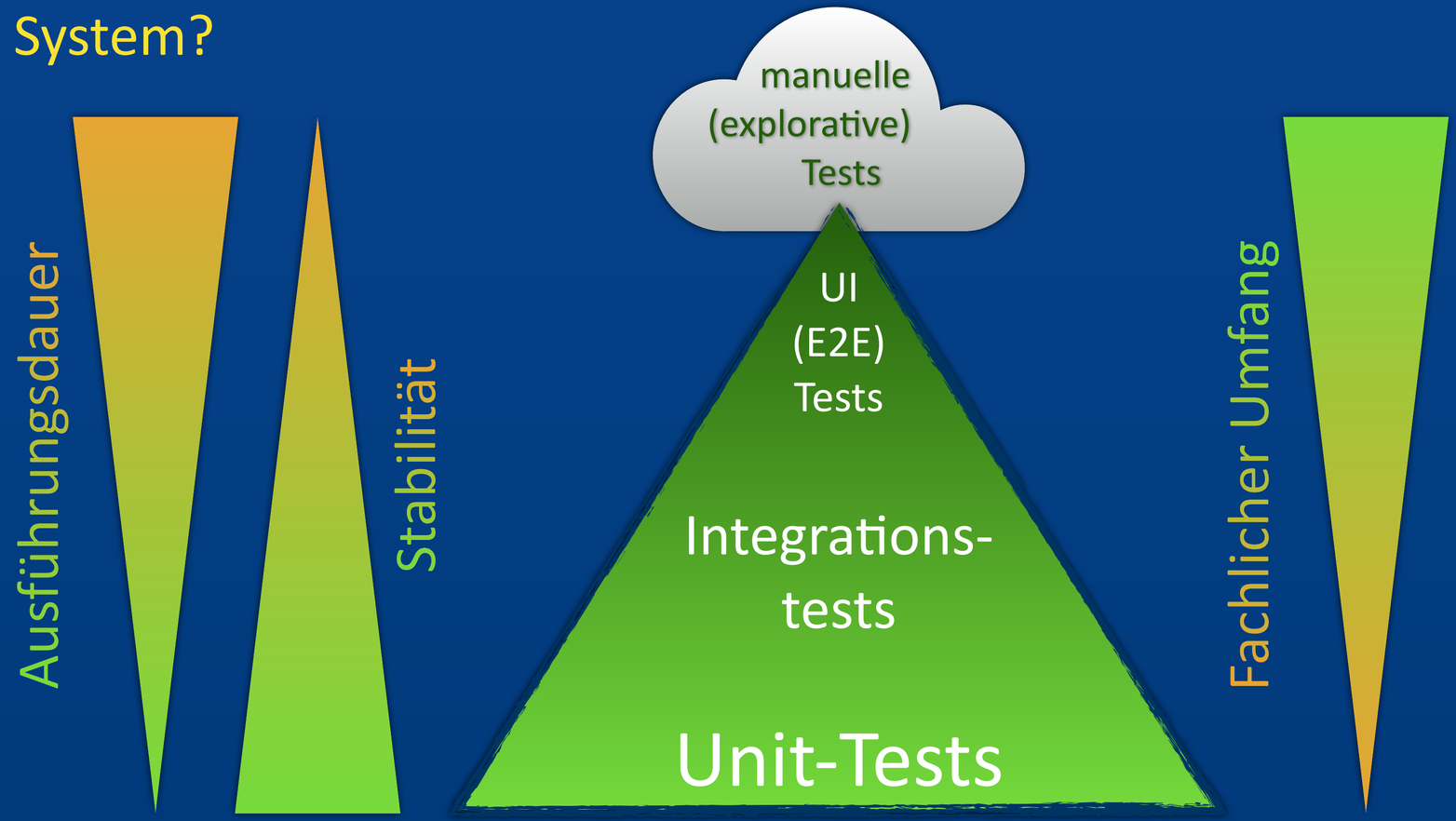


*„Schreibt ganz viele **Unit-Tests**,*

haben wir im letzten Projekt auch so gemacht“

Klassische Testpyramide

Stimmen diese Annahmen
für unser System?

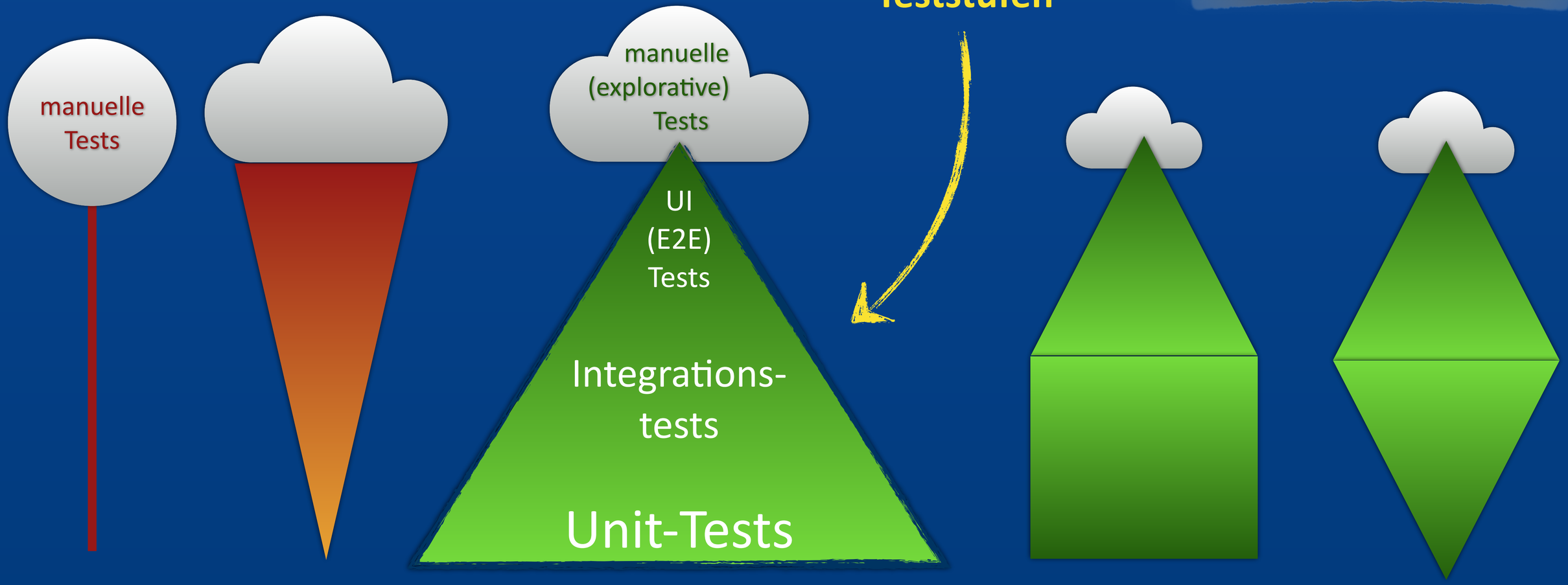


Wie gehen wir mit
dem Spannungsfeld
„Anzahl Tests“
vs.
„fachlicher Umfang“
um?

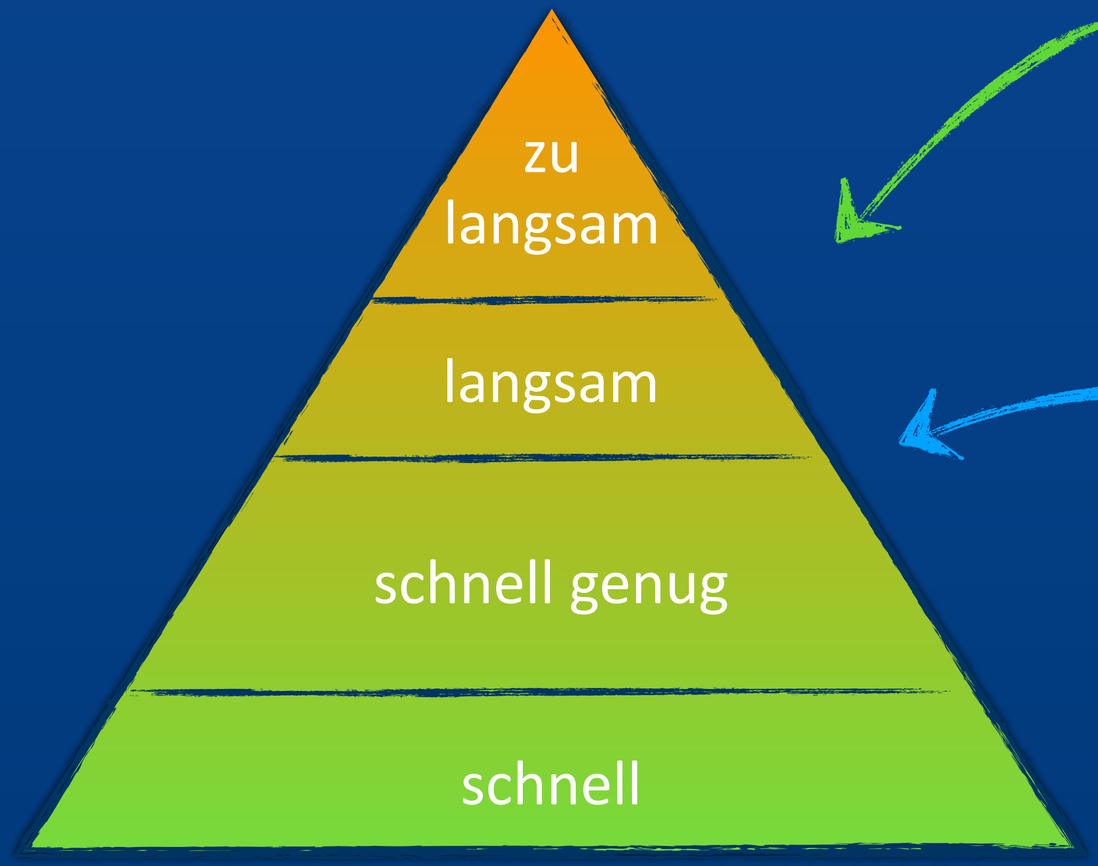
Test„pyramiden“

Immer noch Kopplung an **Teststufen**

Was lässt sich möglichst **schnell & stabil** testen?



Test**geschwindigkeits**pyramide



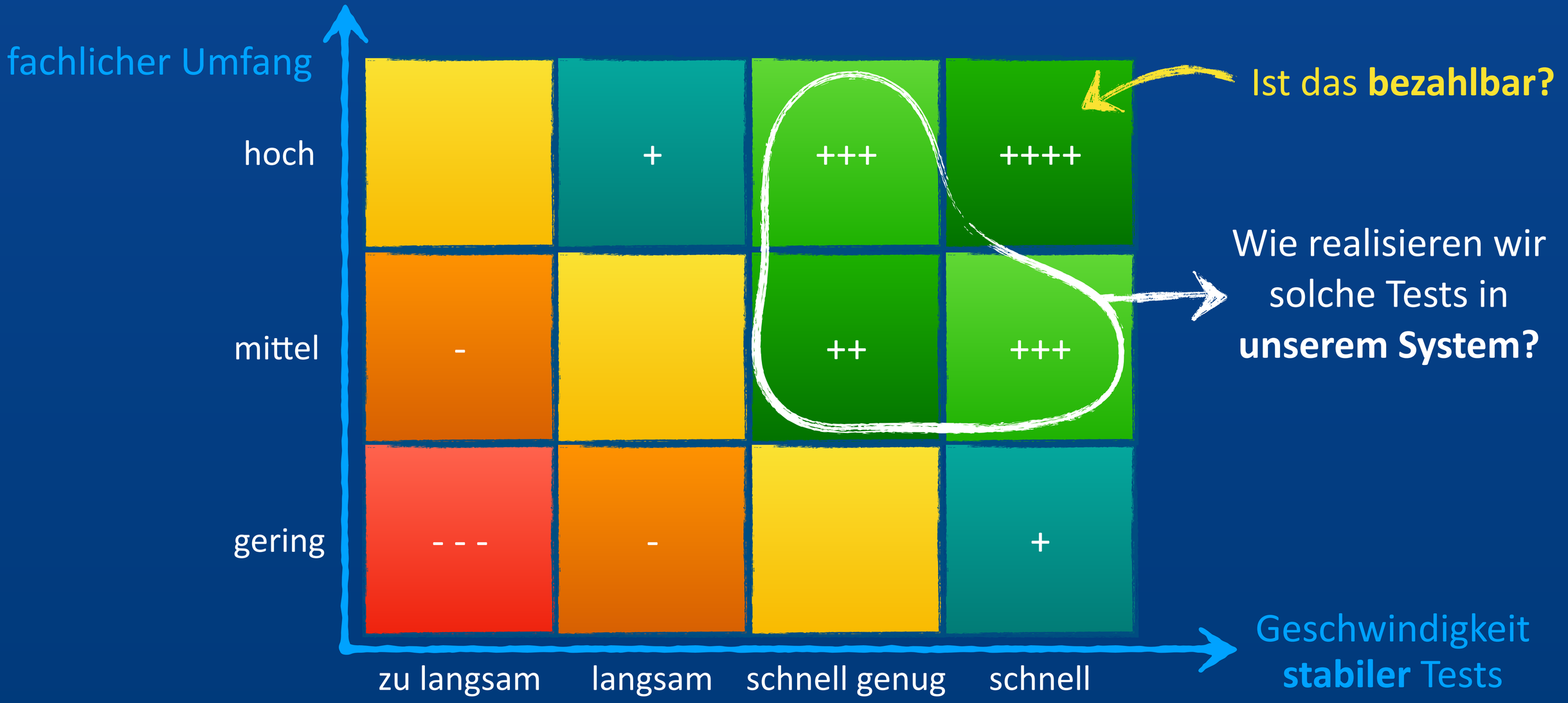
Hilft beim **Beobachten**

Sagt **nicht**,
auf welcher Stufe
getestet werden soll

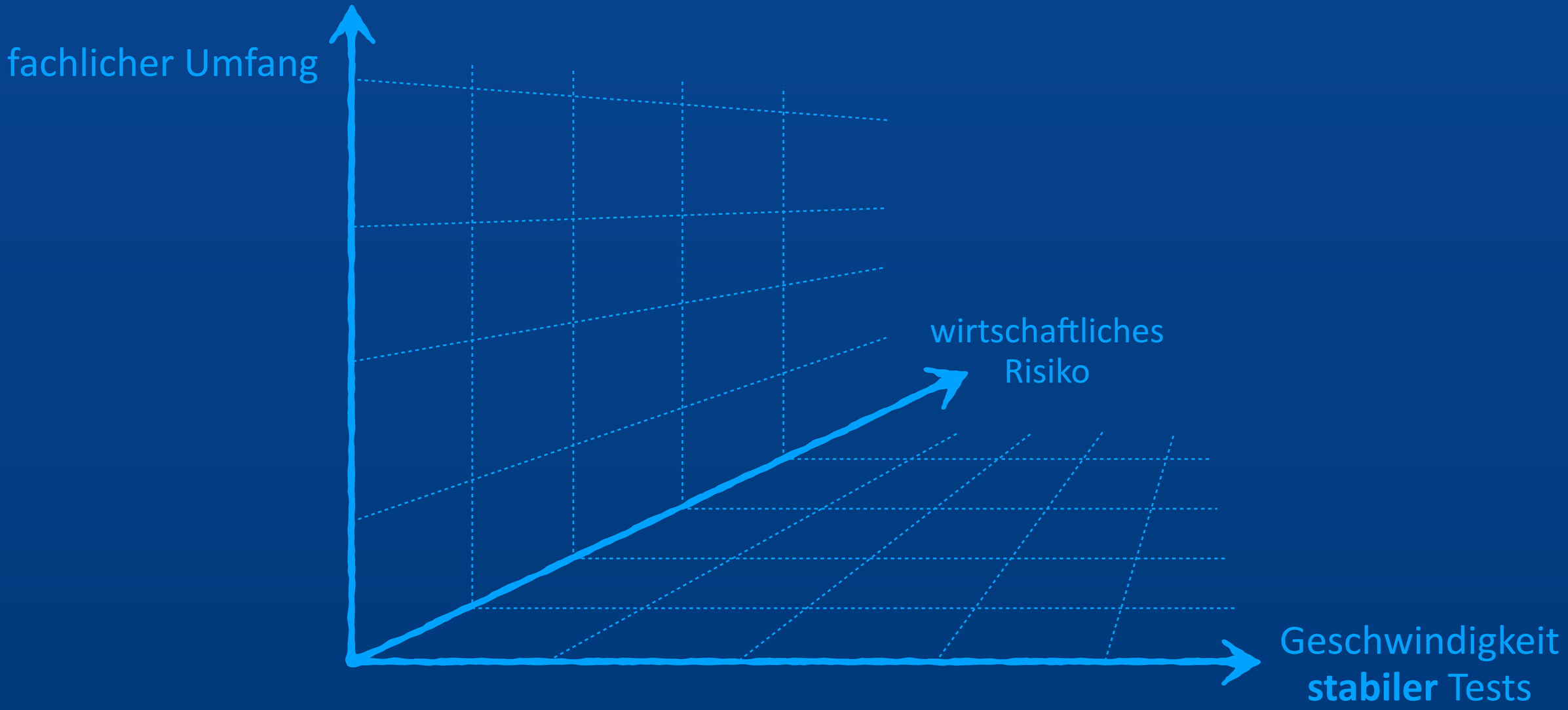
(Systemintegration/E2E,
System, Integration, Unit)

Sind viele schnelle
Tests **ausreichend?**

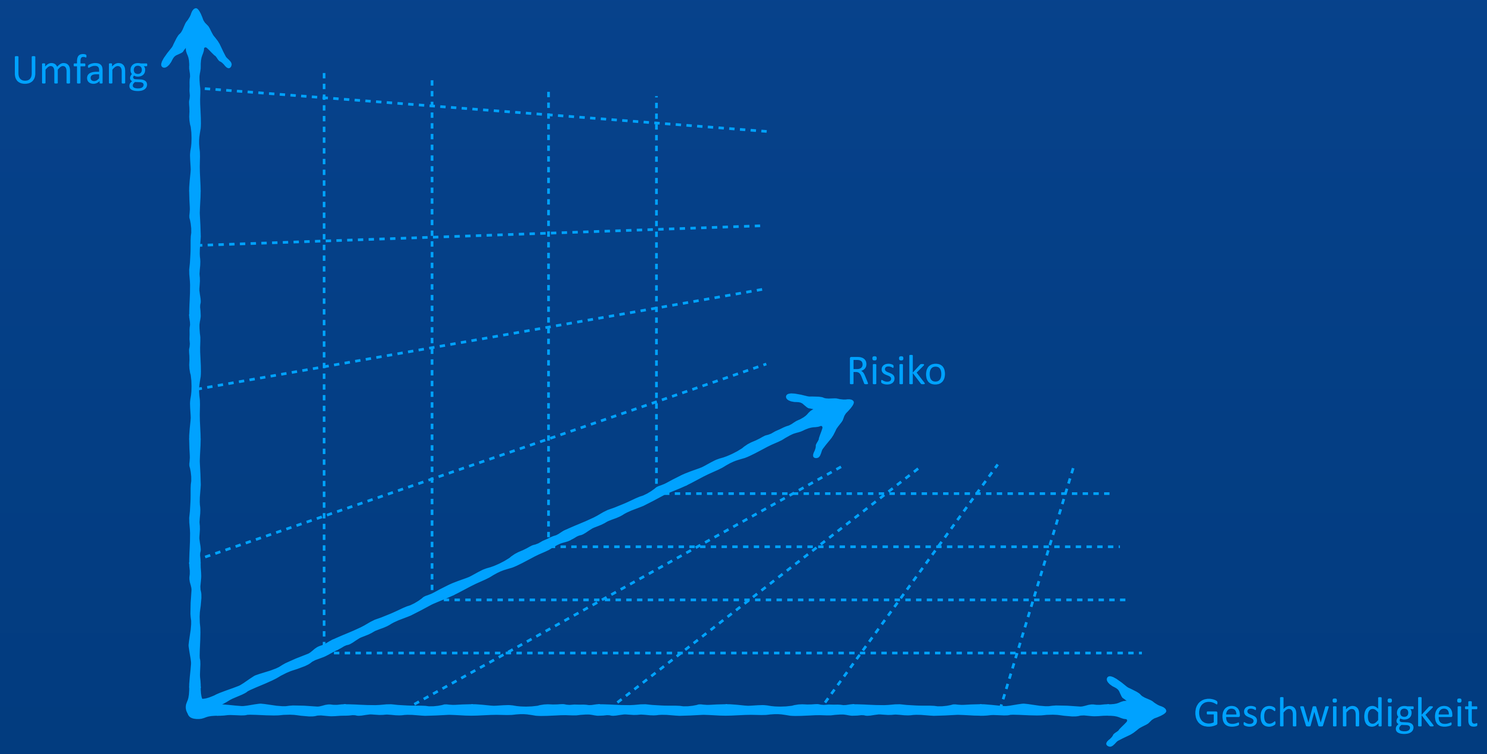
Tests: Sicherheitsnetz-Matrix

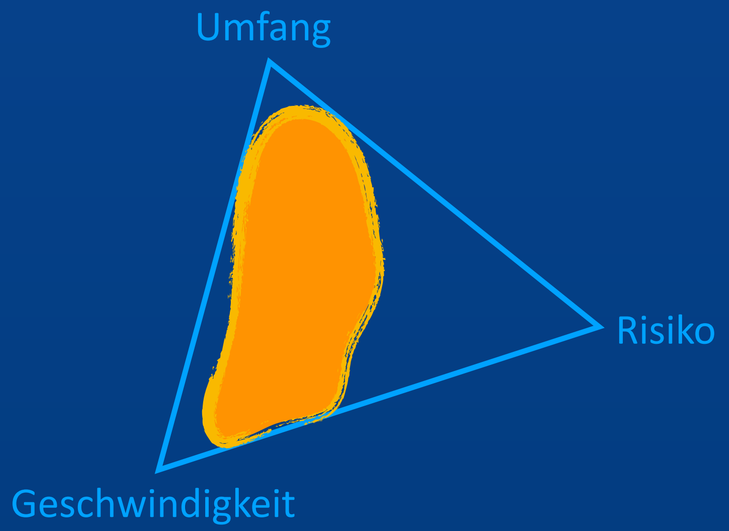


Es fehlen noch Dimensionen…



Wann werden Modelle zu kompliziert?





bzw. wann müssen Modelle so sehr vereinfacht werden, dass ihr Nutzen abnimmt?

*„So viele **Technische Schulden** im Code.*

Macht die weg! Nebenbei.“

Technische Schulden

Metapher Modell

+ Zinsen

Kredit aufnehmen -Zins-Tilgung Restschuld+Zinsen

Ist jeglicher „suboptimaler“ Code so entstanden?

Wenn nicht, sollten wir evtl. unterschiedlich damit umgehen?

Technical Neglect (Vernachlässigung) und **„Cruft“** (Schludrigkeit)

	leichtsinnig	besonnen	
bewusst	„Keine Zeit für Code-Design“	„Jetzt liefern & die Konsequenzen tragen“	„klassische“ technische Schulden
unbeabsichtigt	„Was ist eine Schichten-Architektur“?	„Jetzt wissen wir, wie wir es hätten umsetzen sollen“	Viel **gelernt!**

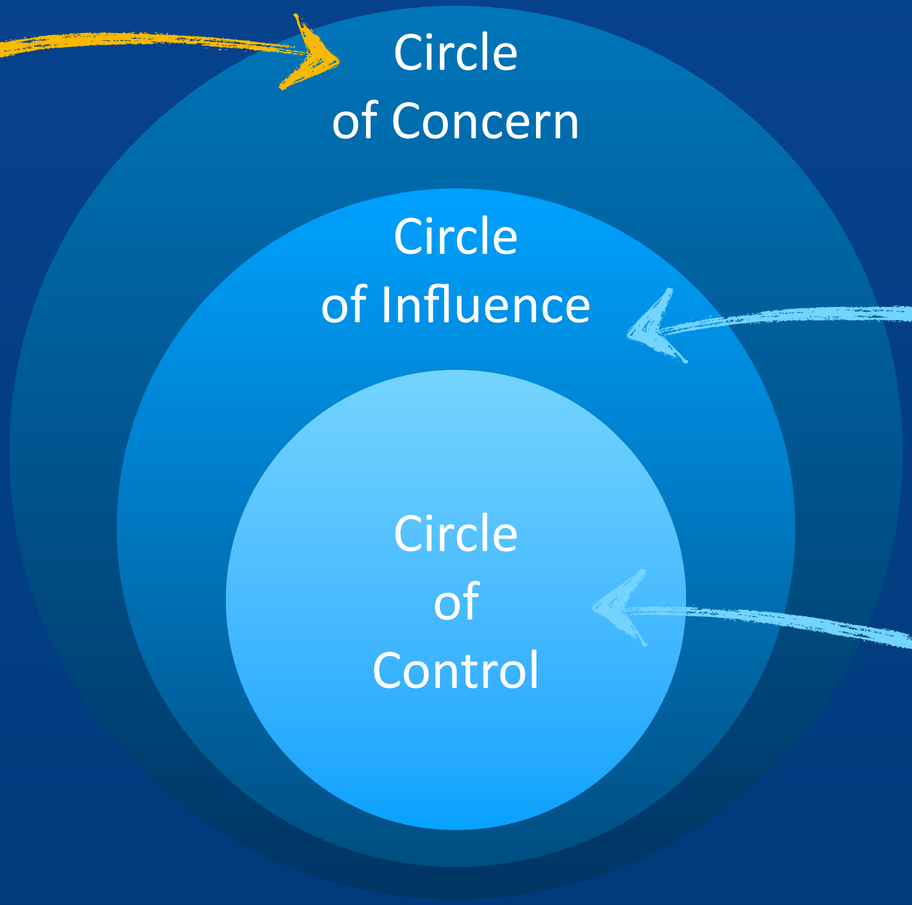
„Technische Schulden“-Metapher von Ward Cunningham
https://martinfowler.com/bliki/TechnicalDebtQuadrant.html

*„Wir müssen das **gesamte System** neu entwerfen,*

um die technischen Schulden in den Griff zu bekommen.“

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“

„Der Kollege kann das alles irgendwie intuitiv.

Wie soll ich das jemals lernen?“

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

Shu Ha Ri

Ab wo ist welche **Haltung** sinnvoll?

Coaching

Mentoring

Training

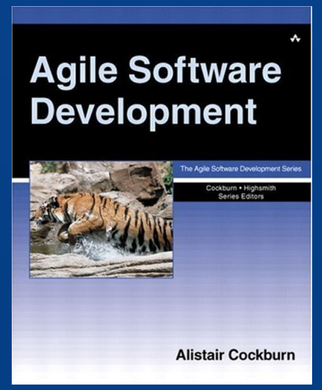
守 破 離

Regeln befolgen

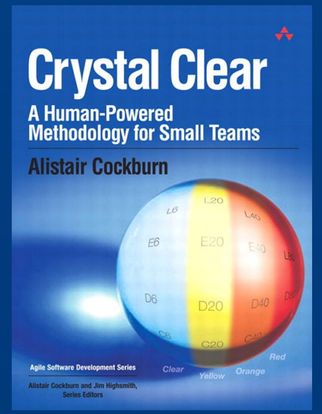
Regeln bewusst brechen

keine Regeln mehr benötigen

2002



2005



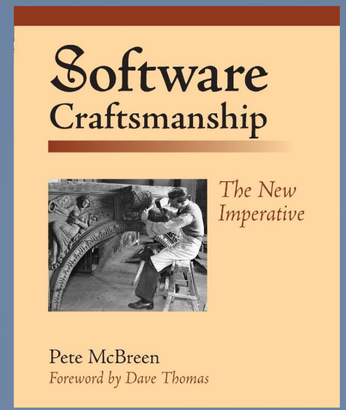
Apprentice – Journeyman – Master

Lehrling – Geselle – Meister

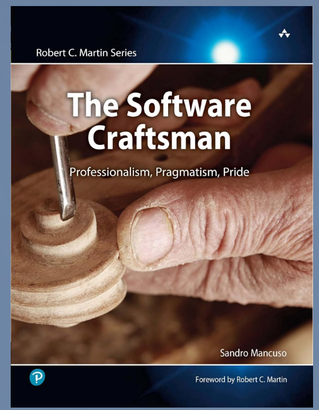
Brauchen Ausbildung.
Dürfen nicht allein gelassen werden!

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

<your model here>

Ist das nicht einfach **gute Architekturarbeit?**

Gutes **Technical Leadership?**

Ja. Genau!

Sind das schon **Modelle** oder nur **Fragen**,
die zu Modellen führen?

Vereinfachte Darstellungen oder Konzepte,

die helfen, komplexe Zusammenhänge

zu verstehen, zu erklären oder zu steuern

Hilft es uns, eine Situation zu **beobachten**, zu **bewerten**
– und angemessene **Schritte** daraus abzuleiten?

„Verstehen wollen“

Modelle

Beobachtung, Erklärung, Impuls & Steuerung

vs.

„Zielzahl/-bereich erreichen“

Messung & Beurteilung

Metriken / KPIs

„Hab was ganz anderes erwartet …

Wie euer Coaching strukturiert ist“

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
- 2h Team-Programming mit **Code-Kata**
- Mini-Retro
- 2,5h Team-Programming mit **Code-Kata**
- Abschluss

„Coding-Skills“

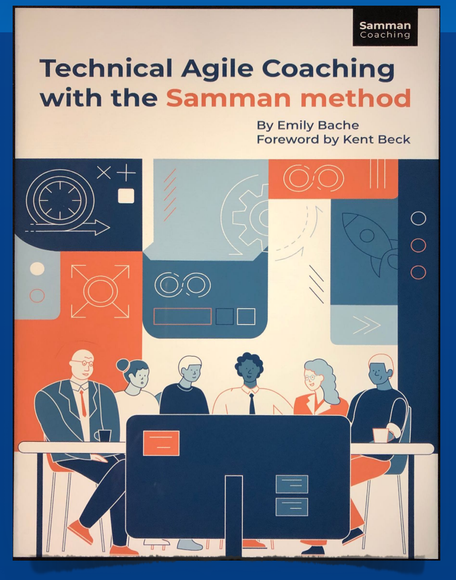
i.d.R. 1 Tag

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:

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

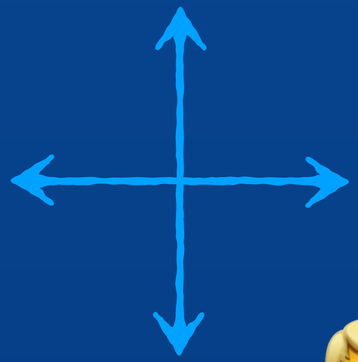
9x

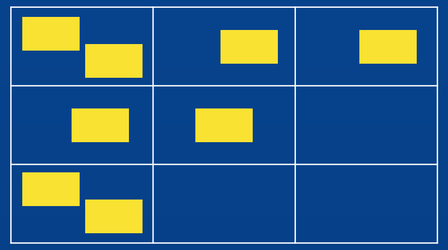
10.

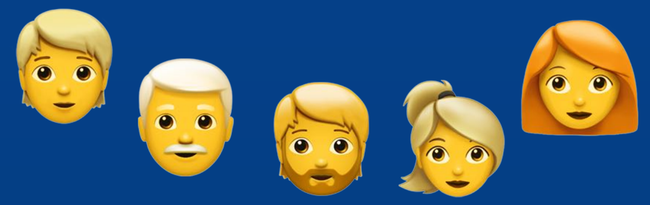
1h **Inspirational Demo**
(evtl. Video)

https://sammancoaching.org/

Modelle für Technical Agile Coaching

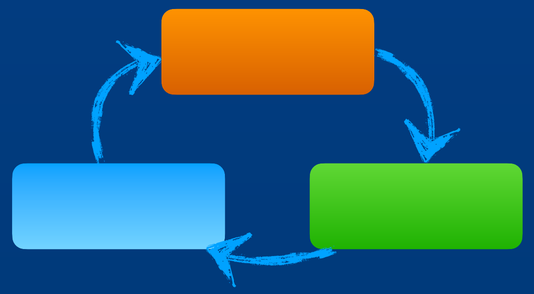




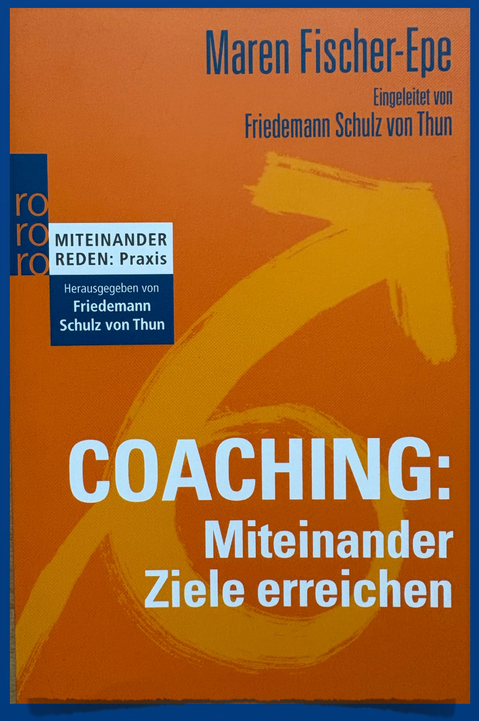


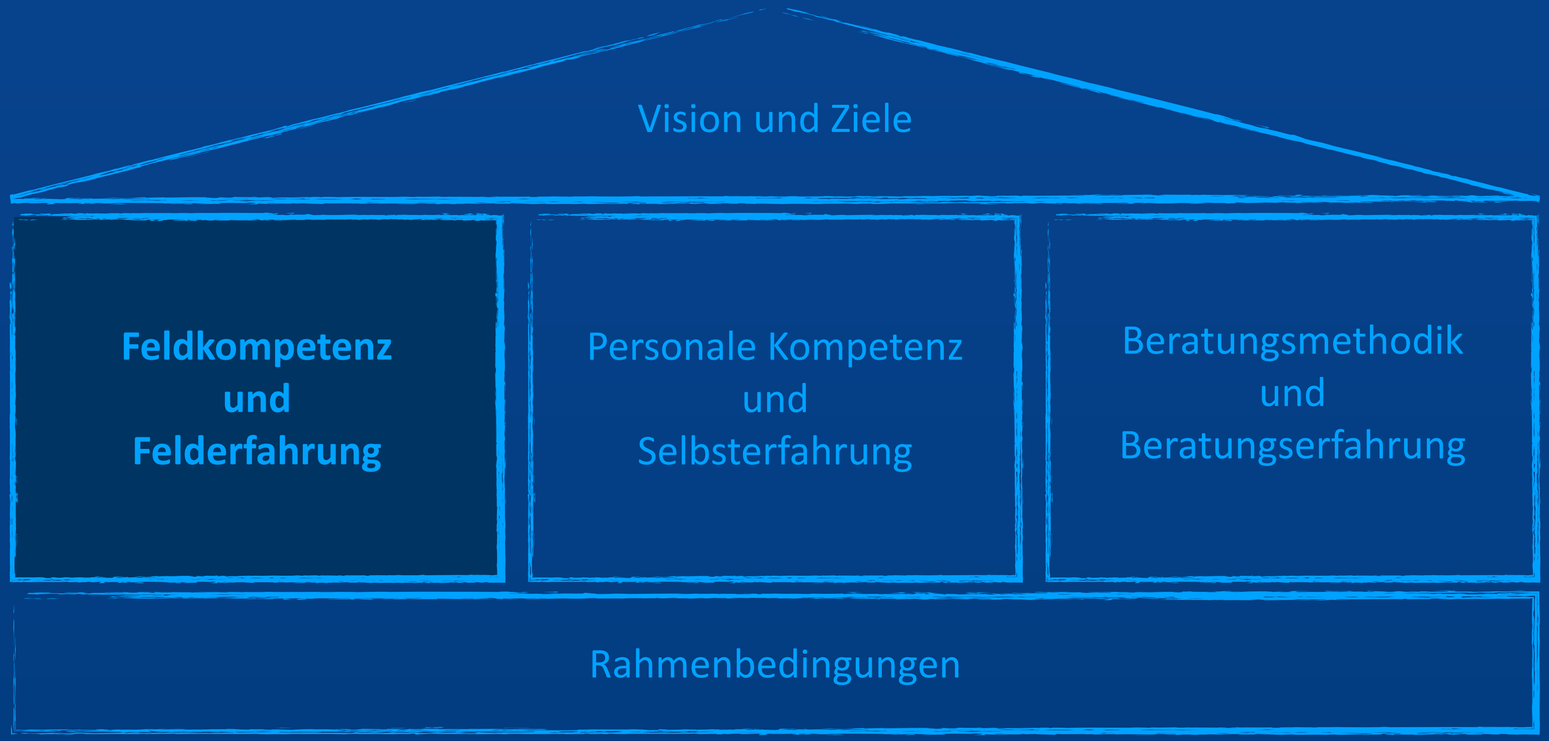
Menschen
Technik
Kontext
verstehen.

Dann erst handeln.



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 Tisch Darmstadt

Vielen Dank!

@thmuch

Die Techniker

www.tk.de/IT