Coaching-Modelle
für

Technical Agile Coaching

Thomas Much
@thmuch
08. Mai 2025

Technical Agile Coaching

Für wen ist das relevant?

Wer begleitet Teams als

- ✅ Senior Developer
- ✅ Tech Lead
- ✅ Architekt

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

Java 1.0
Java 8
Java 11
Java 17
Java 21
Java 24

@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.“

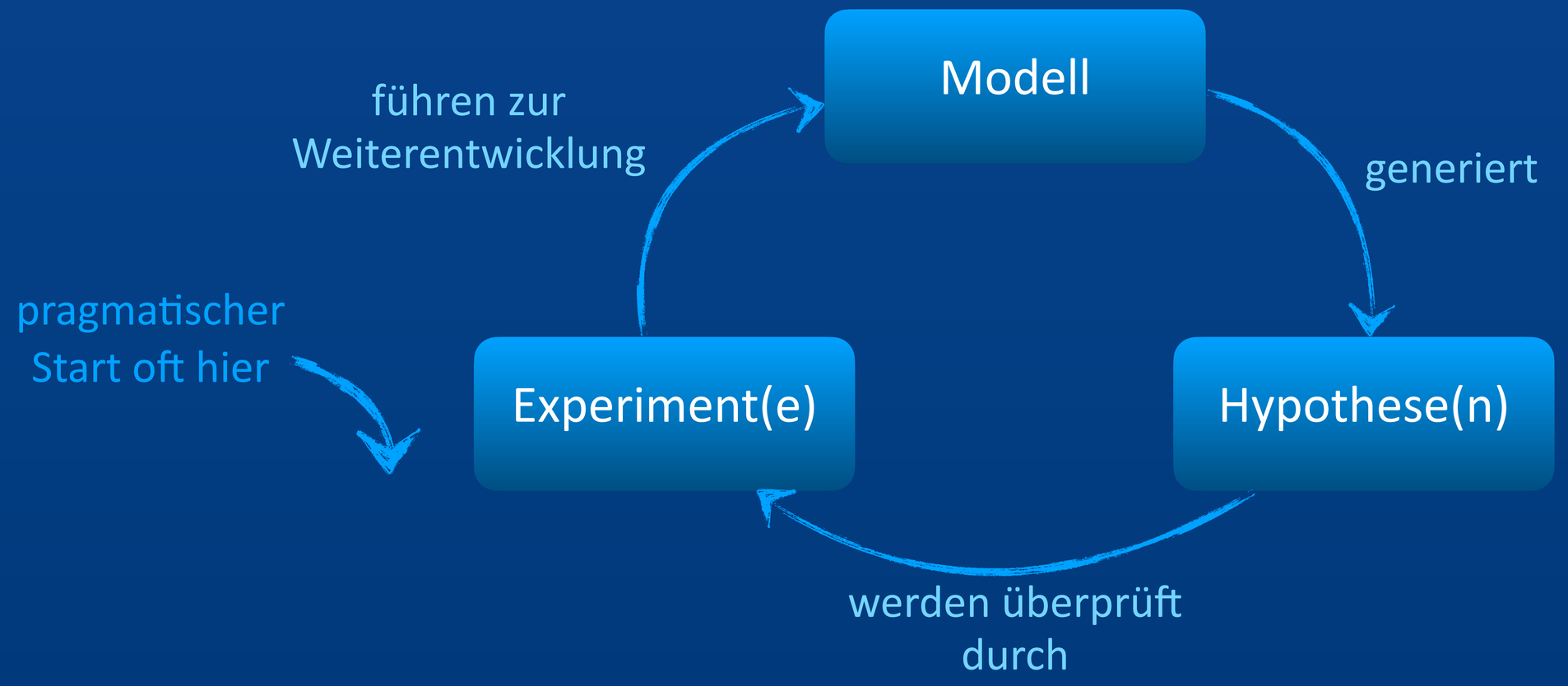
– George E.P. Box

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

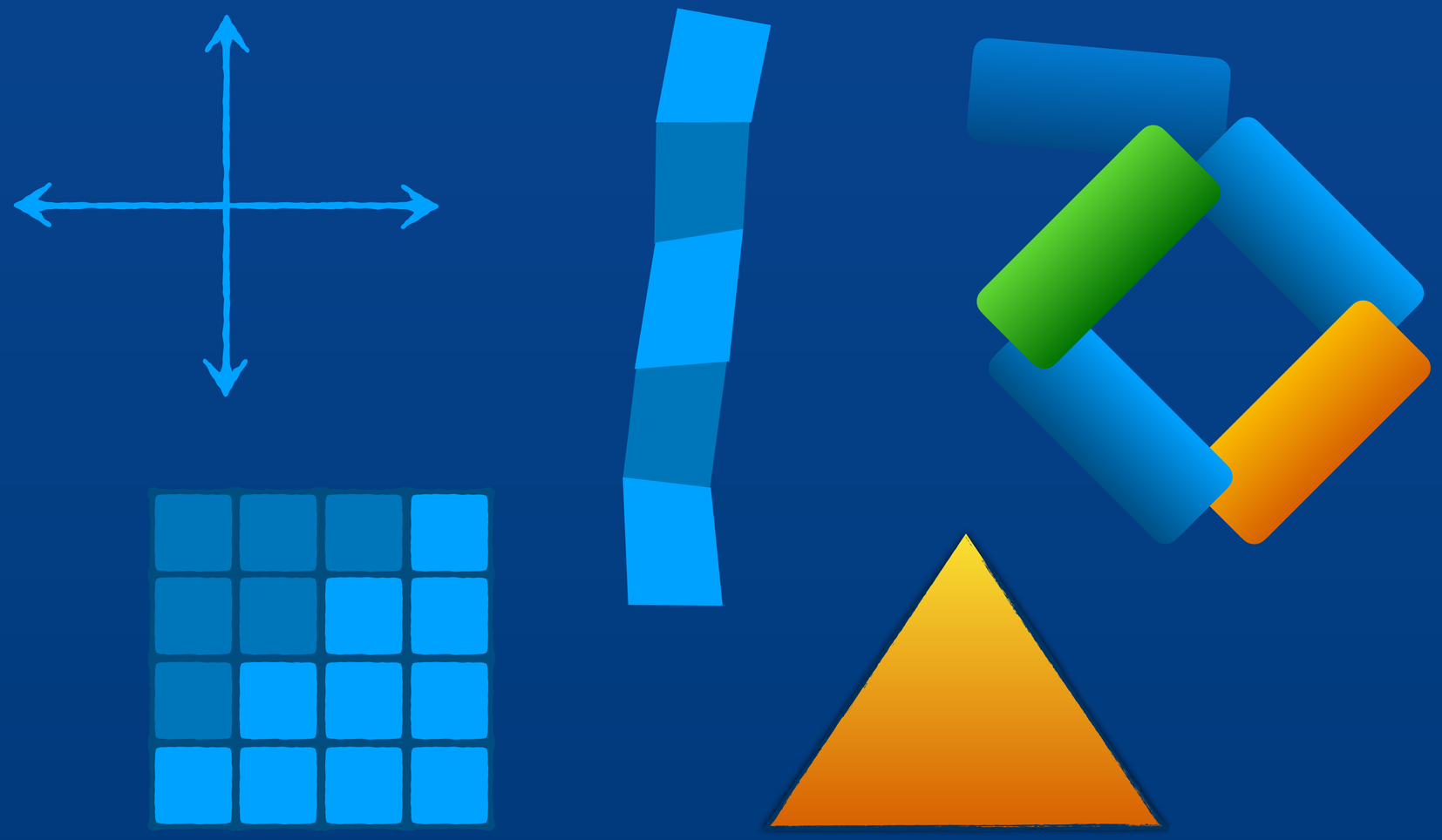
https://medium.com/swlh/all-models-are-wrong-does-not-mean-what-you-think-it-means-610390c40c9c

Modelle, Hypothesen, Experimente

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



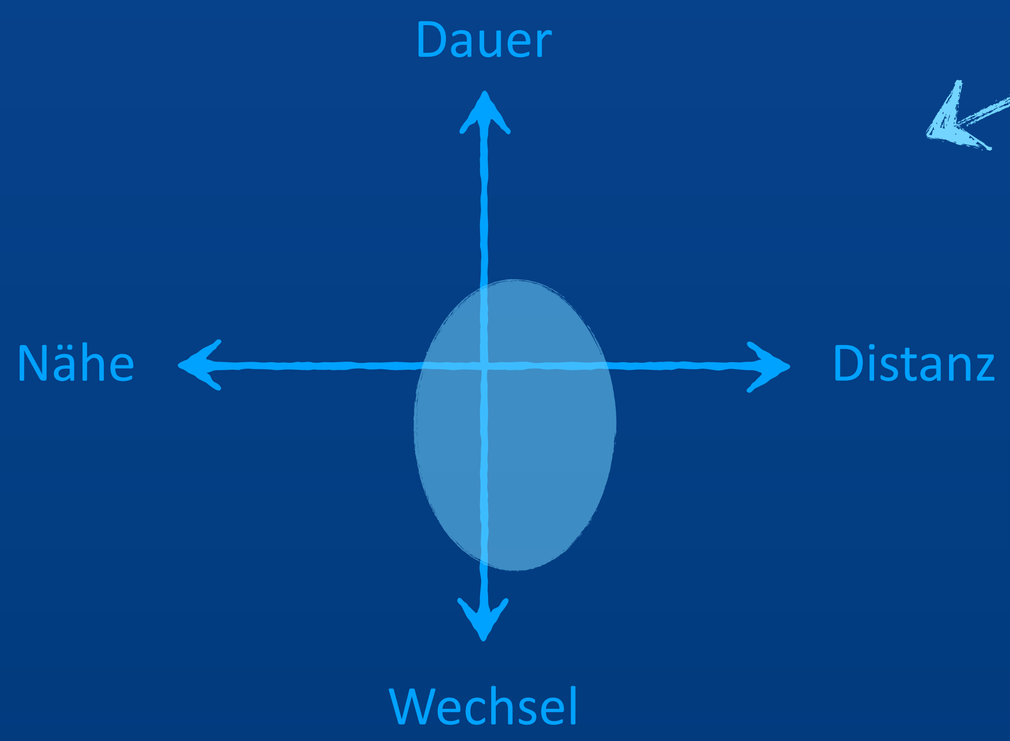
Bekannte Coaching-Modelle



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

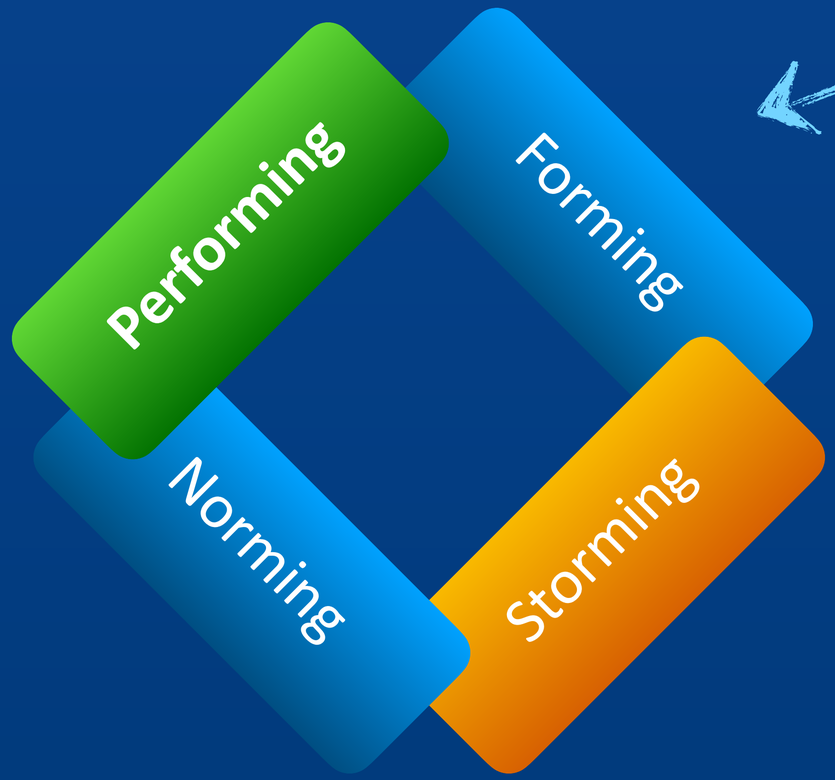
Welche **Belohnungen** und **Bedrohungen** erleben Menschen in diesen Bereichen (Dimensionen)?

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.

… 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	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	

© 2001, the above authors
this declaration may be freely copied in any form,
but only in its entirety through this notice.

Twelve Principles of Agile Software

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

https://agilemanifesto.org

Agile Coaching

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:

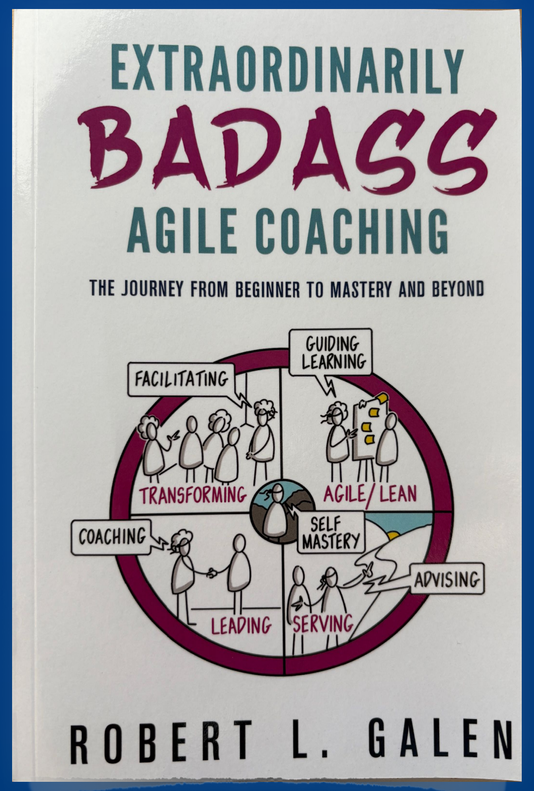
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

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

© 2001, the above authors
this declaration may be freely copied in any form,
but only in its entirety through this notice.

Twelve Principles of Agile Software



Agile Coaching Growth Wheel concept by Shannon Carter; Rickard Jones; Martin Lambert; Stacey Louie; Tom Reynolds; Rohit Ratan; Andre Rubin; Kubair Shirazee; and Mark Summers is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

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

Technical Agile Coaching

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

© 2001, the above authors
this declaration may be freely copied in any form,
but only in its entirety through this notice.

Twelve Principles of Agile Software

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

problematisch, falls hierauf beschränkt

Agile Coaching Growth Wheel concept by Shannon Carter; Rickard Jones; Martin Lambert; Stacey Louie; Tom Reynolds; Rohit Ratan; Andre Rubin; Kubair Shirazee; and Mark Summers is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

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

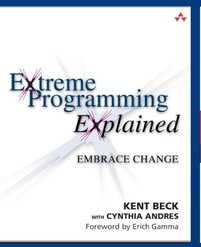
Technical Agile Coaching

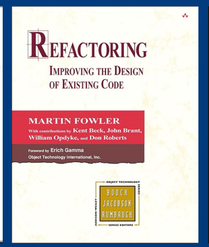
Technical Excellence

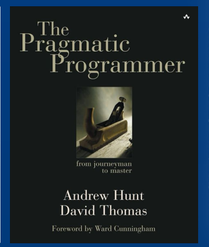
We are uncovering better ways of developing software by doing it and helping others do it.

2001

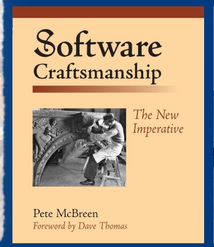
1999

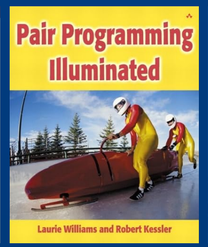


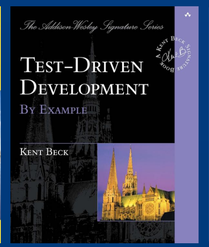




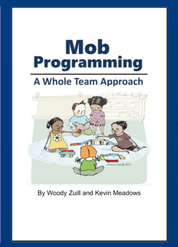
2002



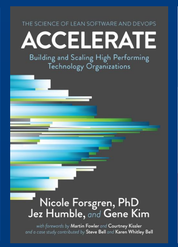




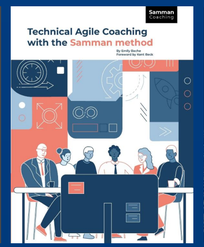
2013



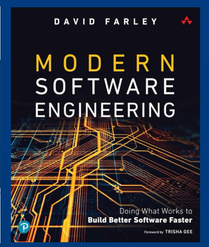
2018



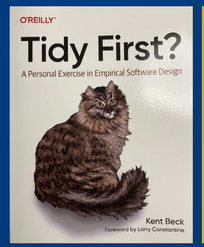
2019



2022

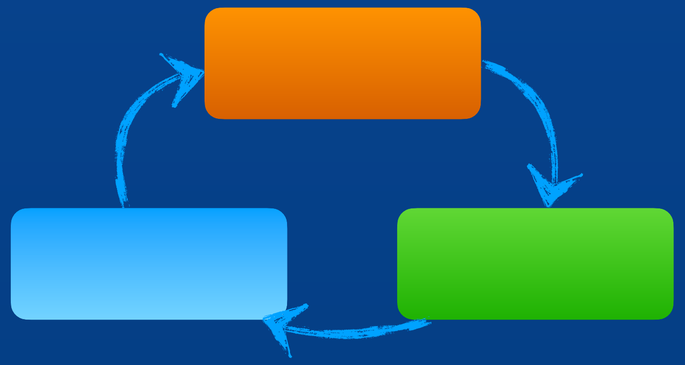


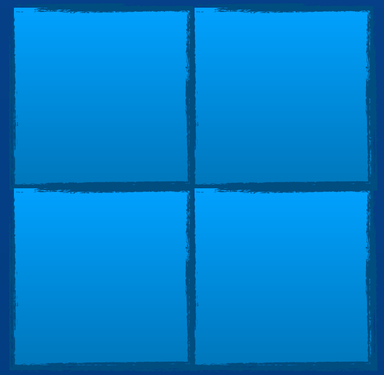
2023

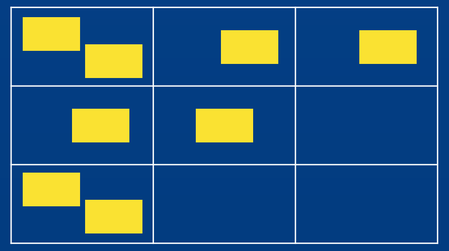


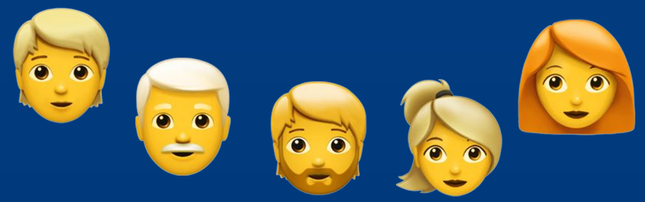
2025

Modelle für Technical Agile Coaching



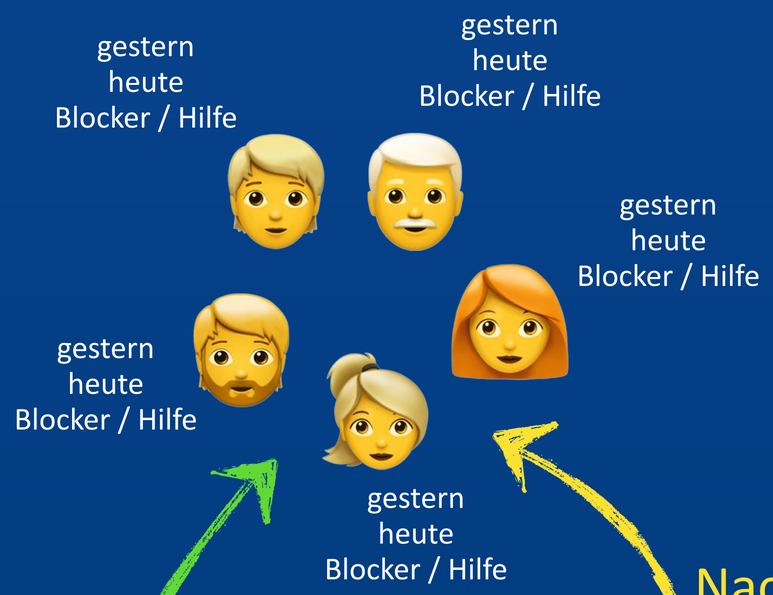






Kommunikation: Daily Standup

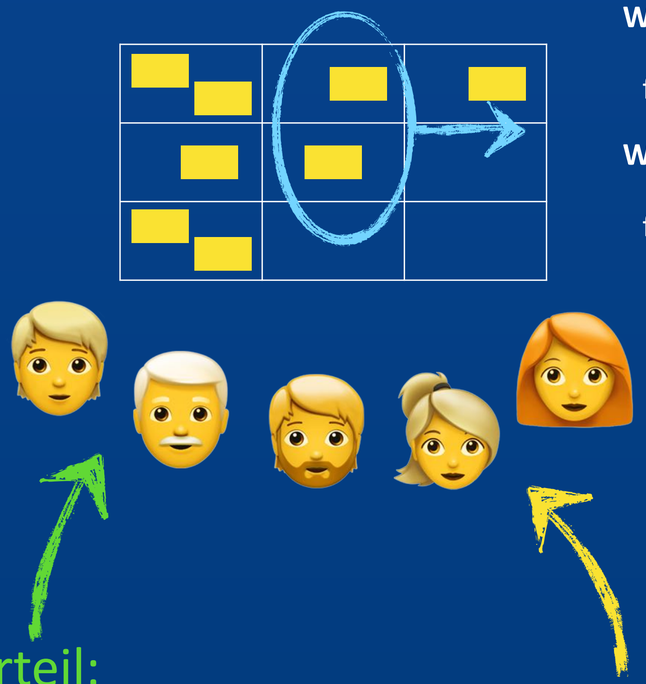
„Walk the People“



Vorteil:
Jeder kommt dran!

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

„Walk the Work“



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

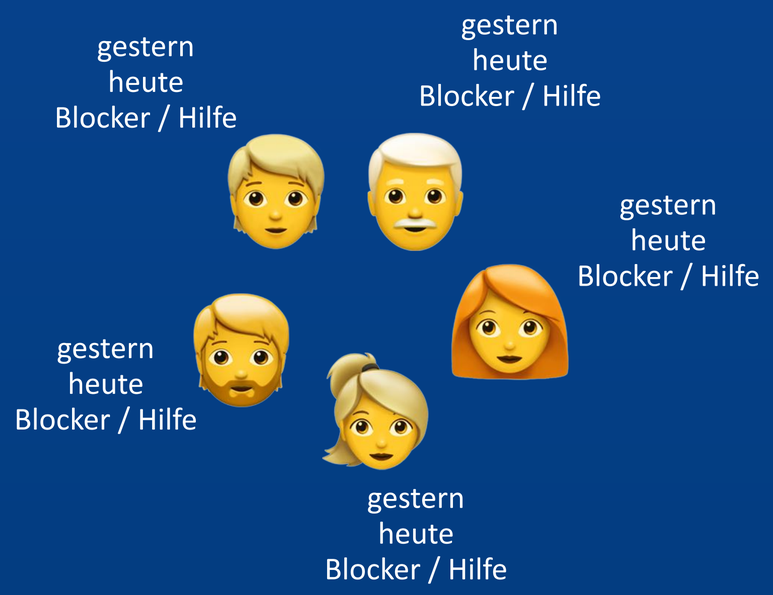
Wer & was ist nötig, um diese Aufgabe zügig fertig zu bekommen?

Vorteil:
Fokus auf „angefangene Arbeit fertig bekommen“

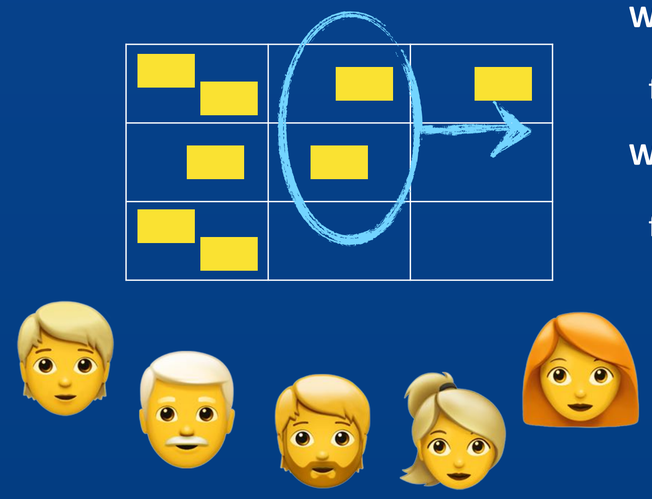
Nachteil:
Was ist mit den eher Stillen?

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

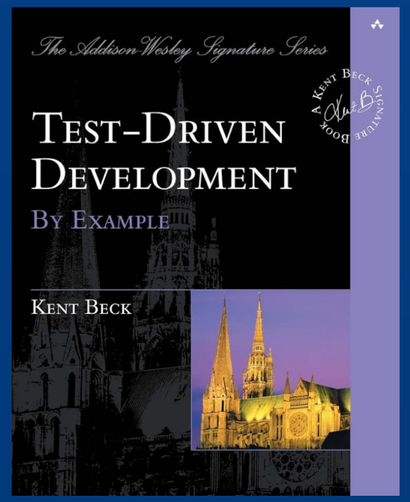
Ist das auch ein gutes Modell für **Legacy Code?**

Red
(neuer Test
für neues Verhalten)

Green
(Implementation)

Refactor
(Struktur verbessern)

2002



Test-Driven Development (TDD)

… in Legacy Code

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

sichere Veränderung & Verbesserung möglich

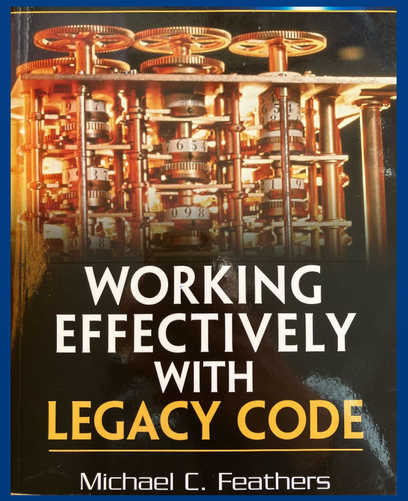
Green
(ungetestet = „korrekt“)

Red
(neuer Approval Test)

Green
(Approval Test)

Change & Refactor
(Implementation)

2005

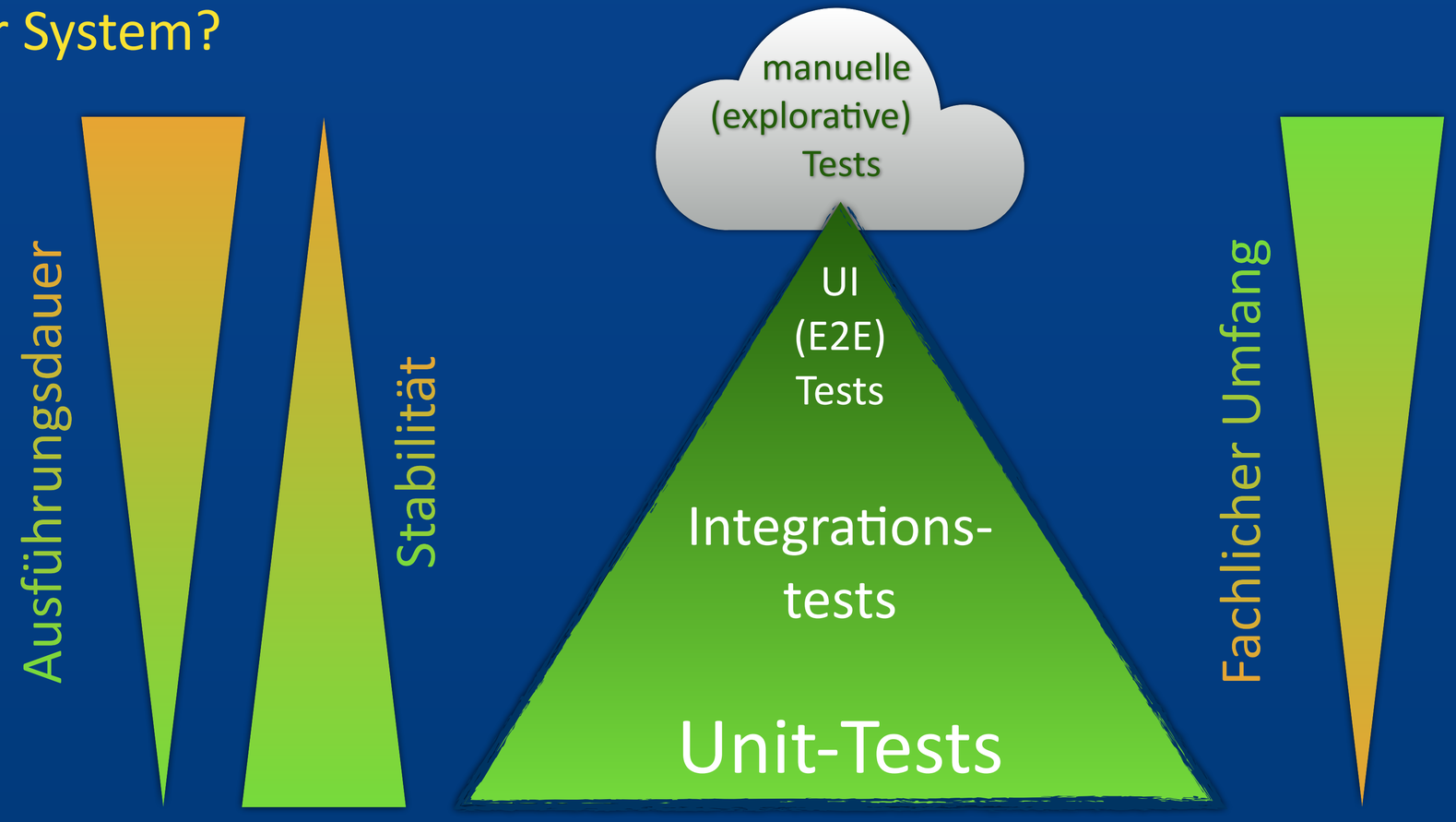


*„Schreibt ganz viele **Unit-Tests**,*

haben wir im letzten Projekt auch so gemacht“

Klassische Testpyramide

Stimmen diese Annahmen für unser System?

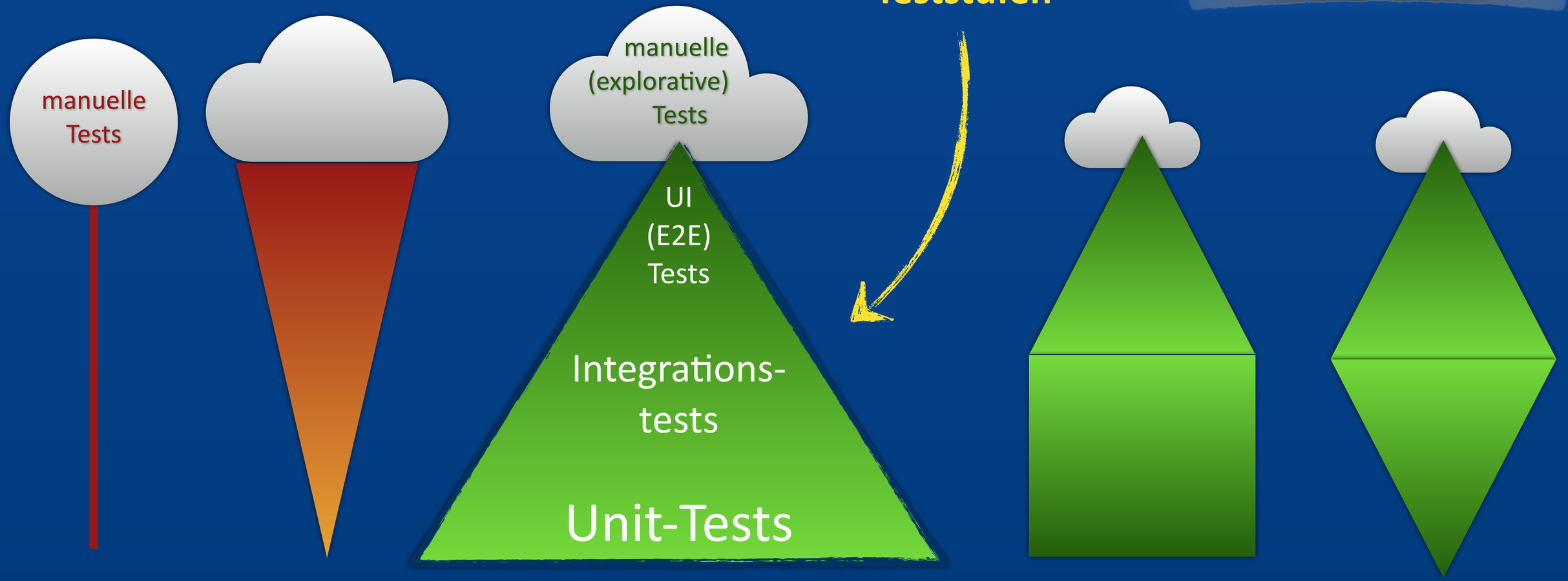


Wie gehen wir mit dem Spannungsfeld „Anzahl Tests“ vs. „fachlicher Umfang“ um?

Test„pyramiden“

Immer noch Kopplung an **Teststufen**

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

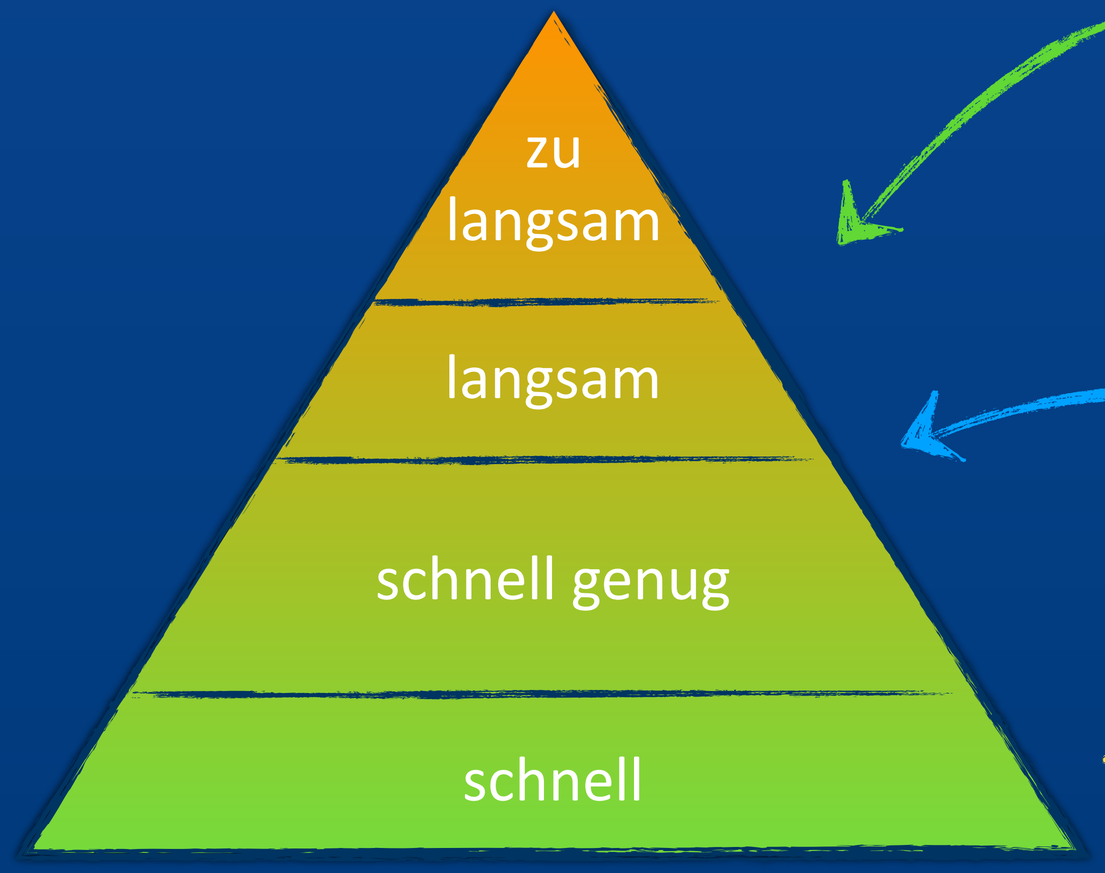


Monolithen

Modulithen & SCS

Microservices

Test**geschwindigkeits**pyramide



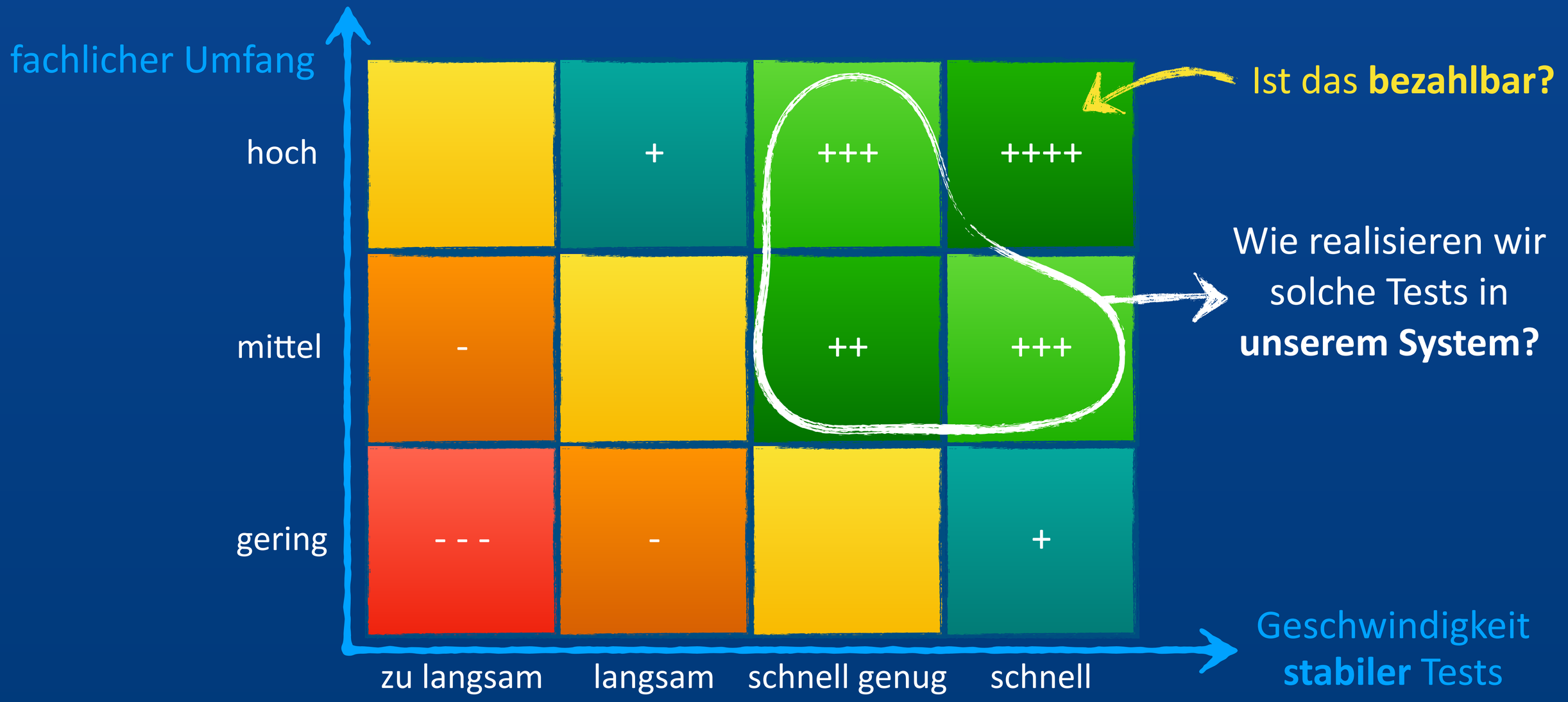
Hilft beim **Beobachten**

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

(Systemintegration/E2E,
System, Integration, Unit)

Sind viele schnelle
Tests **ausreichend?**

Tests: Sicherheitsnetz-Matrix



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

Macht die weg! Nebenbei.“

Technische Schulden

Metapher

+ Zinsen

Kredit aufnehmen → -Zins-Tilgung → Restschuld+Zinsen → -Zins-Tilgung → Restschuld+Zinsen → -Zins-Sondertilgung

Ist jeglicher „suboptimaler“ Code so entstanden?

Wenn nicht, sollten wir evtl. unterschiedlich damit umgehen?

Technical Neglect und **„Cruft“**

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

„klassische“ technische Schulden

Dazu-gelernt!

Modell

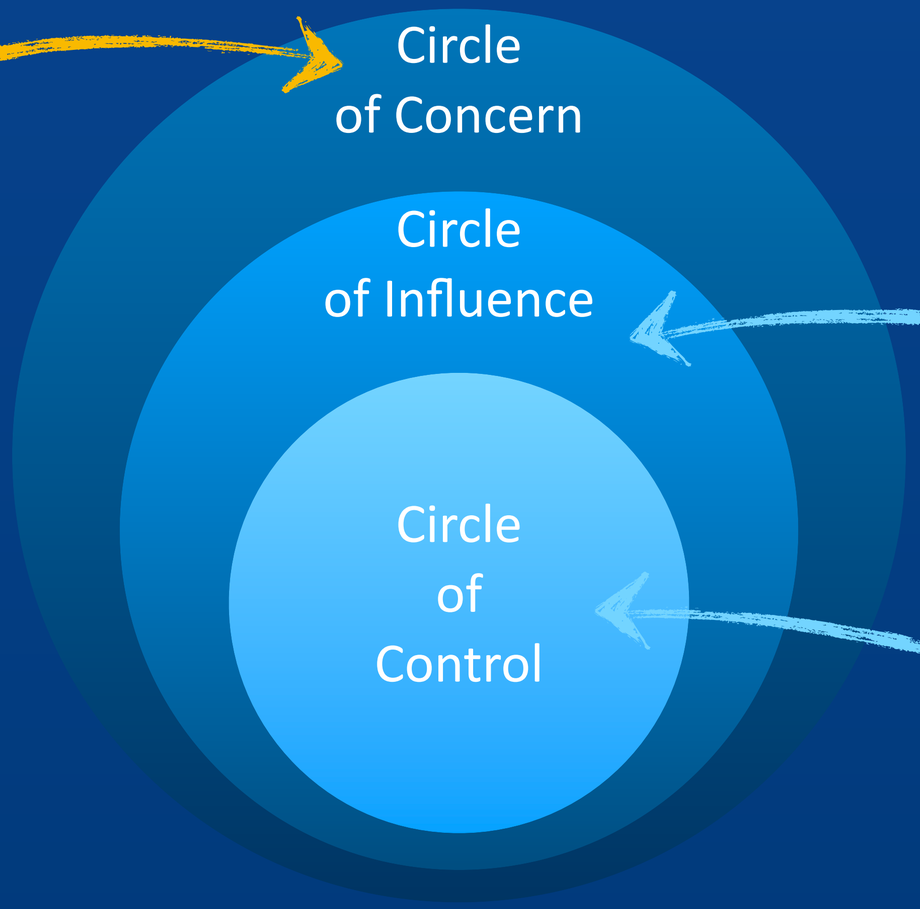
„Technische Schulden“-Metapher von Ward Cunningham
https://martinfowler.com/bliki/TechnicalDebtQuadrant.html

„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

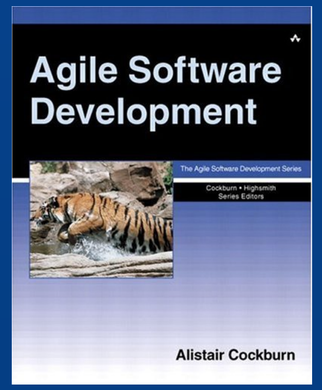
守 破 離

Regeln befolgen

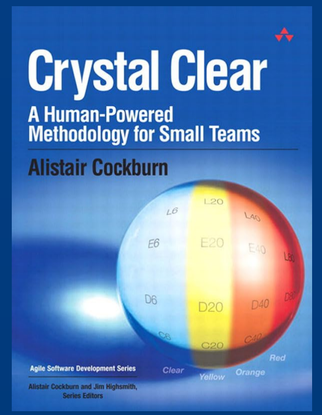
Regeln bewusst brechen

keine Regeln mehr benötigen

2002



2005



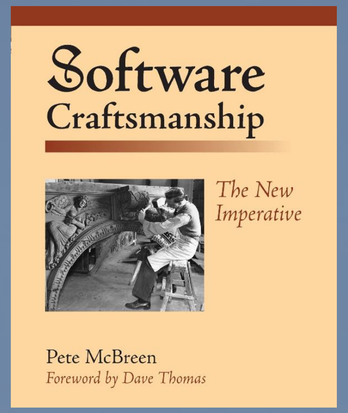
Apprentice – Journeyman – Master

Lehrling – Geselle – Meister

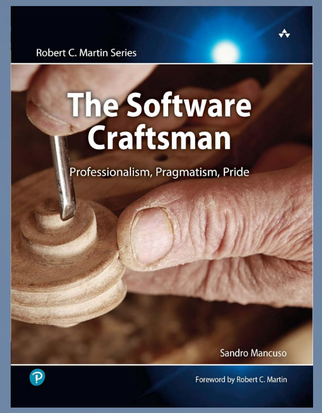
Brauchen Ausbildung.
Dürfen nicht allein gelassen werden!

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!

„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?**

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

Coach **arbeitet mit,** beobachtet und reflektiert

Kick-off

1-2 Wochen Team-Begleitung

Zwischenstand

1-2 Wochen Team-Begleitung

Abschluss

Begleitung bei **normalen Team-Aufgaben.**

Code-Katas nur bei Bedarf.

Workshops mit Team-Programming

„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“

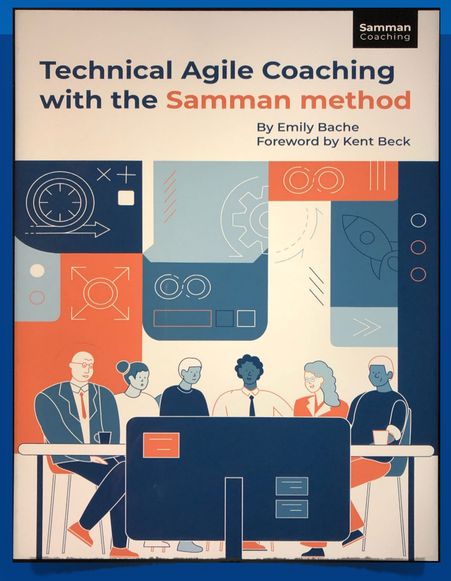
i.d.R. 1 Tag

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“



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

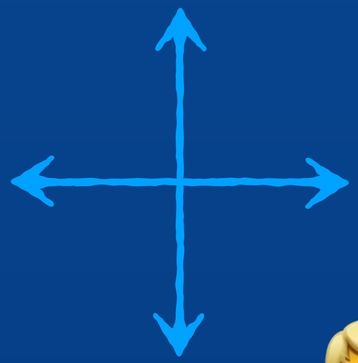
9x

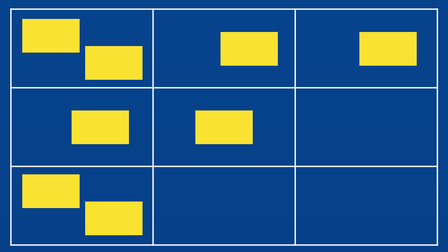
10.

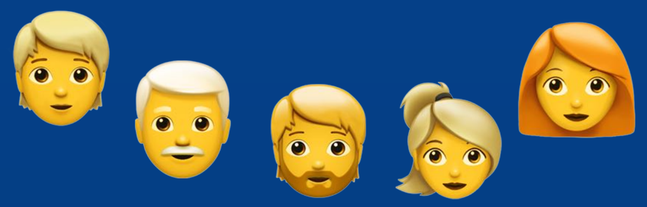
1h **Inspirational Demo** (evtl. Video)

https://sammancoaching.org/

Modelle für Technical Agile Coaching

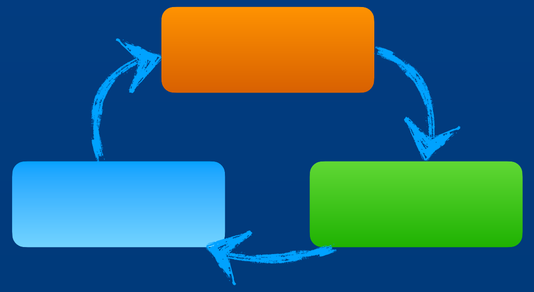






Menschen
Technik
Kontext
verstehen.

Dann erst handeln.



jax by entwickler.de 30 Jahre

Vielen Dank!

@thmuch

Die Techniker

www.tk.de/IT