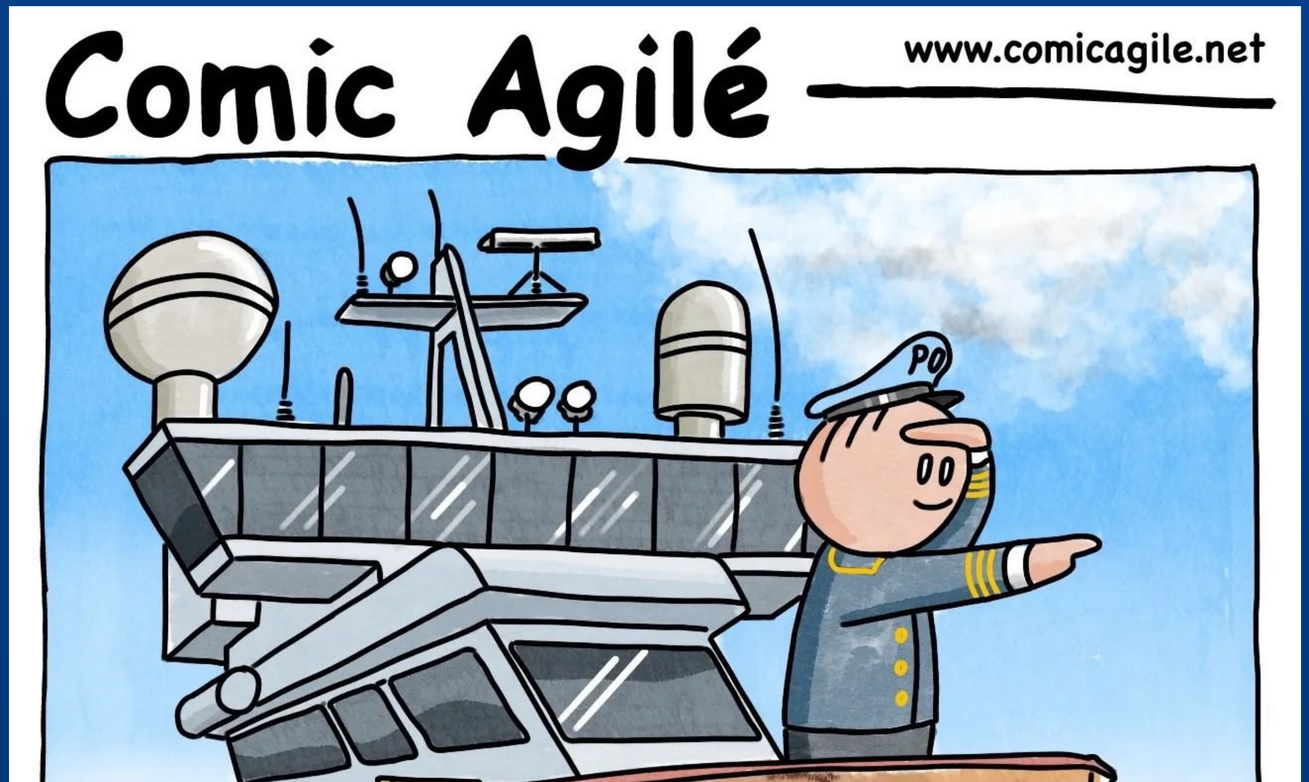
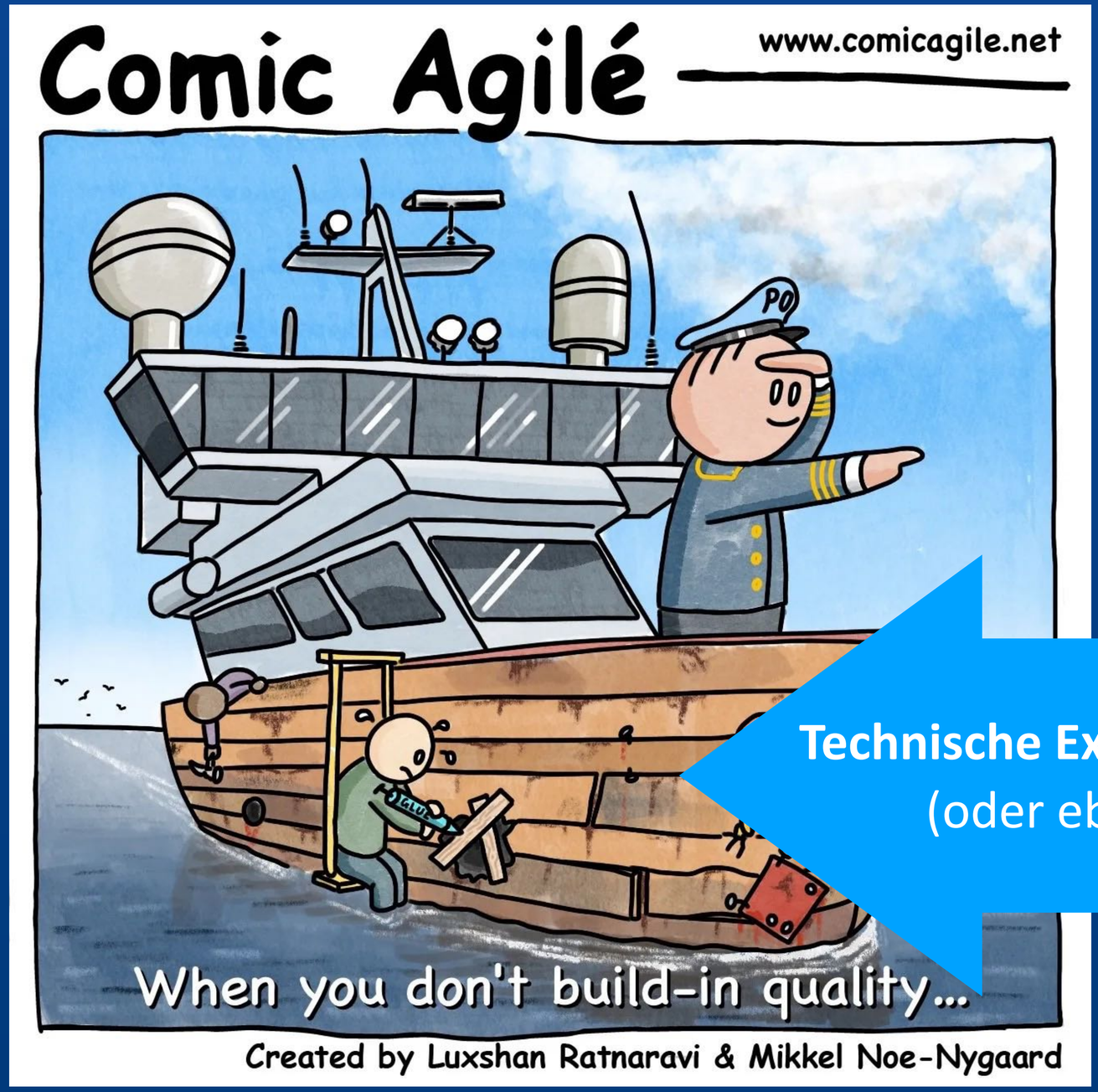
Eine kurze Geschichte der technischen Exzellenz

Thomas Much
@thmuch

27. März 2025

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/

Die Techniker

www.tk.de/IT

Technical Agile Coach

TK

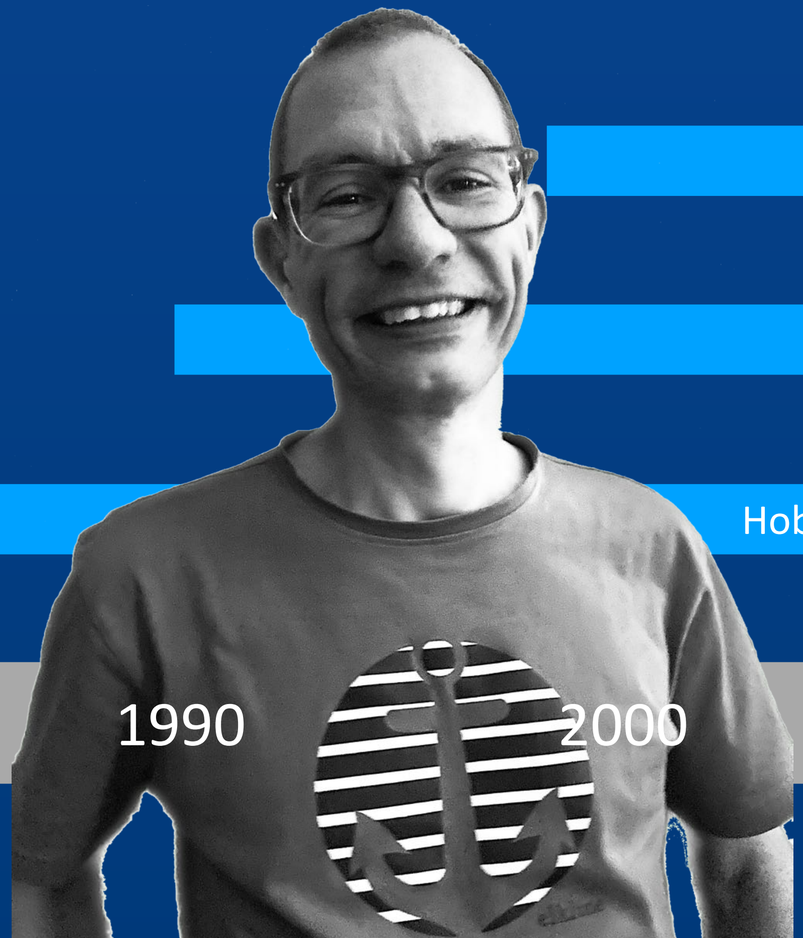
Technical Coaching

Coding Architect

Programmiertrainer

Professionelle Softwareentwicklung

Hobby: Computer, Coding, …



1970 1980 1990 2000 2010 2020

@thmuch

Meilensteine der **technischen Exzellenz?**

Agile Softwareentwicklung
im Unternehmen

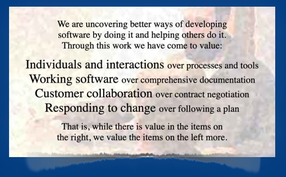
Inside Agile

Manifest für
agile Softwareentwicklung

„Agiles Manifest“

SUnit

JUnit



GenAI

1990 2000 2010 2020 2030

„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

Wer waren die **Autoren**
vom „Agilen Manifest“?

Welchen Hintergrund bzw.
welche **Erfahrungen** brachten sie mit?

Kyle Griffin Aretae
3 Monate · Bearbeitet

Who are/were the authors of the Agile Manifesto

I listed what they do, when I know it.

Mike Beedle -- Scrum + XP
Arie van Bennekum -- DSDM
Alistair Cockburn -- Crystal (and use Cases)
Ward Cunningham -- Dev -- XP
Martin Fowler -- Dev -- XP
Jim Highsmith -- Adaptive
Andrew Hunt -- Dev -- Pragmatic Programmers
Ron Jeffries -- Dev -- XP
Jon Kern -- Dev / Architect -- FDD
Brian Marick -- Tester
Uncle Bob -- Dev - XP
Ken Schwaber -- Dev+PM -- Scrum
Jeff Sutherland -- IT Leader -- Scrum
Dave Thomas -- Dev -- Pragmatic Programmers
Kent Beck -- dev -- XP
James Grenning -- dev -- XP
Steve Mellor -- dev
--
Seems rather heavily Dev + XP weighted, no? ANd how most of them understood you can't do agile decently without the tech parts.

Interesting how very little of Agile does that any more.

116 · 99 Kommentare

XP – Extreme **Programming**

Projekt „C3“ bei Chrysler **1996–1999** (1993-2000)

Kent Beck (JUnit), Ron Jeffries*, Ward Cunningham (WikiWikiWeb, Fit)

Values

Communication
Simplicity
Feedback
Courage
Respect
+ ?

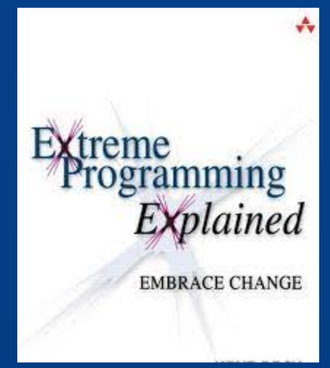
Principles

Humanity
Economics
…
Improvement
Diversity
Reflection
…
Failure
Quality
Baby Steps
Accepted Responsibility

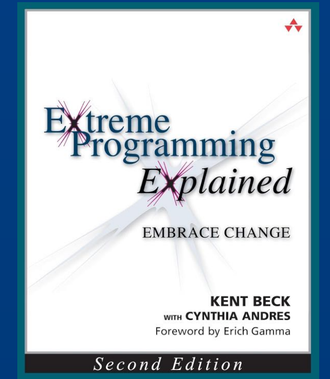
Primary Practices

Sit Together
Whole Team
…
Pair Programming
…
Slack
Ten-Minute Build
Continuous Integration
Test-First Programming
Incremental Design

1999



2004



* https://ronjeffries.com/categories/xprogramming/

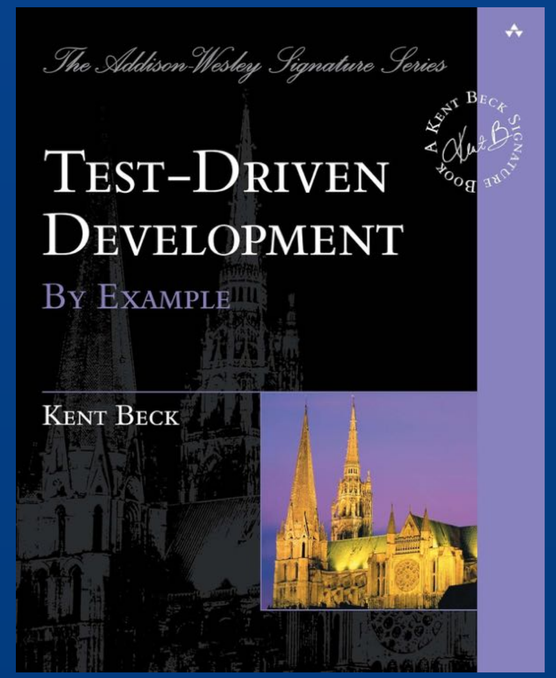
Test-Driven Development (TDD)

Red – Green – Refactor

„Test First“

„wiederentdeckt“ (1950er und 1960er)*

2002

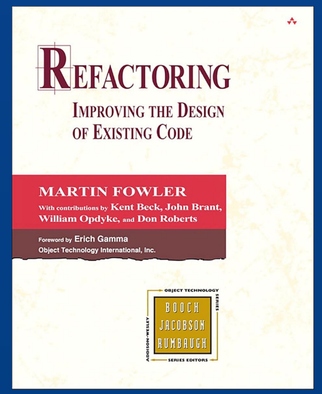


* https://arialdomartini.wordpress.com/2012/07/20/you-wont-believe-how-old-tdd-is/

Refactoring

Veränderung ist der **Normalfall**

1999

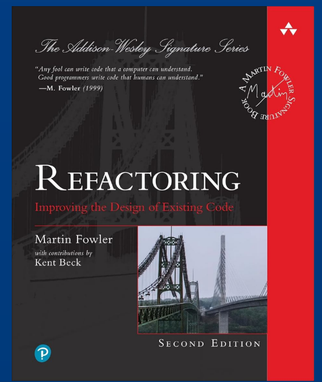


Nicht neu schreiben. Nicht redesignen.

Geordnet umbauen & **aufräumen!** Schritt für Schritt.

Wie sicher ist das in einem ungetesteten Altsystem?

2018



Pragmatic Programming

„**Good Enough** Software“

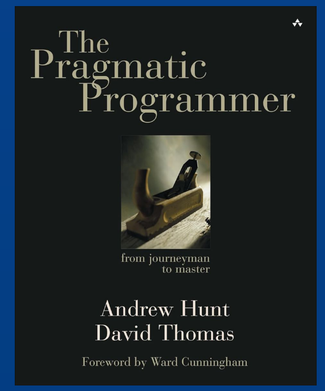
Angemessenheit. Realismus. **Pragmatismus.**

Software-Entropie, „Broken Windows“ & **Sorgfalt**

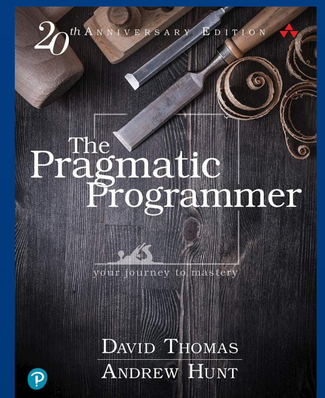
Design, Tools, Testing & Refactoring …

… Sammlung von 100 „Tipps“

1999



2019



Software **Crafting**

Haltung

Engagement

Anspruch

Berufsethos

Ausbildung

Praxis

Gemeinsames arbeiten

Test- & Wartbarkeit

u.a.

https://manifesto.softwarecraftsmanship.org/

Manifesto for Software Craftsmanship

Die Messlatte anheben.

Als engagierte Software-Handwerker heben wir die Messlatte für professionelle Softwareentwicklung an, indem wir üben und anderen dabei helfen, das Handwerk zu erlernen. Durch diese Tätigkeit haben wir diese Werte zu schätzen gelernt:

Nicht nur funktionierende Software,
sondern **auch gut gefertigte Software**

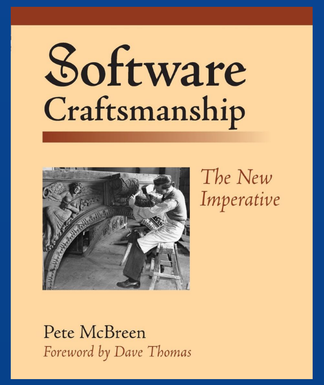
Nicht nur auf Veränderung zu reagieren,
sondern **stets Mehrwert zu schaffen**

Nicht nur Individuen und Interaktionen,
sondern **auch eine Gemeinschaft aus Experten**

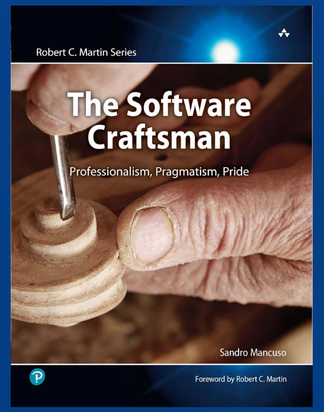
Nicht nur Zusammenarbeit mit dem Kunden,
sondern **auch produktive Partnerschaften**

Das heißt, beim Streben nach den Werten auf der linken Seite halten wir die Werte auf der rechten Seite für unverzichtbar.

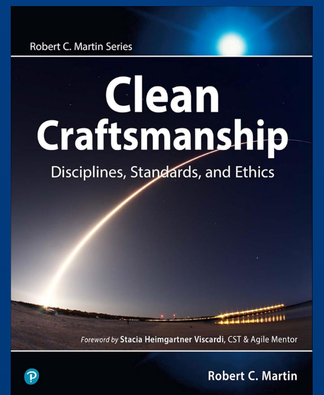
2001



2014



2021



Shu Ha Ri

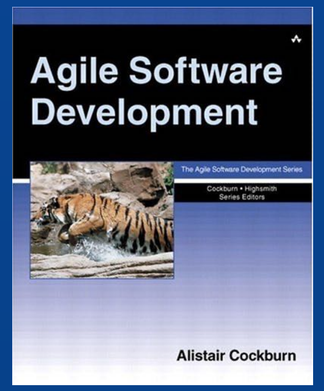
守 破 離

Regeln befolgen

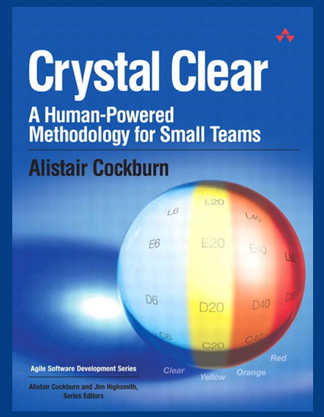
Regeln bewusst brechen

keine Regeln mehr benötigen

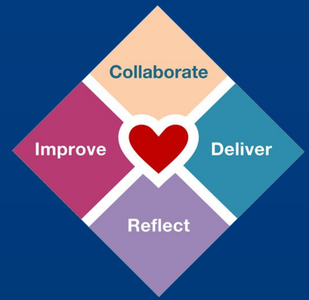
2002



2005



2015



https://heartofagile.com/shu-ha-ri/

Apprentice – Journeyman – Master

Lehrling – Geselle – Meister

„Viel zu lernen du noch hast“

– Meister Yoda zu seinem Padawan

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

„Es ist **noch kein Meister**
vom Himmel gefallen!“

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

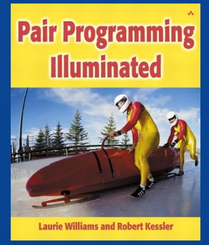
About the Authors
About the Manifesto

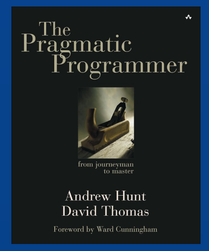
Manifesto for Agile Software Development

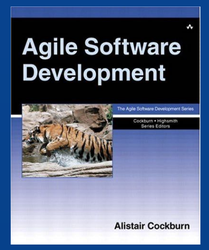
We are uncovering better ways of developing
software by doing it and helping others do it.
Through this work we have come to valu

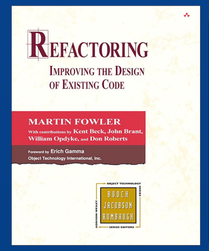
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

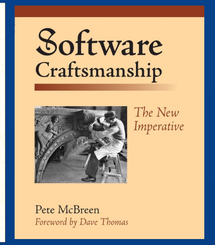
That is, while there is value in the items on
the right, we value the items on the left more.

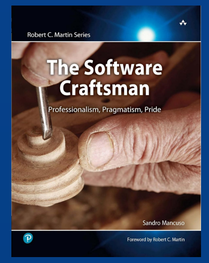


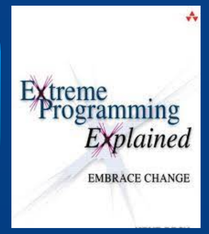


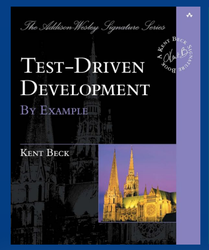


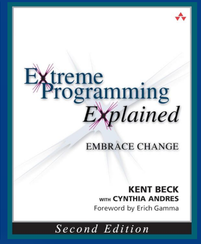


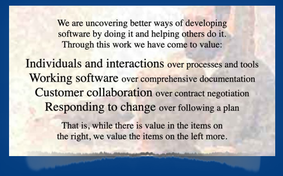












Projekt „C3“

Wiki Wiki Web

SUnit

JUnit

GenAI

1990

2000

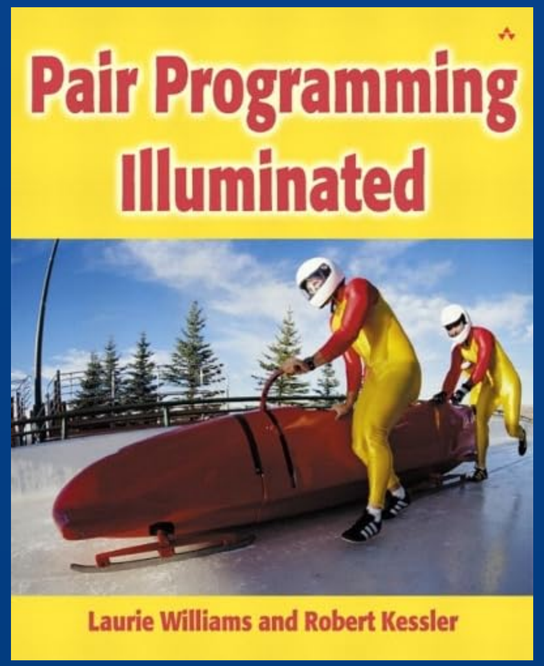
2010

2020

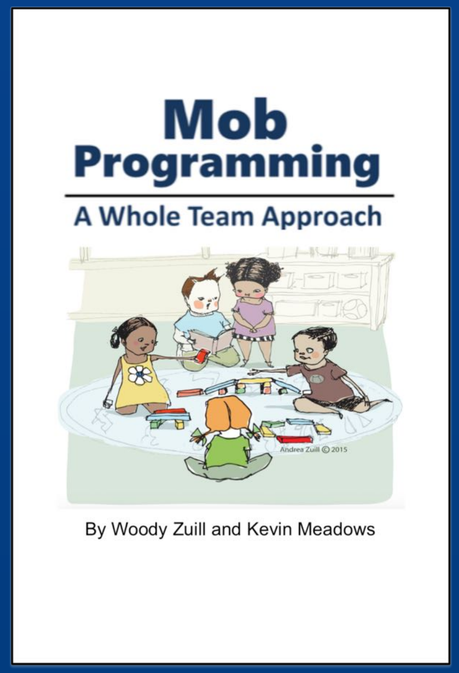
2030

Als Team liefern & **lernen**

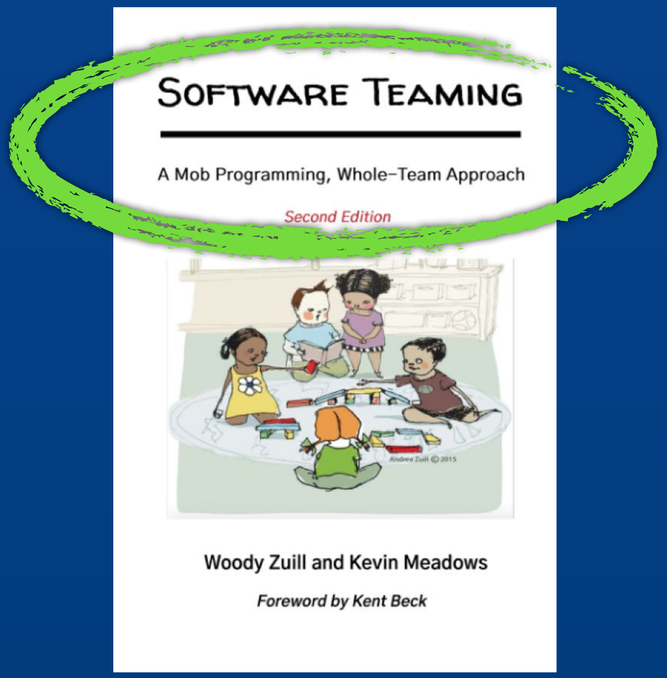
2002



2013

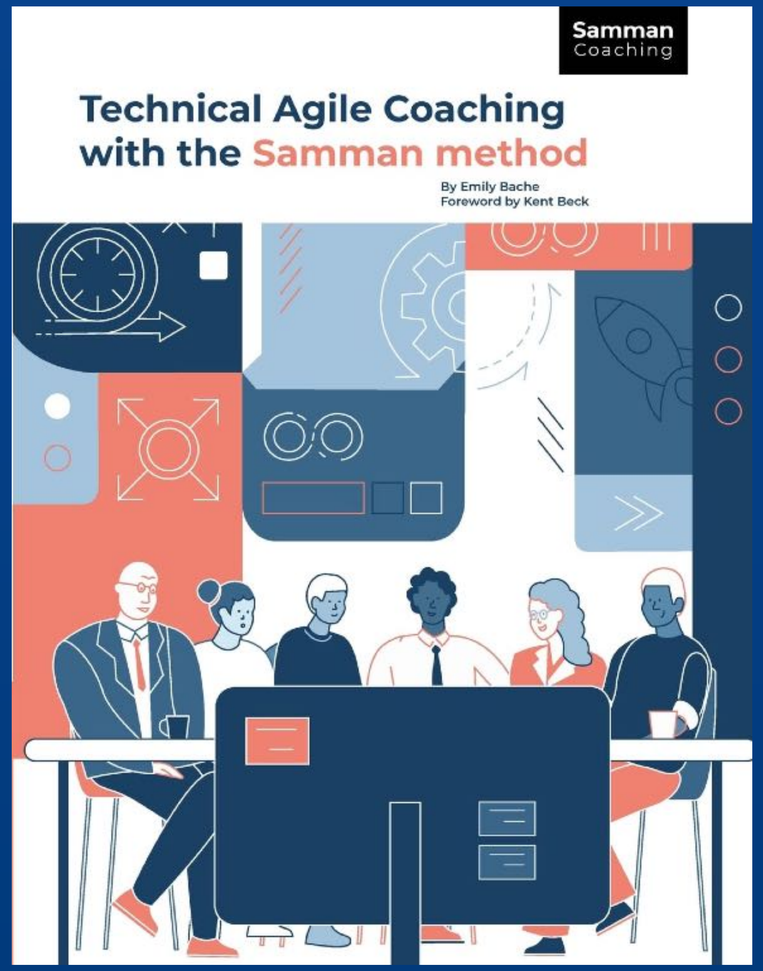


2022



Whole Team Programming. Ensemble Programming. **Team Programming.**

Lernen **skalieren**

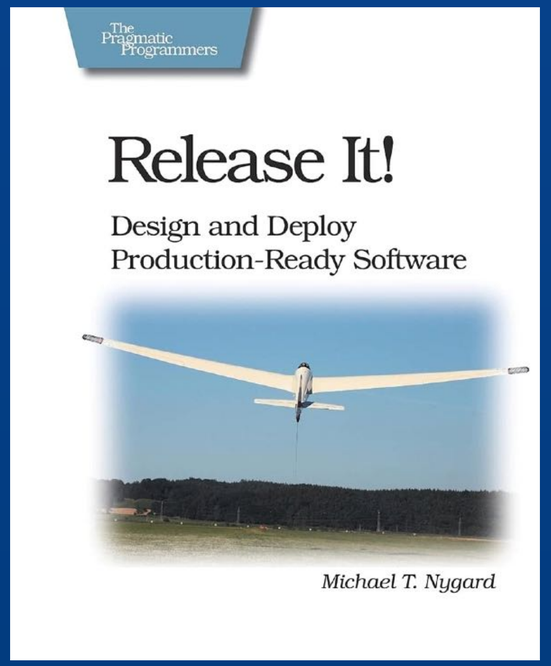


2019

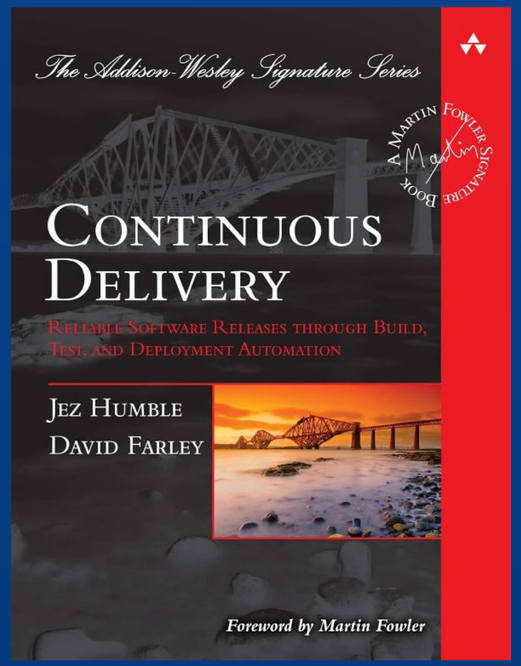
https://sammancoaching.org/

Exzellent bauen – und **ausliefern**

2007



2010



2018

Warum ist das alles **relevant?**

DevOps Research and Assessment (**DORA**)

Metriken & **Fähigkeiten**

Capabilities that enable a
Climate for Learning

Code maintainability
Documentation quality
Empowering teams to choose tools
Generative organizational culture
Job satisfaction
Learning culture
Team experimentation
Transformational leadership
Well-being

Capabilities that enable
Fast Flow

Continuous delivery
Database change management
Deployment automation
Flexible infrastructure
Loosely coupled teams
Streamlining change approval
Trunk-based development
Version control
Visual management
Work in process limits
Working in small batches

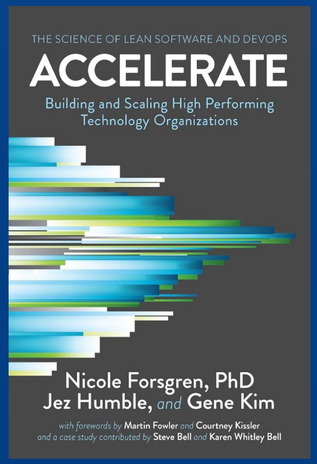
Capabilities that enable
Fast Feedback

Continuous integration
Customer feedback
Monitoring and observability
Monitoring systems to inform business decisions
Pervasive security
Proactive failure notification
Test automation
Test data management
Visibility of work in the value stream

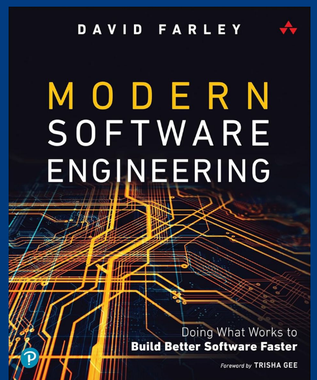
https://dora.dev/capabilities/ https://www.puppet.com/resources/history-of-devops-reports

2013

2018



2022



Zukunft **AI?**

Wie sieht technische Exzellenz dann aus? Geschwindigkeit vs. Qualität?

Wer konzipiert? Wer implementiert? Kann man das noch trennen?

Wer hat das **Wissen**? Wer das **Können**? Welche Skills?

Wer schreibt die Tests?

Wird das Schreiben **ausführbarer Spezifikationen** zum essenziellen Skill?
AI kann dann die ausführbare Spezifikation implementieren.

Es geht um

Softwareentwicklung. Gemeinsam.

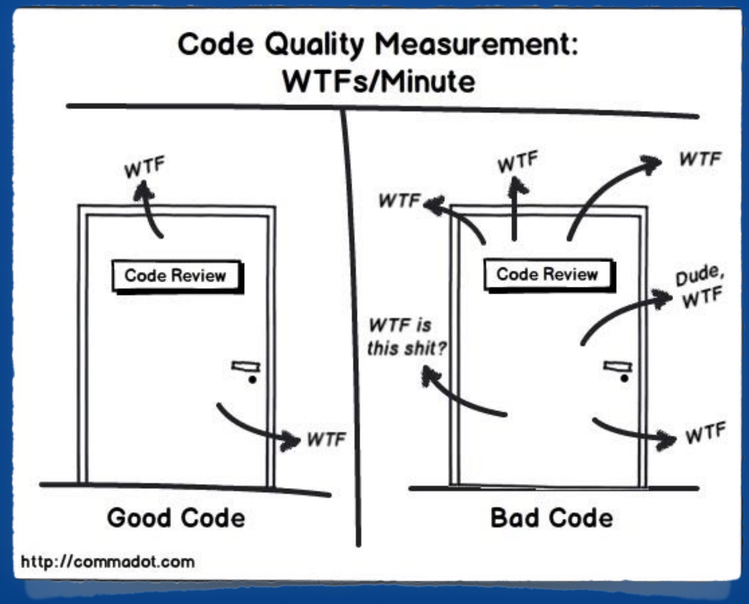
Wissen & Können. Lernen.

Haltung & Anspruch.

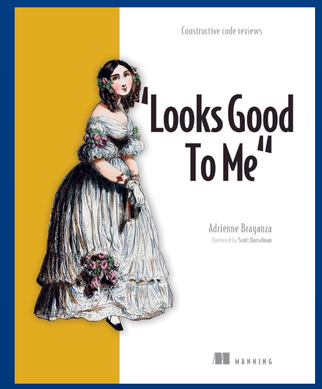
Qualität. Verständlichkeit. Wartbarkeit.

Technische Exzellenz.

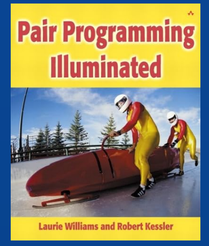
https://commadot.com/wtf-per-minute/

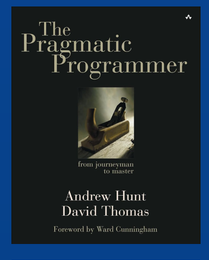


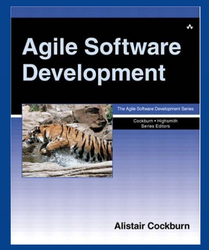
2025

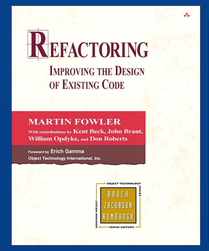


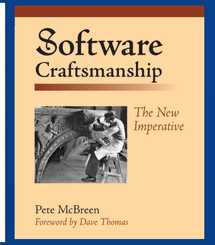
Wissen gemeinsam erleben

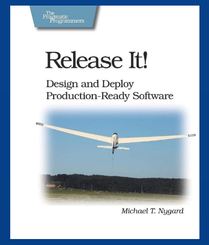


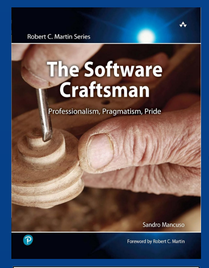


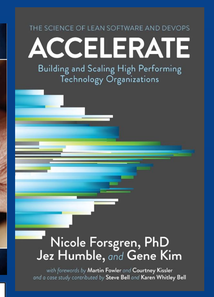


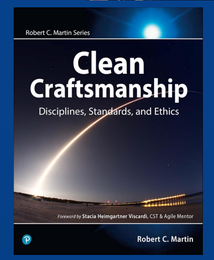


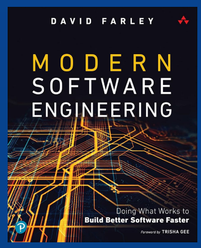


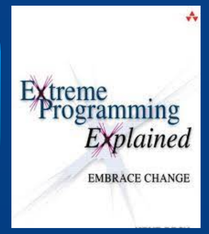


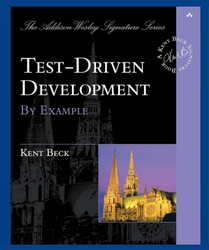


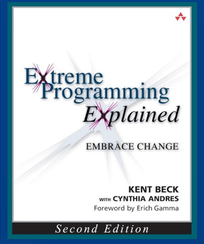


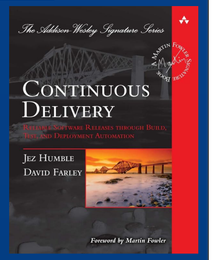


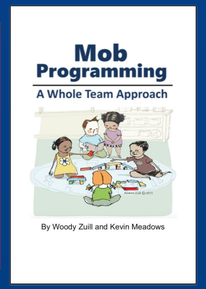


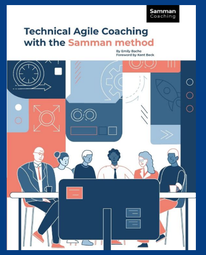


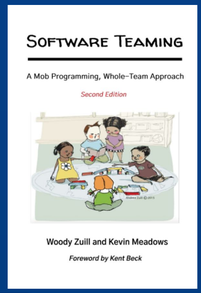


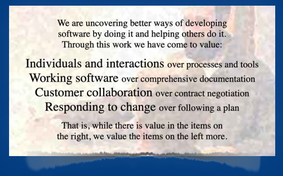












Projekt „C3“

Wiki Wiki Web

SUnit

JUnit

GenAI

1990 2000 2010 2020 2030

Die Techniker

www.tk.de/IT

inside agile

Vielen Dank!

@thmuch

"Any fool can write code that a computer can understand.

Good programmers write code that humans can understand."

– Martin Fowler (1999)