Modern Software Development

DORA All the Way or What?

Thomas Much, Techniker Krankenkasse

WeAreDevelopers World Congress

Berlin, 11 July 2025

@thmuch

1984

"Works on my machine (only)"

2025

"Always in a releasable state, everywhere"

Modern Software Development

High quality, built-in

Fast feedback on **customer value**

Continuous software development

TL'DR

DORA capabilities super important!
resp. **underlying software development practices & skills**

Most probably **even in the age of (Gen)AI**

Get to know them. Learn them. Use 'em.

DORA

👨‍💼 👩‍💼 🏦

DORA

Since 2022
..2025 (EU)

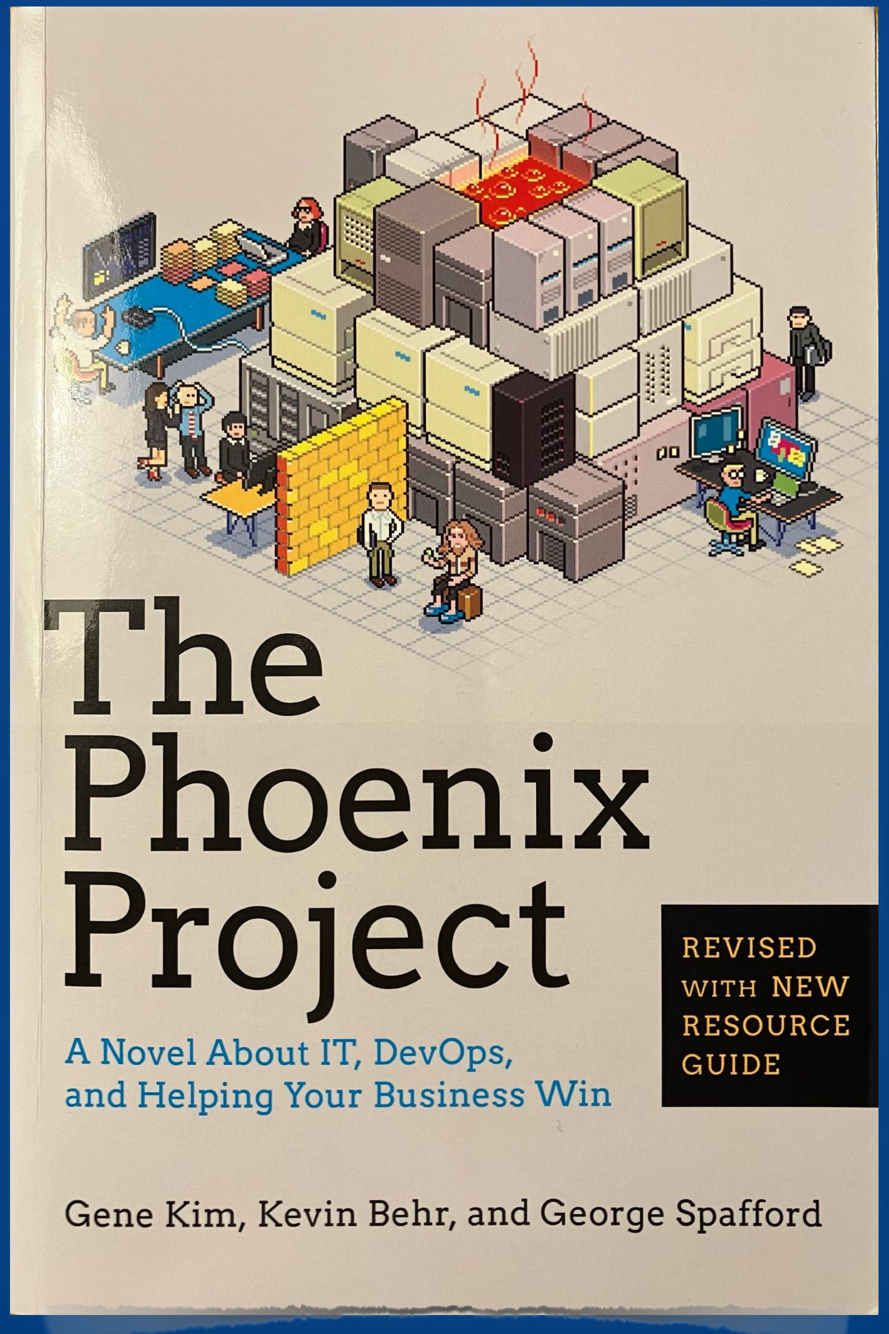
Digital Operational Resilience Act

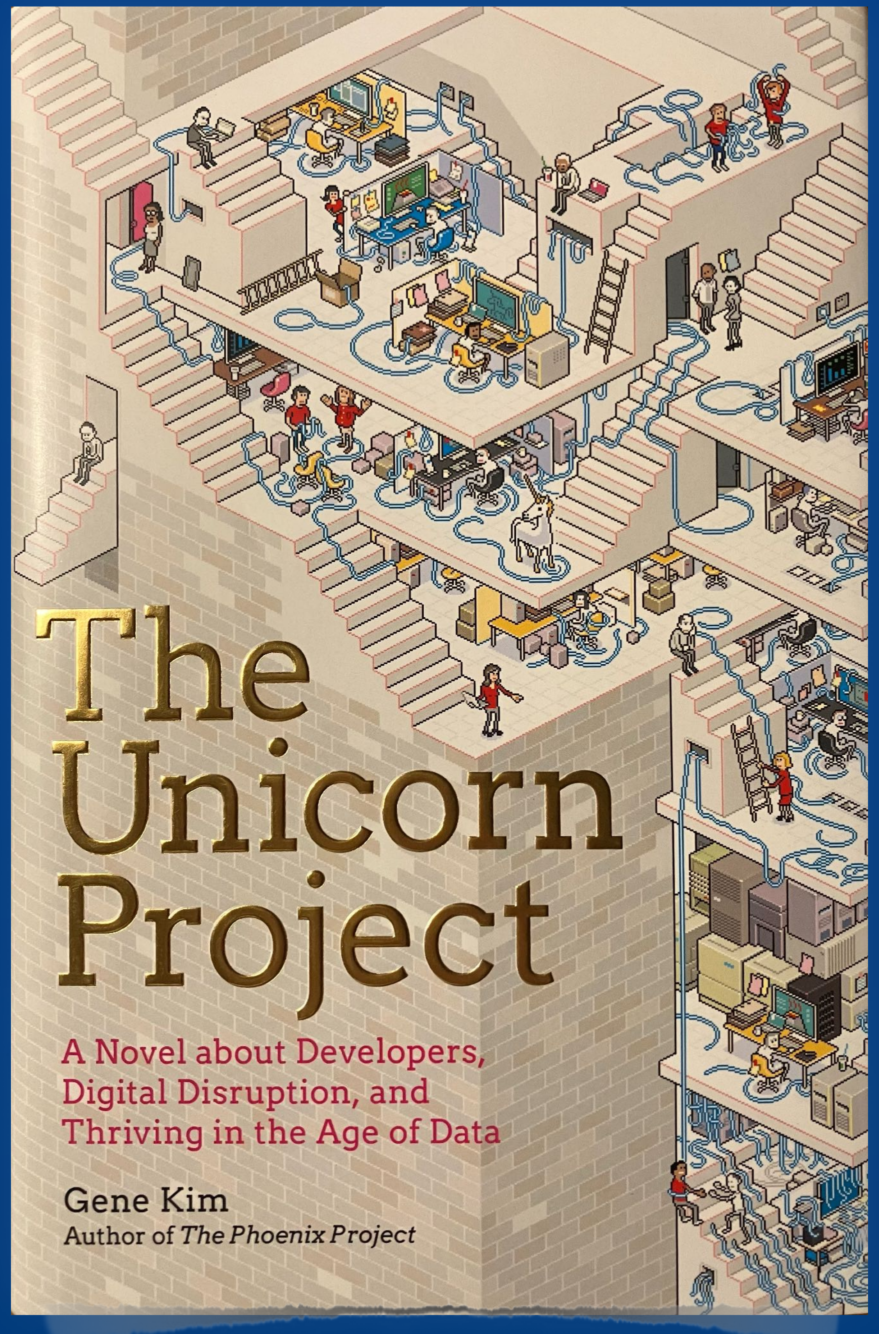
DevOps Research and Assessment

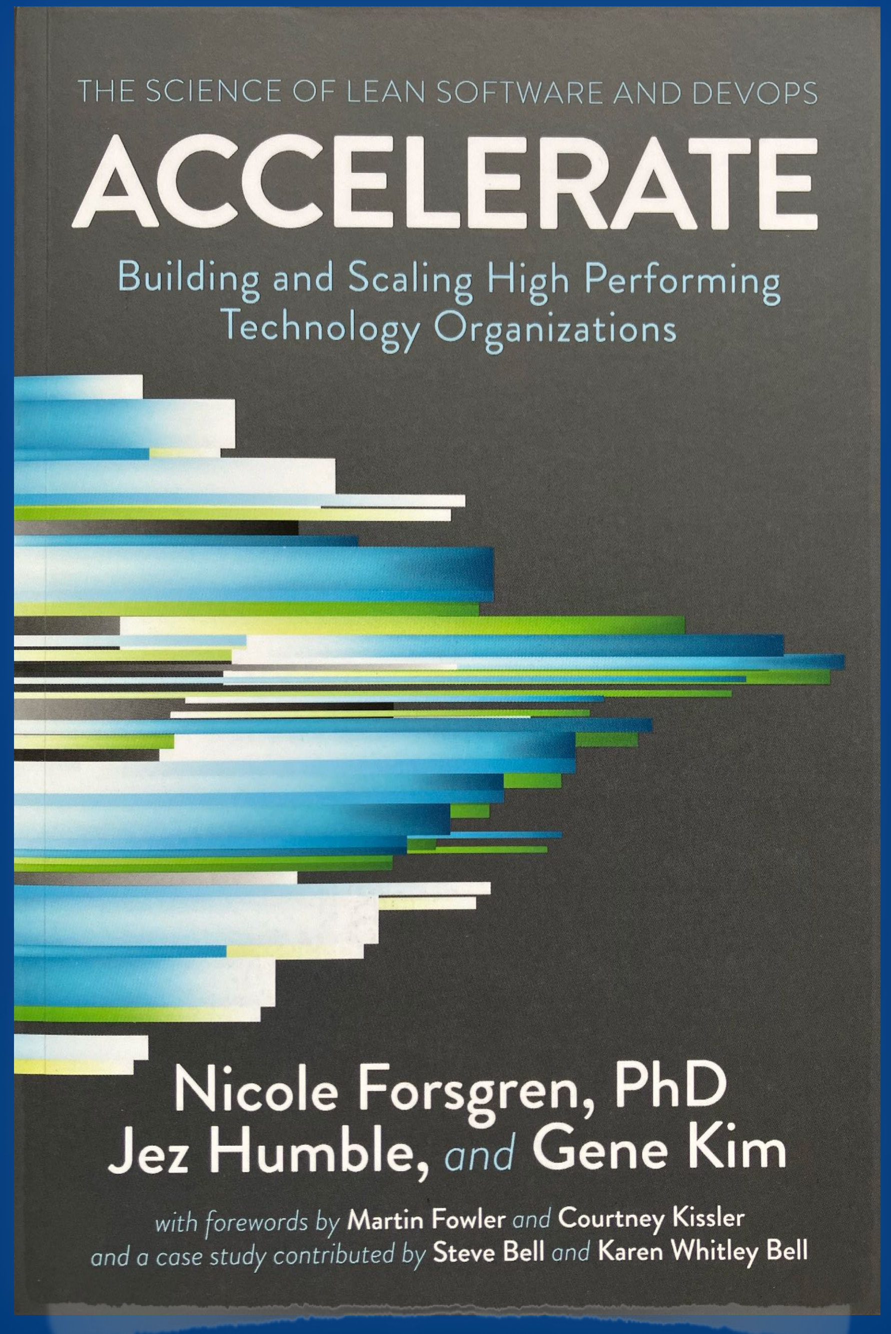
DORA

Since 2015

Once Upon a Time …


The Phoenix Project
REVISED WITH NEW RESOURCE GUIDE
A Novel About IT, DevOps, and Helping Your Business Win
Gene Kim, Kevin Behr, and George Spafford


The Unicorn Project
A Novel about Developers, Digital Disruption, and Thriving in the Age of Data
Gene Kim
Author of The Phoenix Project


THE SCIENCE OF LEAN SOFTWARE AND DEVOPS
ACCELERATE
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim
with forewords by Martin Fowler and Courtney Kissler
and a case study contributed by Steve Bell and Karen Whitley Bell

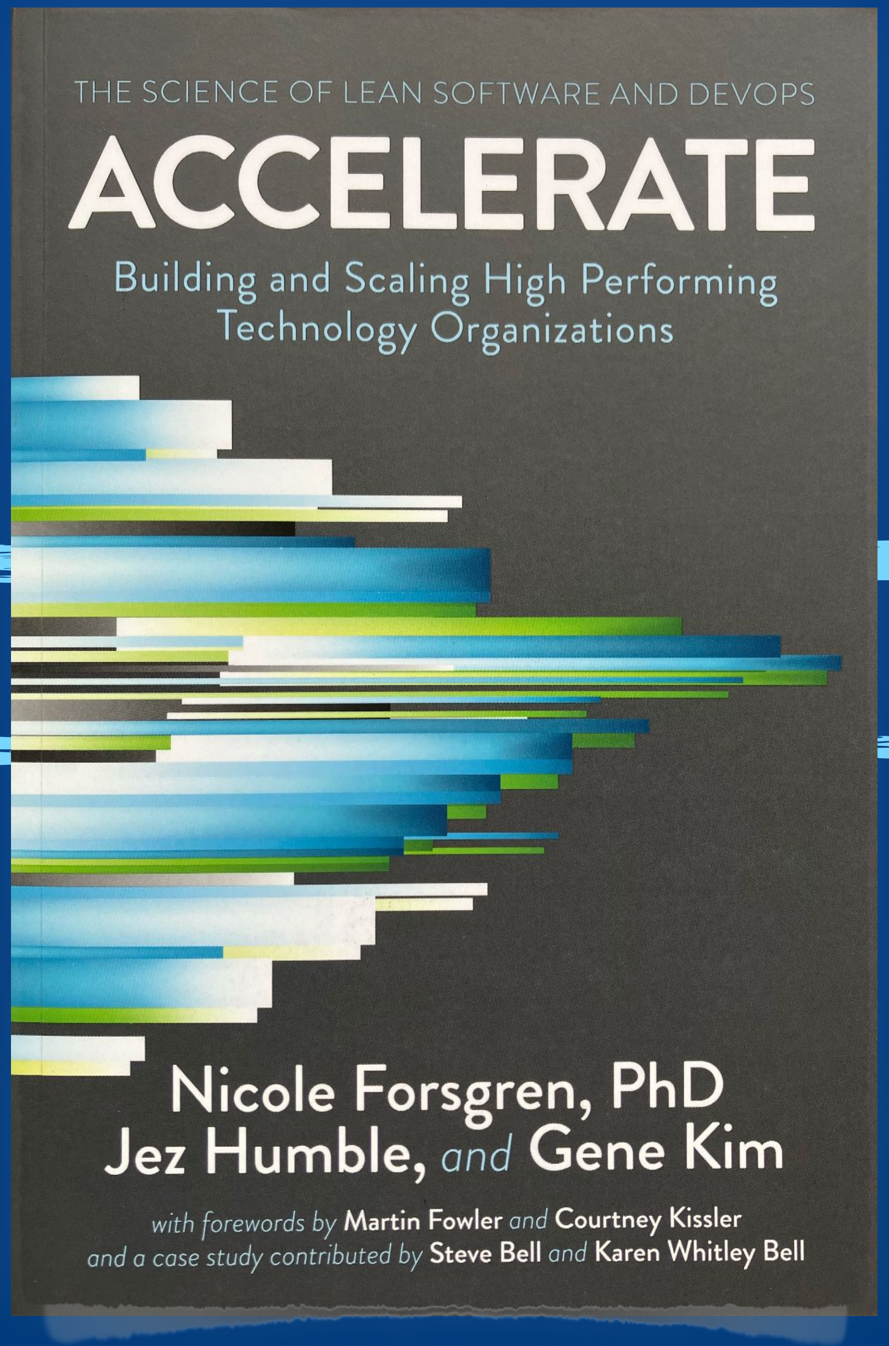
State of DevOps Report

(since 2013)

DORA

(2015)

2018



THE SCIENCE OF LEAN SOFTWARE AND DEVOPS

ACCELERATE

Building and Scaling High Performing Technology Organizations

Nicole Forsgren, PhD
Jez Humble, *and* Gene Kim

with forewords by Martin Fowler *and* Courtney Kissler
and a case study contributed by Steve Bell *and* Karen Whitley Bell

Latest report
2024

https://dora.dev/

High Performing Teams

High Performing Organizations

Smarter. Not Harder.

What do

High Performing Teams

High Performing Organizations

have **in common?**

Survey

(every year)

DORA Four Key Metrics

- Change lead time
- Deployment frequency
- Change fail percentage
- Failed deployment recovery time

https://dora.dev/guides/dora-metrics-four-keys/

DORA Core Model

Capabilities

predict

Performance

Software delivery

Reliability (SLOs)

predicts

Outcomes

Organizational performance

Well-being

https://dora.dev/research/

DORA Core Model Predictions

Capabilities

🎯 ✅

predict

Performance

❌

🎯 ✅

? ✅

predicts

predicts

predicts

Outcomes

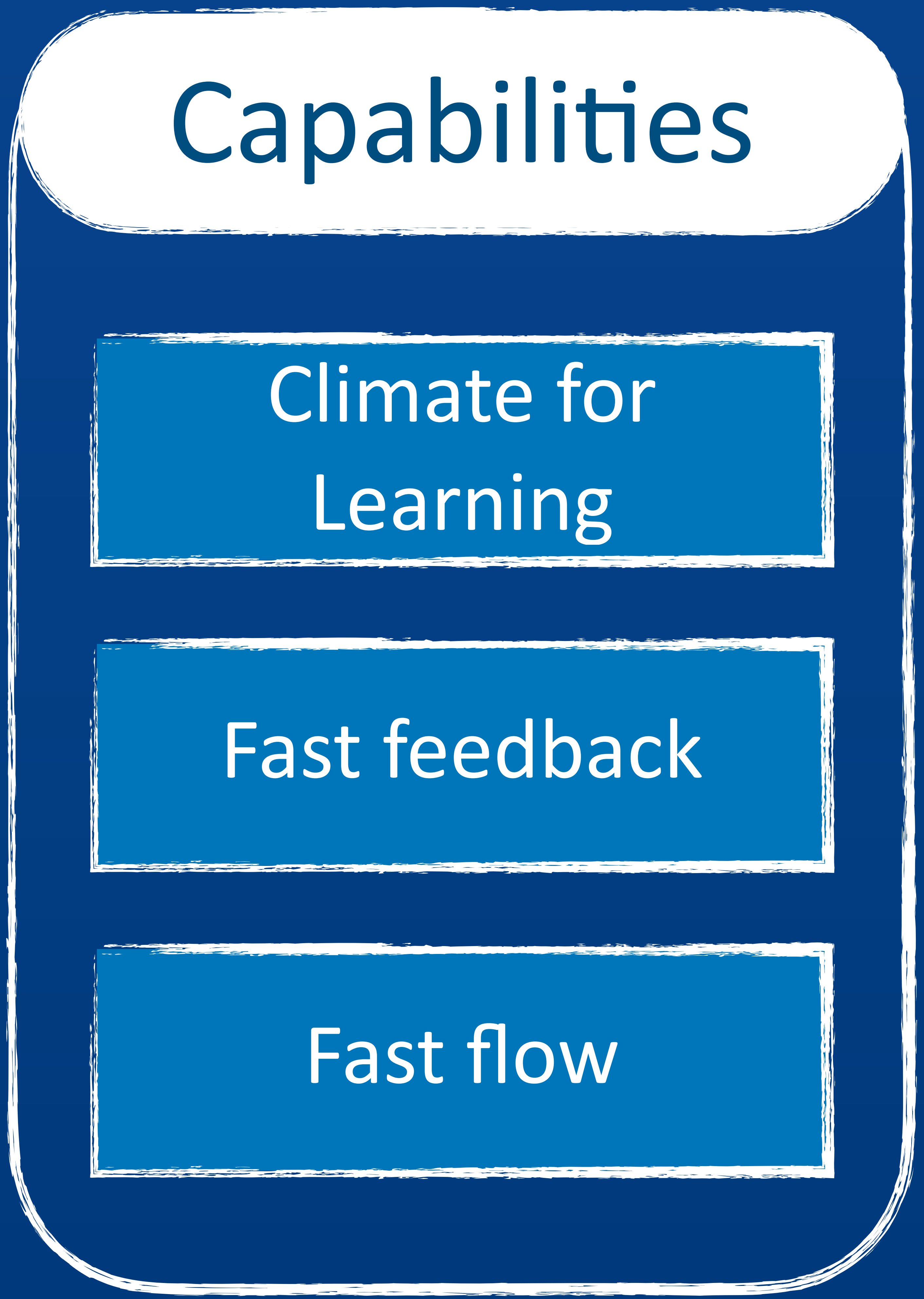
? ❌

???

? ✅

https://dora.dev/research/

DORA Capabilities



Continuous Delivery

Test automation

Monitoring and observability

Documentation quality

Empowering teams to choose tools

Well-being

…

29 capabilities (2024)

https://dora.dev/capabilities/

Let's have a look at

selected

capabilities

Version Control

Why?

Reproducibility

Traceability

(auditors love this!)

dora.dev/capabilities/version-control/

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

Version Control

Source code 🥱

Build & test scripts

Infrastructure configuration

Database schema

Everything 🎖️🏆

(auditors love this even more!)

dora.dev/capabilities/version-control/

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

Version Control

> "… **full version control** is that
> I should be able to walk up with a very minimally configured environment […] and **be able to easily build, and run the product after cloning the repository**."
>
> – Martin Fowler

martinfowler.com/articles/continuousIntegration.html

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

Continuous Integration (CI)

"We have a CI server" ❌

Well, yes, sure, but …

CI is not a tool.

dora.dev/capabilities/continuous-integration/

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

Continuous Integration (CI)

Every commit triggers a build

Every commit triggers **automated tests** that provide **feedback within minutes**

Keeping the build green is highest priority

Integrate all work **into the main line** **at least daily** 🏅🏆

dora.dev/capabilities/continuous-integration/

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

Branching Strategies for CI

Don't branch 😁

Short(est)-lived branches 🤷‍♂️

https://pixabay.com/de/photos/wald-b%C3%A4ume-ge%C3%A4st-baumkrone-238379/

Trunk-Based Development

Three or fewer active branches.

Merge branches **to trunk** at least once a day.

No code freezes. No integration phases.

Comprehensive automated testing.

Synchronous code reviews.

Work in small batches.

dora.dev/capabilities/trunk-based-development/

https://pixabay.com/de/photos/wald-b%C3%A4ume-ge%C3%A4st-baumkrone-238379/

Continuous Delivery (CD)

Ability to deploy to production **at any time, on demand.**

Not necessarily automatically. **But all steps automated.**

Increases flexibility. Minimizes risk.

Builds on many other capabilities!

dora.dev/capabilities/continuous-delivery/

https://pixabay.com/de/photos/paket-postdienst-lieferung-postamt-7225976/

Architecture

Microservices? Monolith?

Deployment strategy **does not matter that much.**

Loose coupling makes the difference.

dora.dev/capabilities/loosely-coupled-teams/

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

Loose Coupling

Team can change design of their systems without permission or dependency on other teams.

Team can complete work without fine-grained communication or coordination with people outside the team.

Team can deploy and release independently of surrounding systems.

Team can do most of their testing w/o an integrated test environment.

Team deploys during normal business hours.

dora.dev/capabilities/loosely-coupled-teams/

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

Is This Really New?

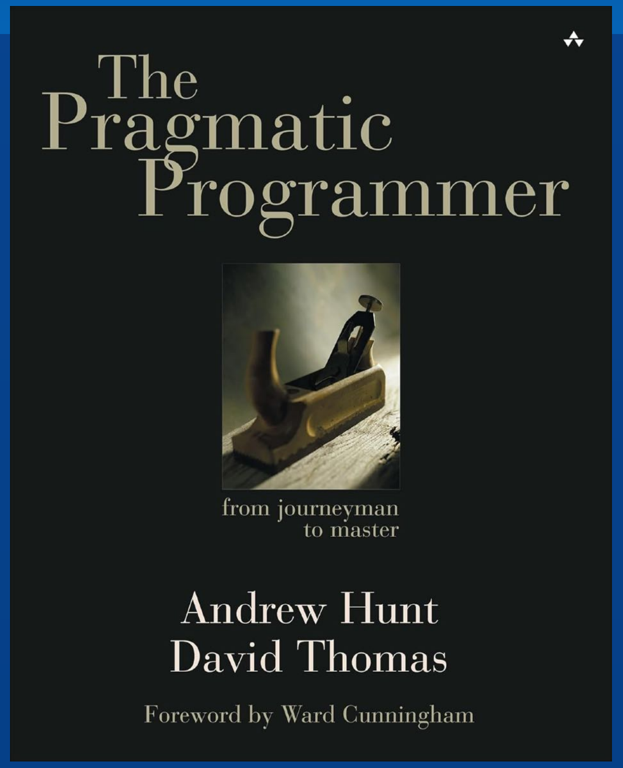
Not at all.

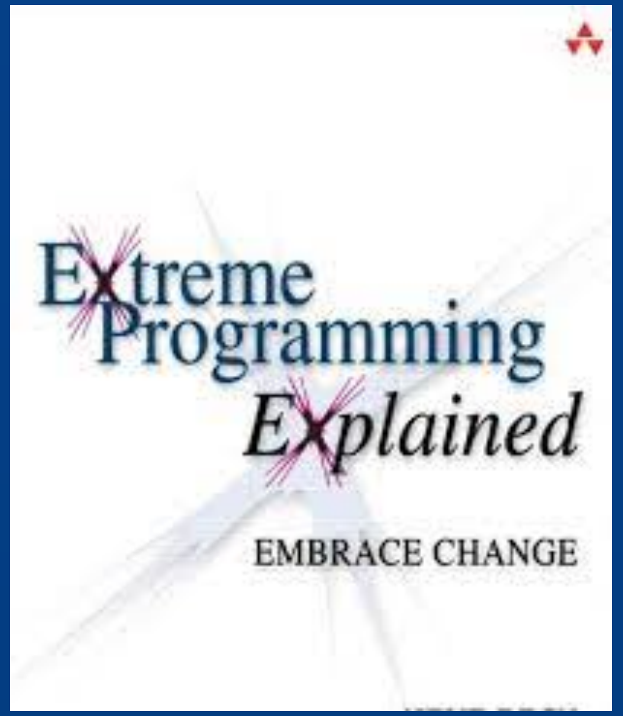
Remember Extreme Programming (XP)?

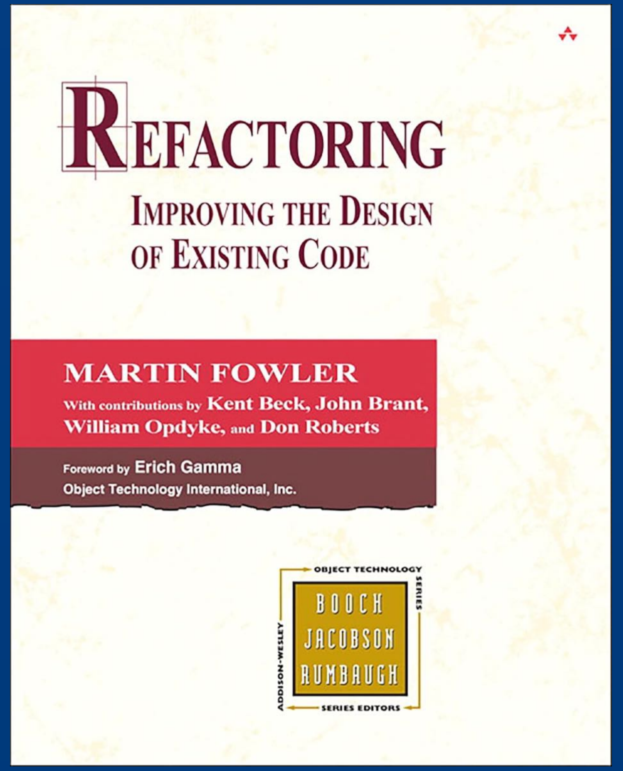
Good practices. Worst naming ever.

History

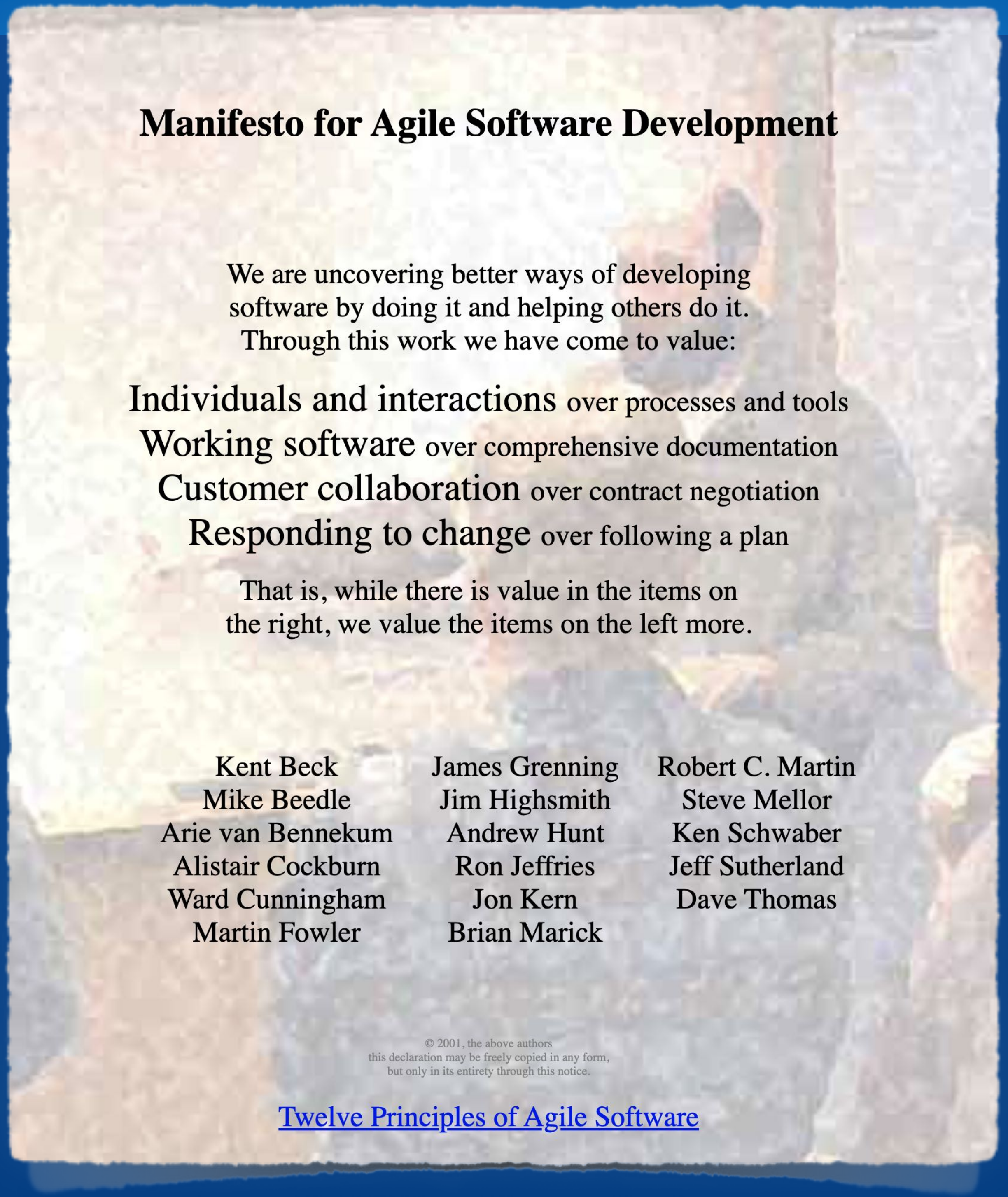
1999






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

2001



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

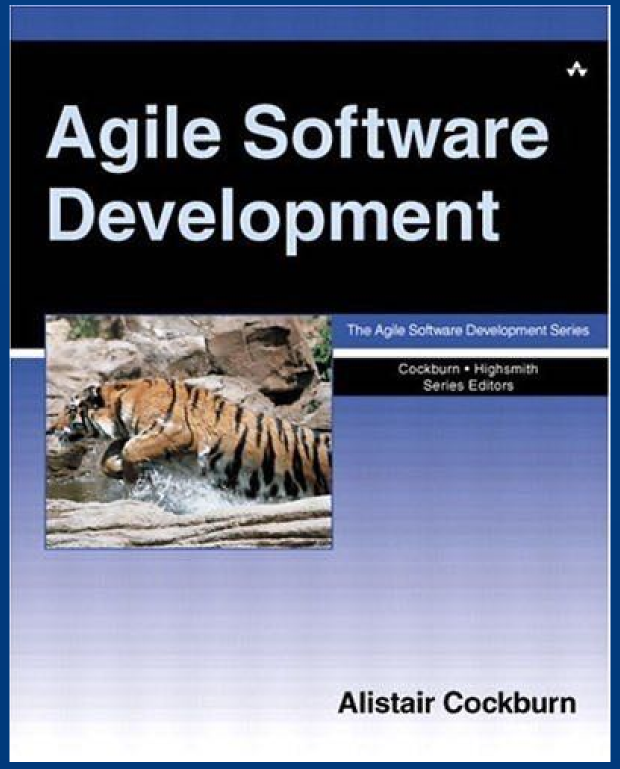
That is, while there is value in the items on
the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

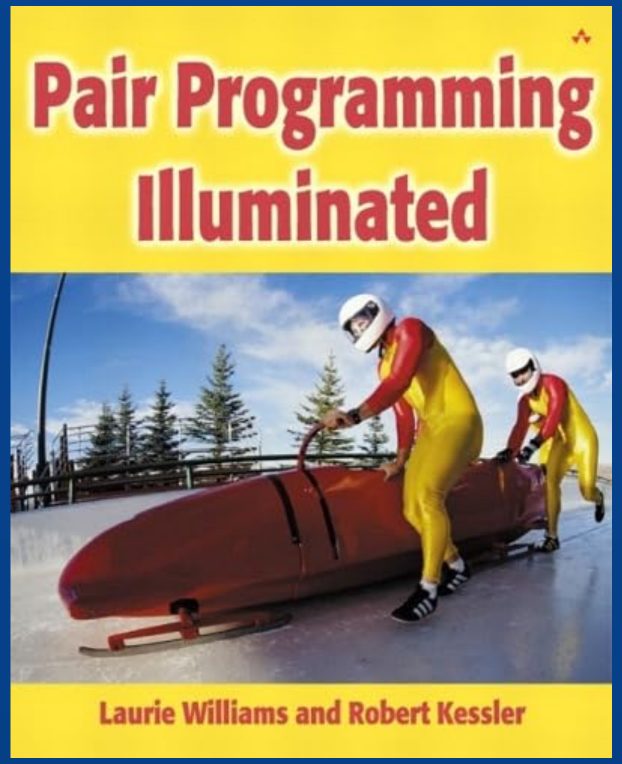
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

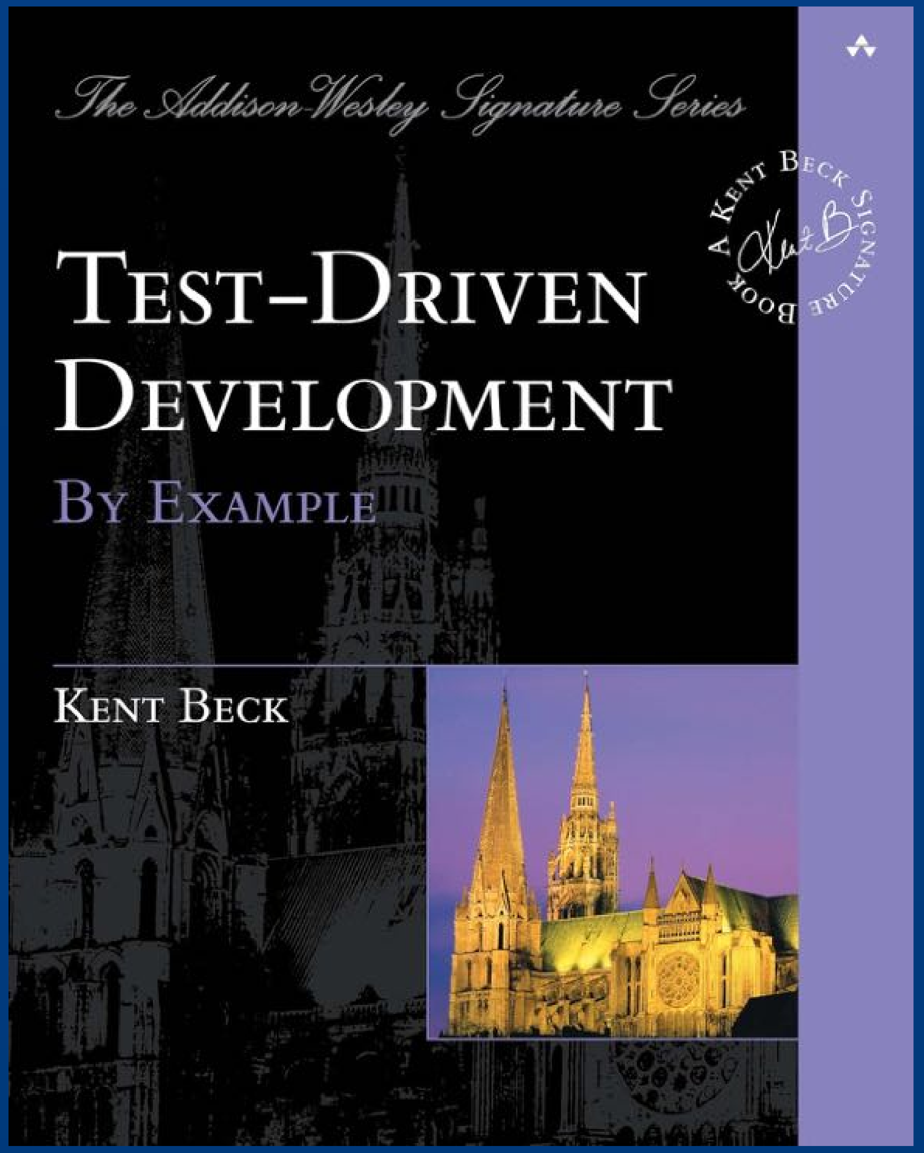
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Twelve Principles of Agile Software


Agile Software Development
The Agile Software Development Series
Cockburn • Highsmith
Series Editors
Alistair Cockburn

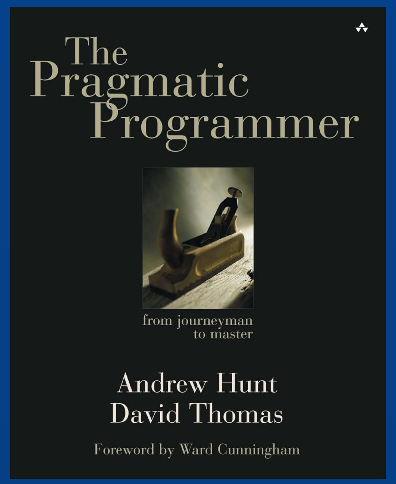
2002


Pair Programming Illuminated
Laurie Williams and Robert Kessler

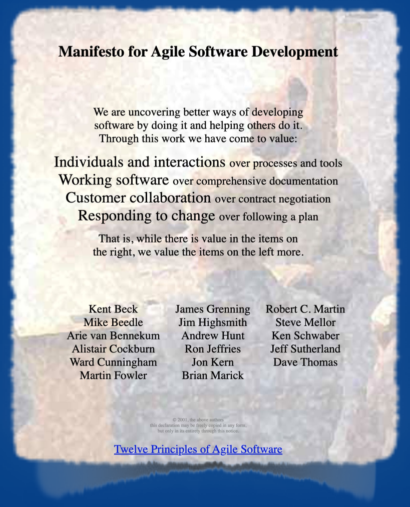

The Addison Wesley Signature Series
A Kent Beck Signature Book
TEST-DRIVEN DEVELOPMENT
BY EXAMPLE
KENT BECK

History ~~Repeating~~ Evolving

1999



2001



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

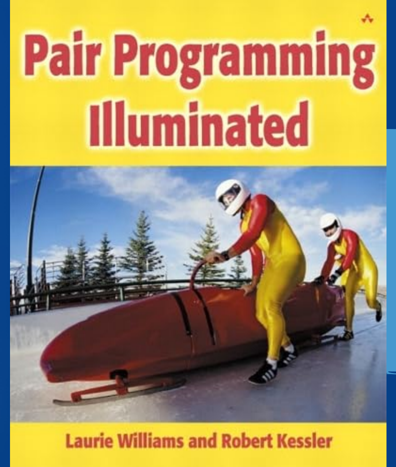
That is, while there is value in the items on
the right, we value the items on the left more.

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

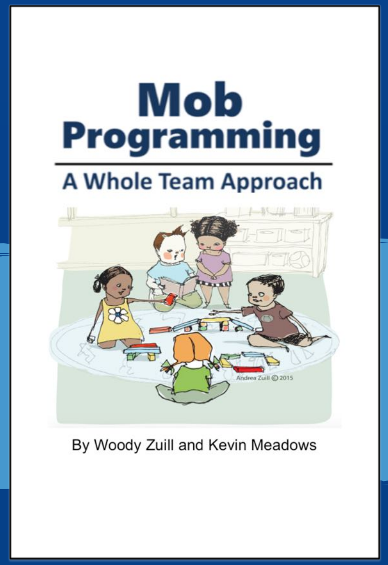
© 2001, the above authors
this declaration may be freely copied in any form,
but only in its entirety through this notice.

Twelve Principles of Agile Software

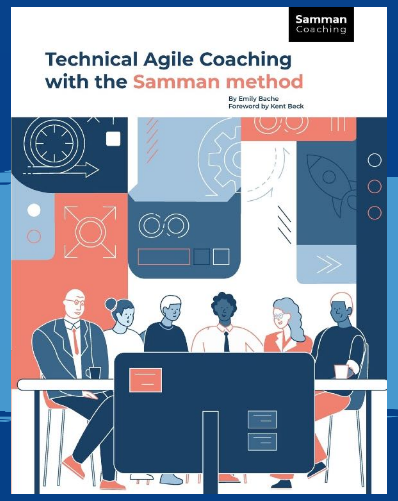
2002


Pair Programming Illuminated
Laurie Williams and Robert Kessler

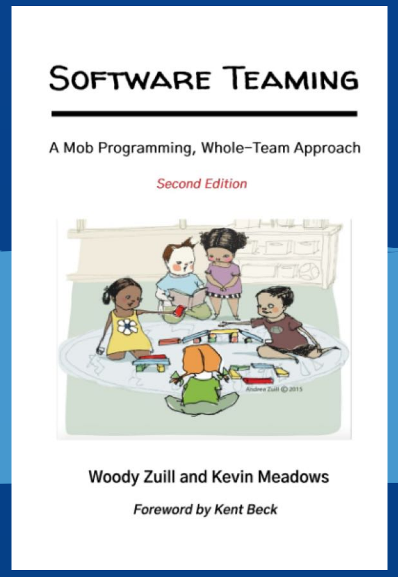
2013



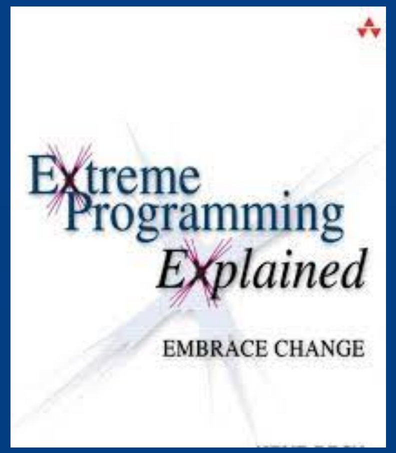
2019



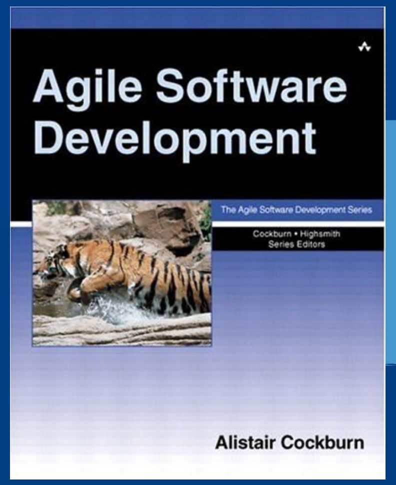
2022


Software Teaming
A Mob Programming, Whole-Team Approach
Second Edition
Woody Zuill and Kevin Meadows
Foreword by Kent Beck

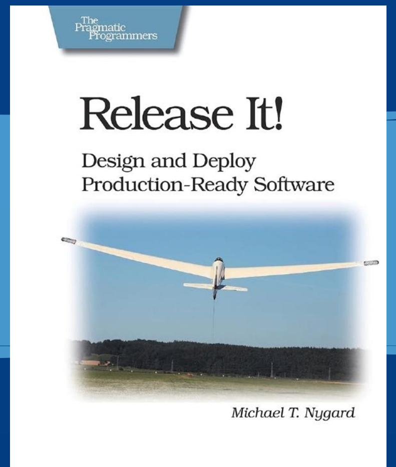
1999 (& 2004)



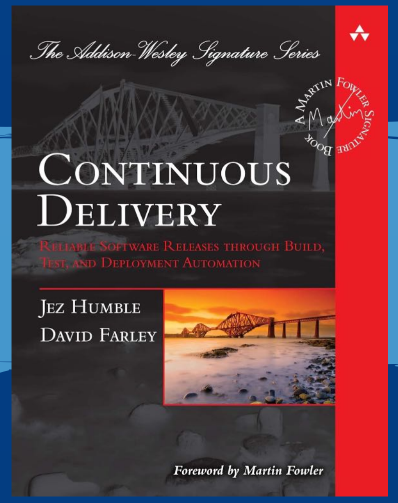
2001



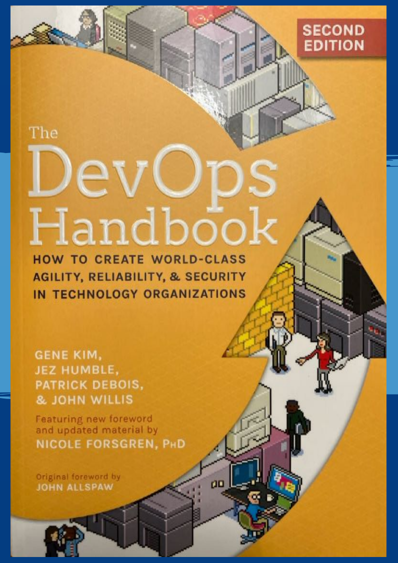
2007 (& 2018)



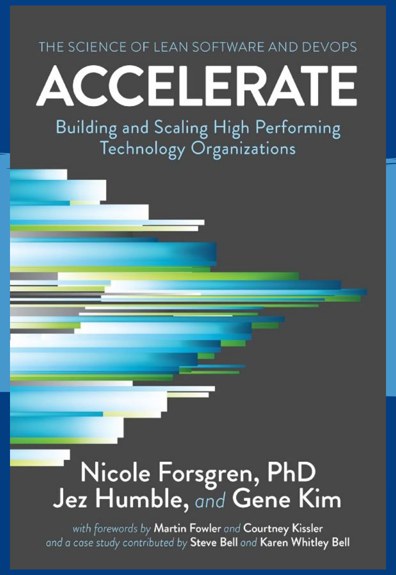
2010



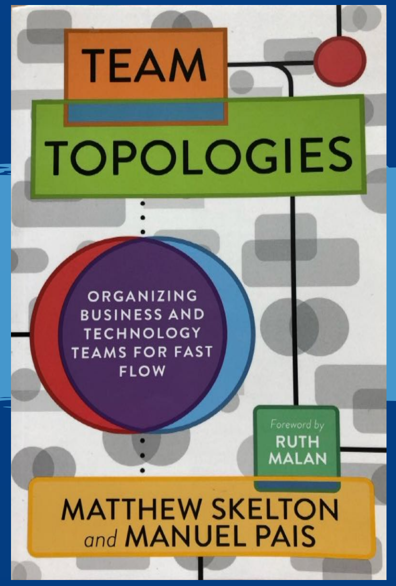
2016 (& 2021)


Second Edition
The DevOps Handbook
How to Create World-Class Agility, Reliability, & Security in Technology Organizations
Gene Kim, Jez Humble, Patrick Debois, & John Willis

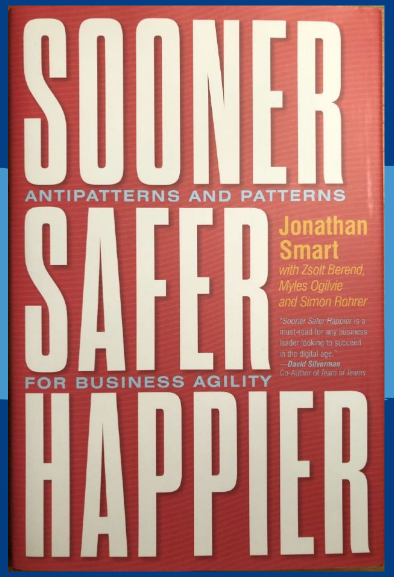
2018


Accelerate
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim

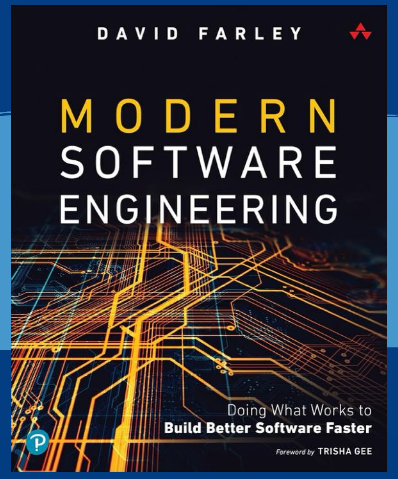
2019


Team Topologies
Organizing Business and Technology Teams for Fast Flow
Ruth Malan
Matthew Skelton and Manuel Pais

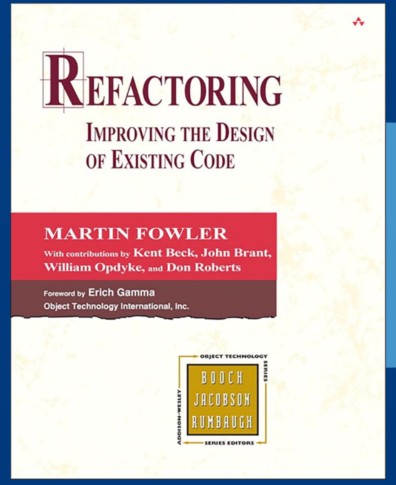
2020



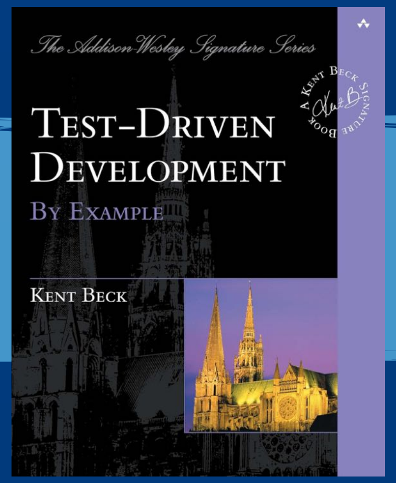
2022


David Farley
Modern Software Engineering
Doing What Works to Build Better Software Faster

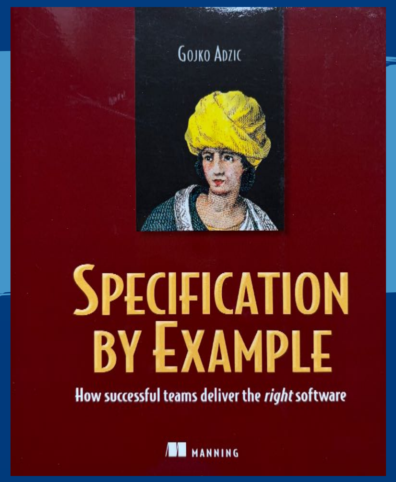
1999 (& 2019)



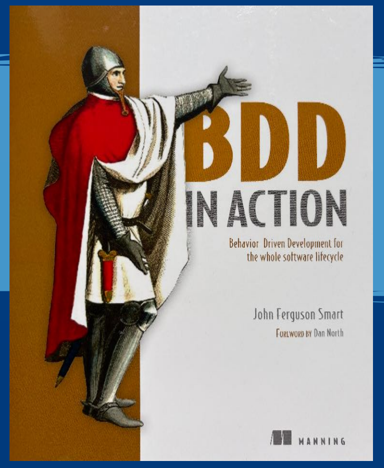
2002



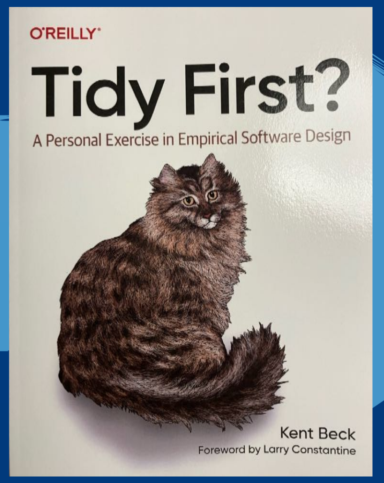
2011



2015 (& 2023)



2023



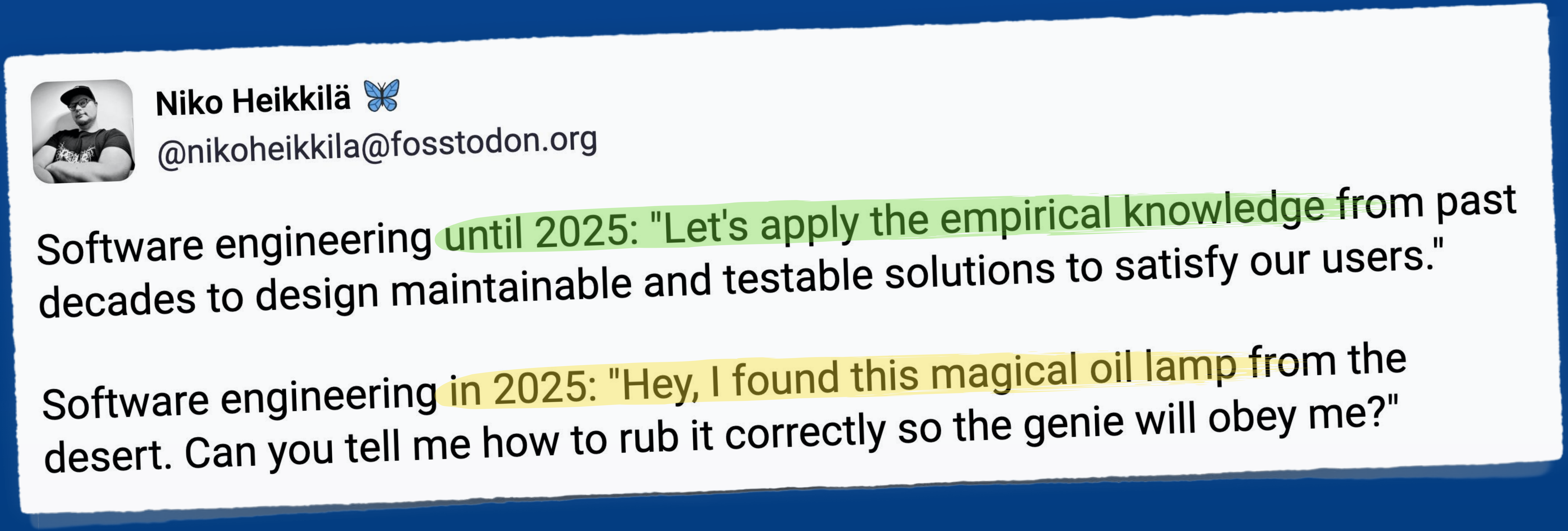
Modern Software Development 2025

Applying all the **good practices**
we've been knowing for 15, 20, 30, 50 years.

Only this time we've got **PROOF***
and decision-maker friendly documentation & guides.

*) empirical underpinning

What about **(Gen)AI?** 🤖



Niko Heikkilä 🦋
@nikoheikkila@fosstodon.org

Software engineering until 2025: "Let's apply the empirical knowledge from past decades to design maintainable and testable solutions to satisfy our users."

Software engineering in 2025: "Hey, I found this magical oil lamp from the desert. Can you tell me how to rub it correctly so the genie will obey me?"

https://mastodon.social/@nikoheikkila@fosstodon.org/114765667066055508

DORA 2025 Findings on AI

Developer Happiness ↗

Software Delivery Performance ↘

More Bugs, Quality ↘

Can we still **review the code** being generated?

How **fast** can we **understand?**

Typing code fast was never the bottleneck.

Thinking is.

Skills & capabilities are.

Required Future Skill? 🔮

Writing

testable

executable

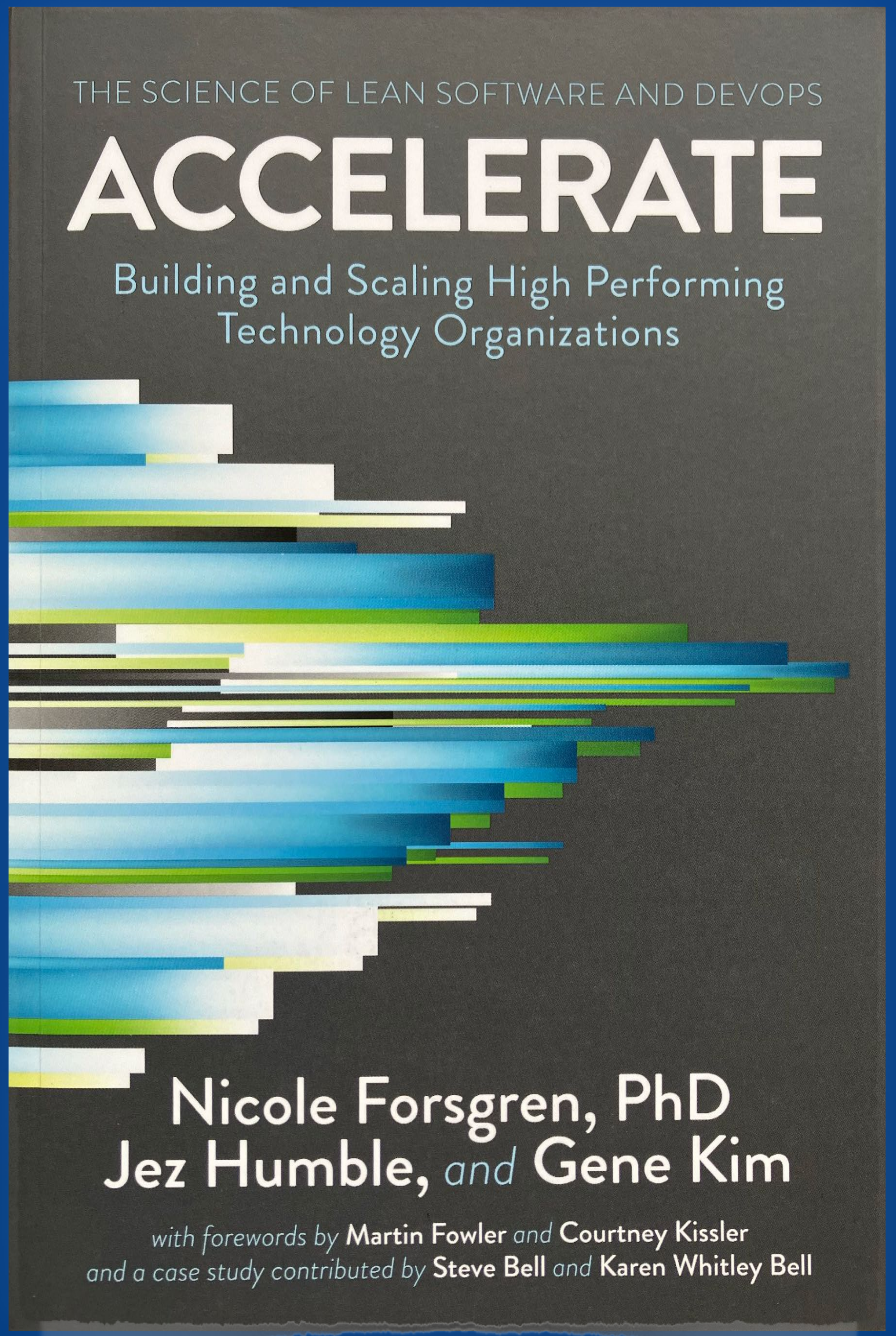
specifications

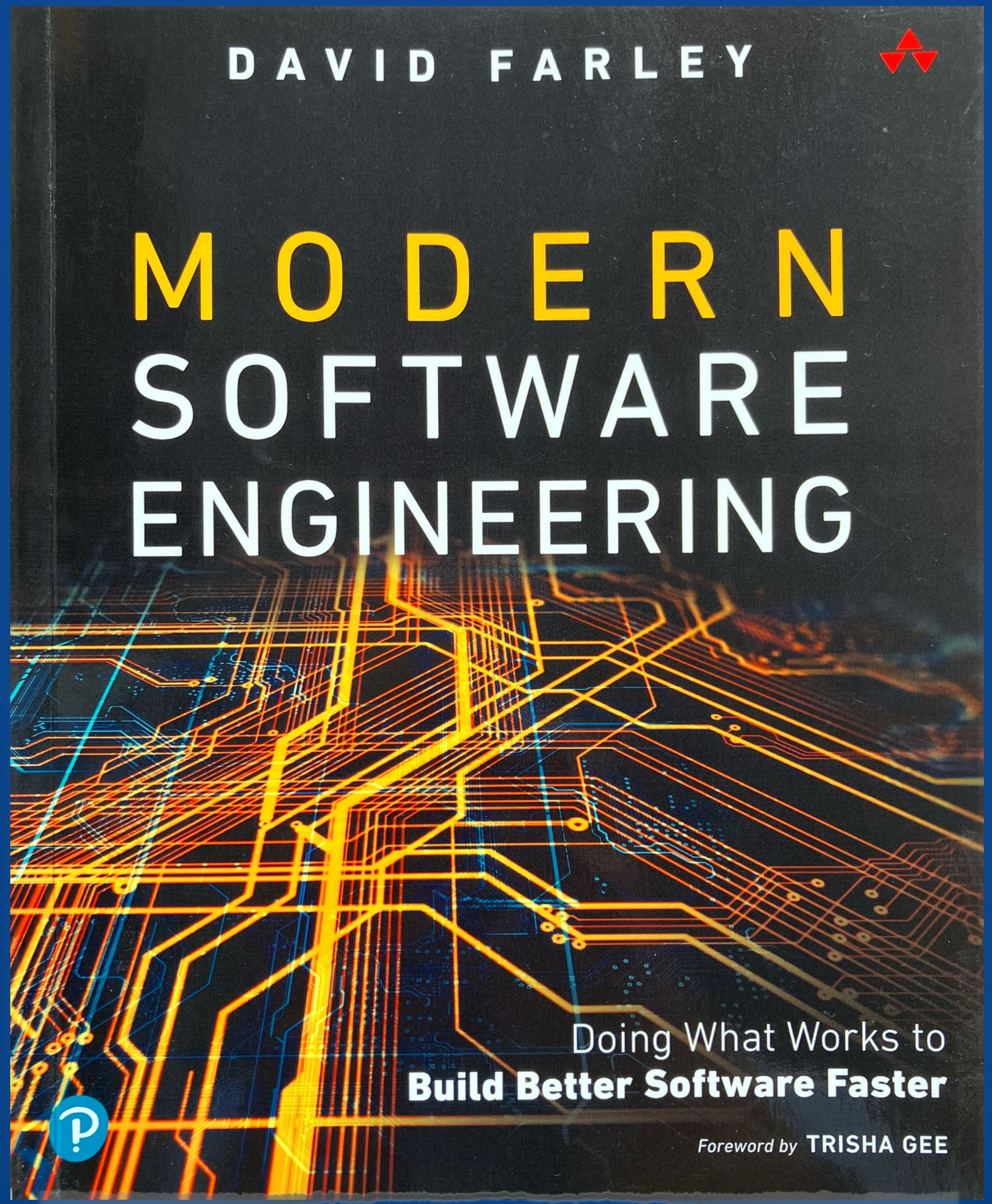
Recap

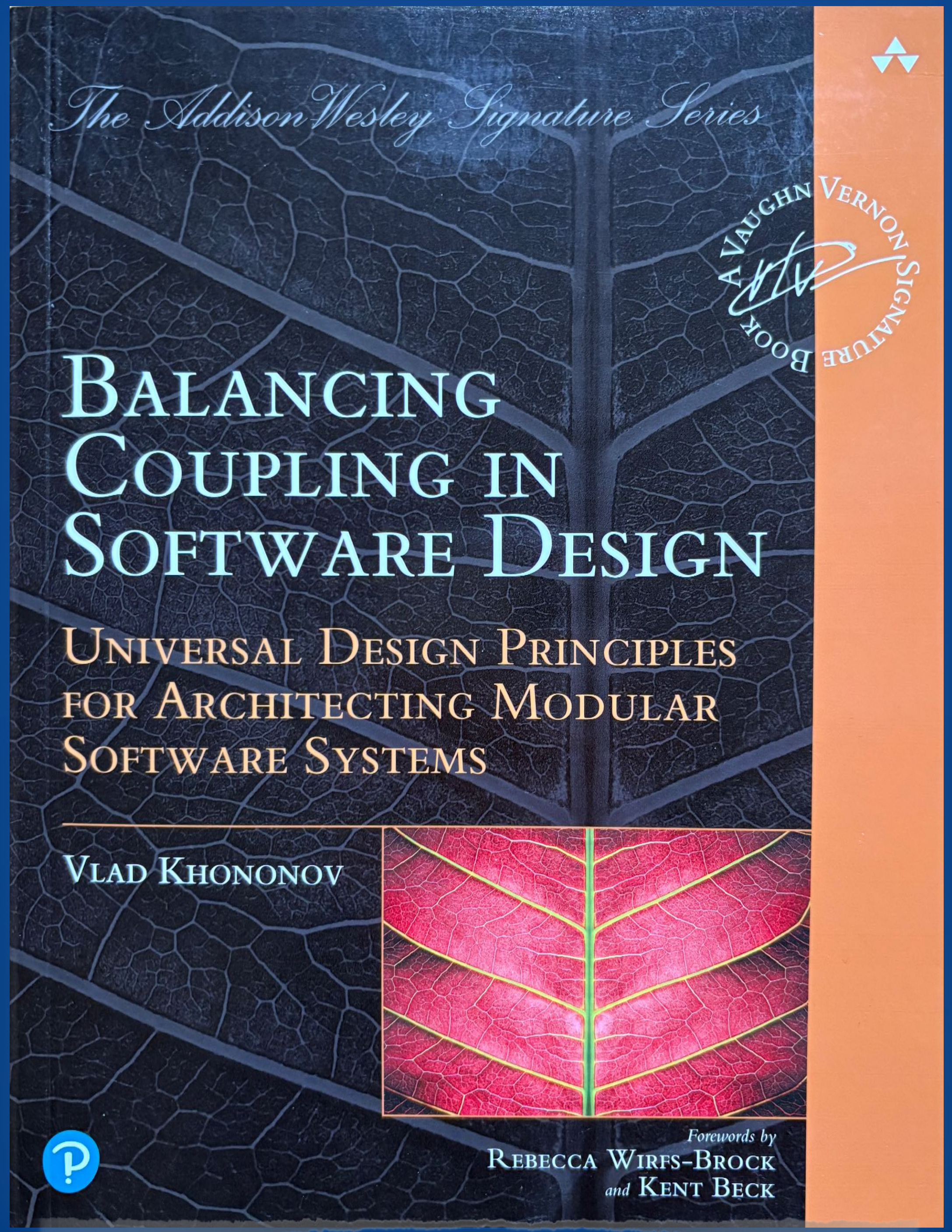
DORA capabilities super important!
resp. **underlying software development practices & skills**

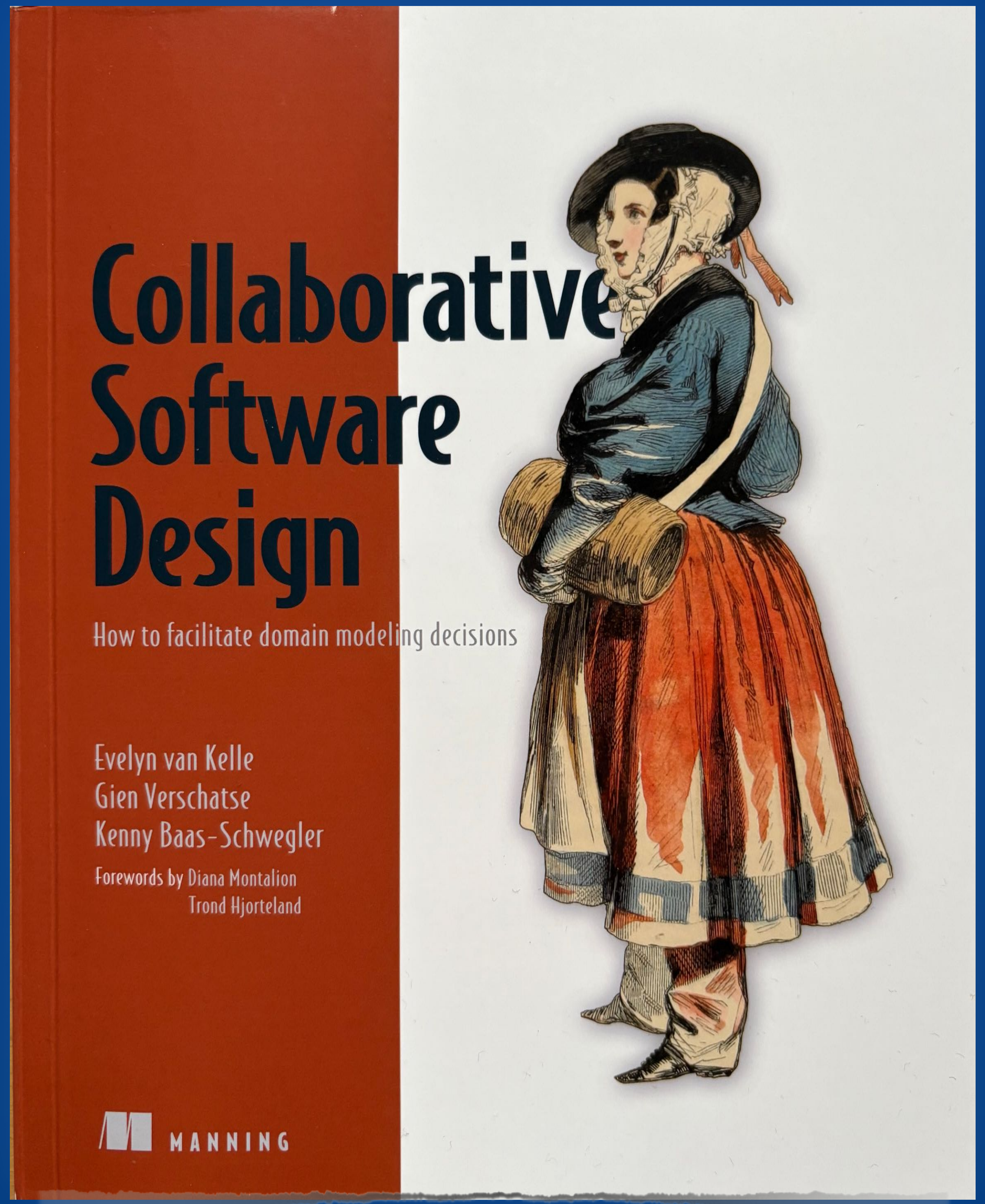
Most probably **even in the age of (Gen)AI**

Get to know them. Learn them. Use 'em.


THE SCIENCE OF LEAN SOFTWARE AND DEVOPS
ACCELERATE
Building and Scaling High Performing Technology Organizations
Nicole Forsgren, PhD
Jez Humble, and Gene Kim
with forewords by Martin Fowler and Courtney Kissler
and a case study contributed by Steve Bell and Karen Whitley Bell







'... the real trade-off, over long periods of time, is between "**better software faster**" and "**worse software slower**"' – Dave Farley www.infoq.com/articles/replace-process-dogma-engineering/

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

minimumcd.org

dora.dev

trunkbaseddevelopment.com

www.continuous-delivery.co.uk

youtube.com/@ModernSoftwareEngineeringYT

martinfowler.com/articles/continuousIntegration.html

scs-architecture.org

collaborative-software-design.com

coupling.dev

www.soonersaferhappier.com

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

Die Techniker

www.tk.de/IT

Thank You!

@thmuch

Generated with GPT-4o