

SCRUM MASTERS 

Wie

Testbarkeit

unterstützt

eure Agilität

Thomas Much

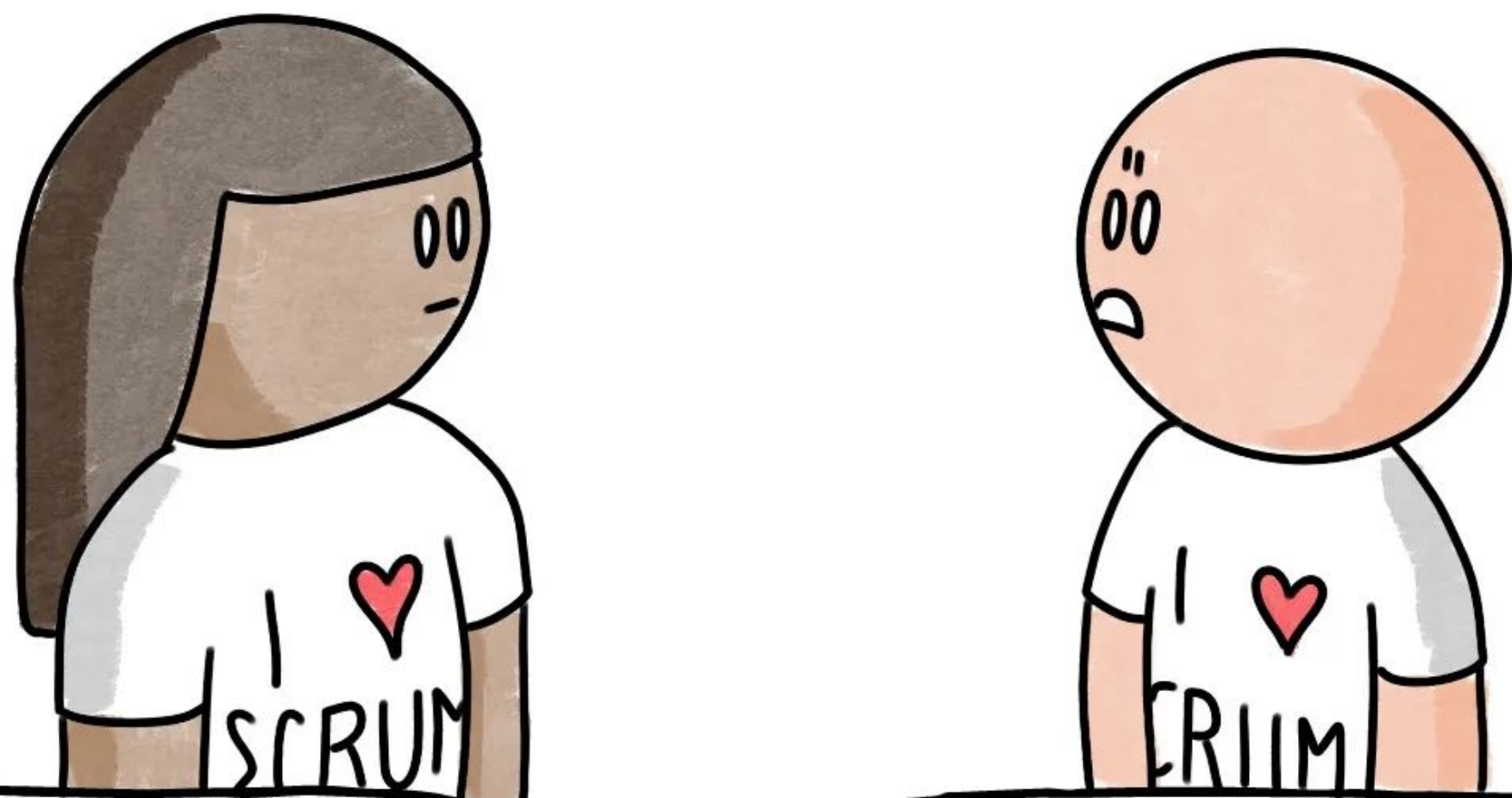
   @thmuch

23. November 2023

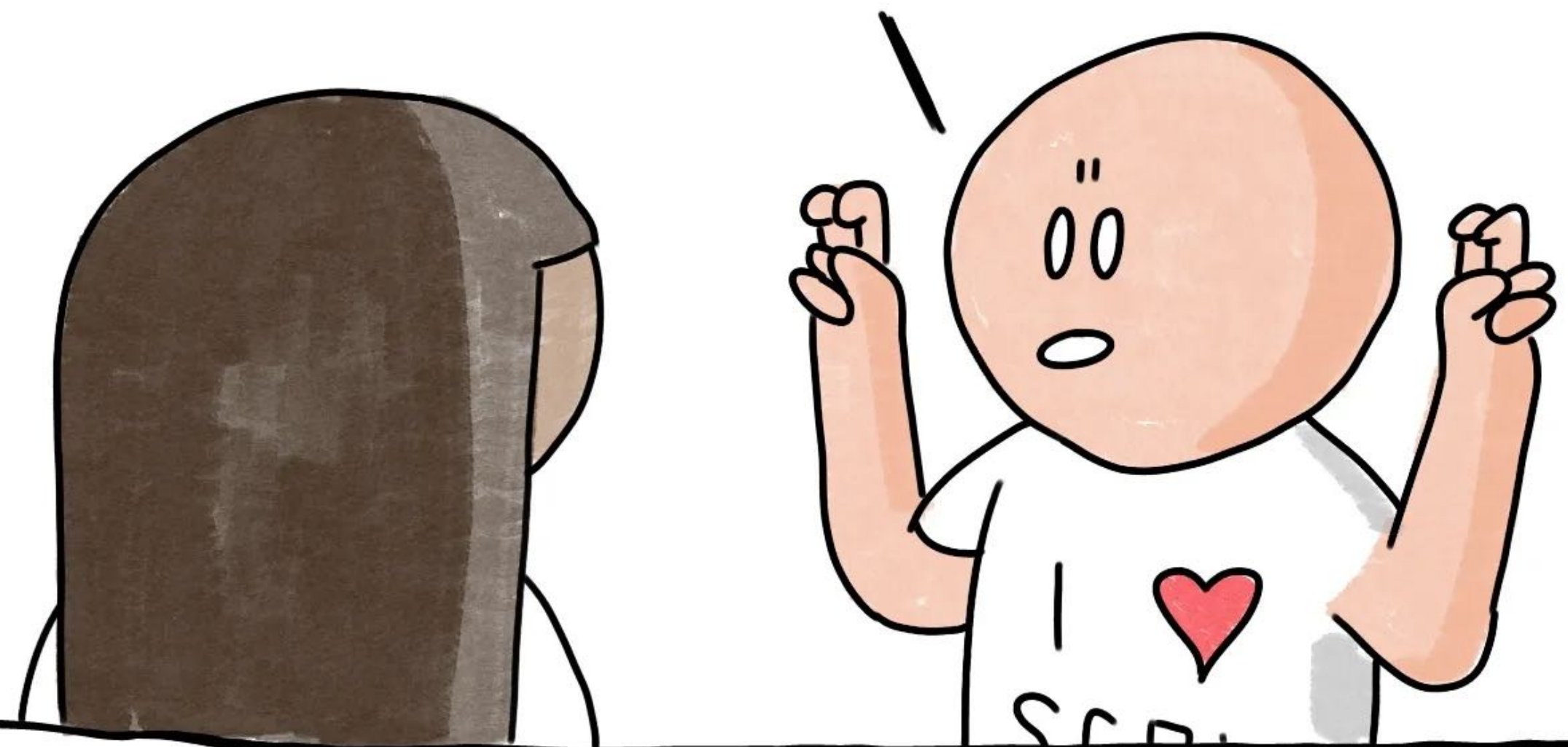
Warum dieser Talk?

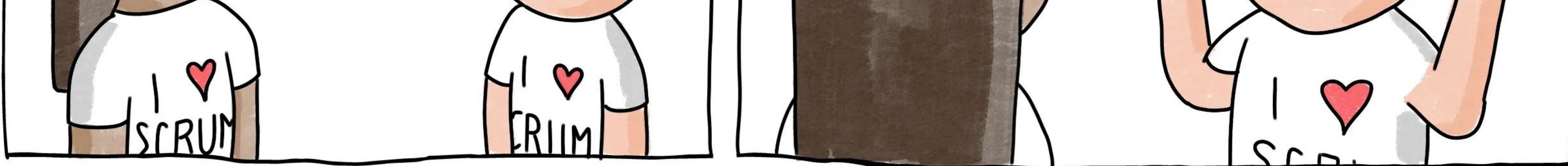
Comic Agilé

It's difficult to get through to developers because I don't have the technical understanding of software development.



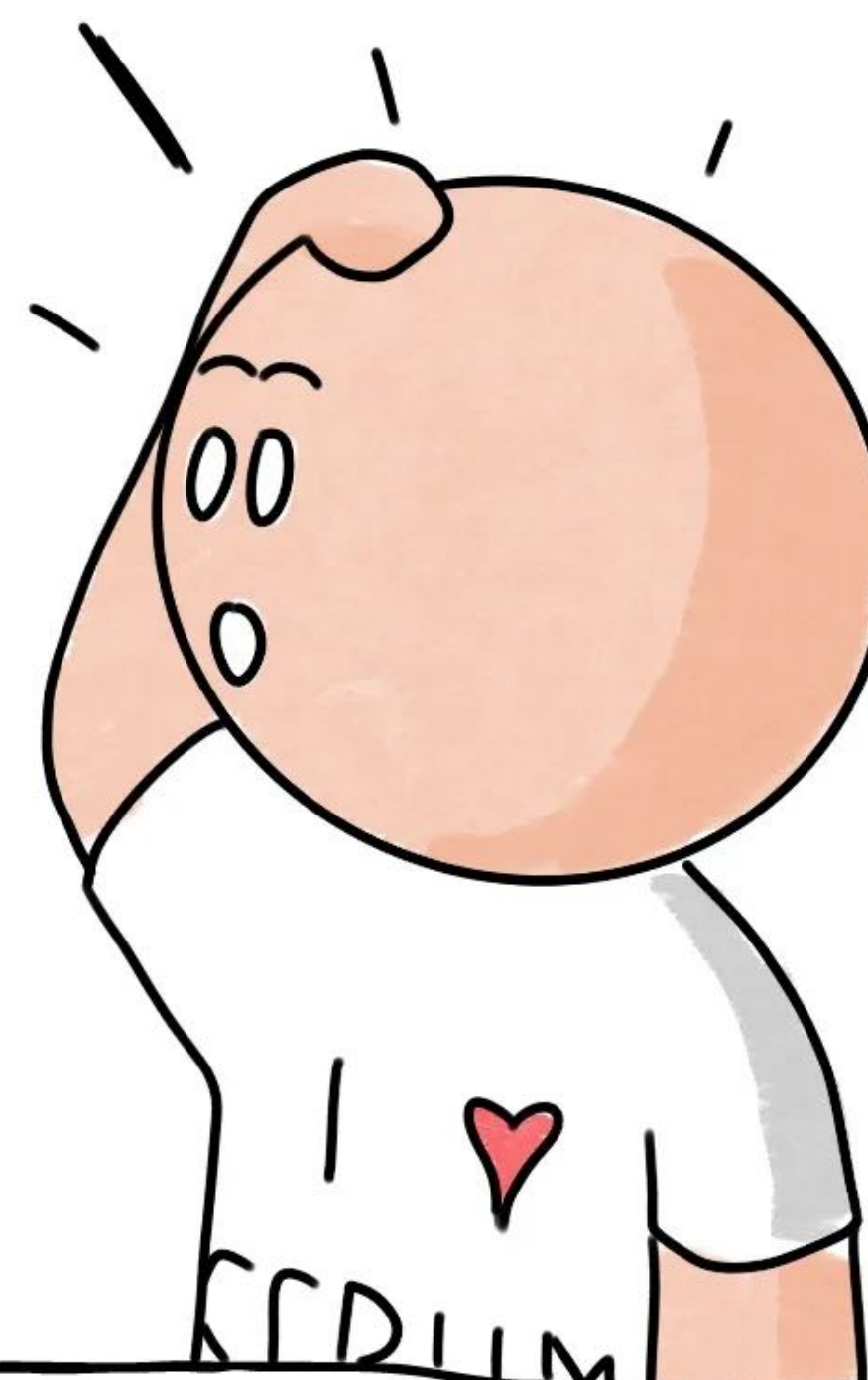
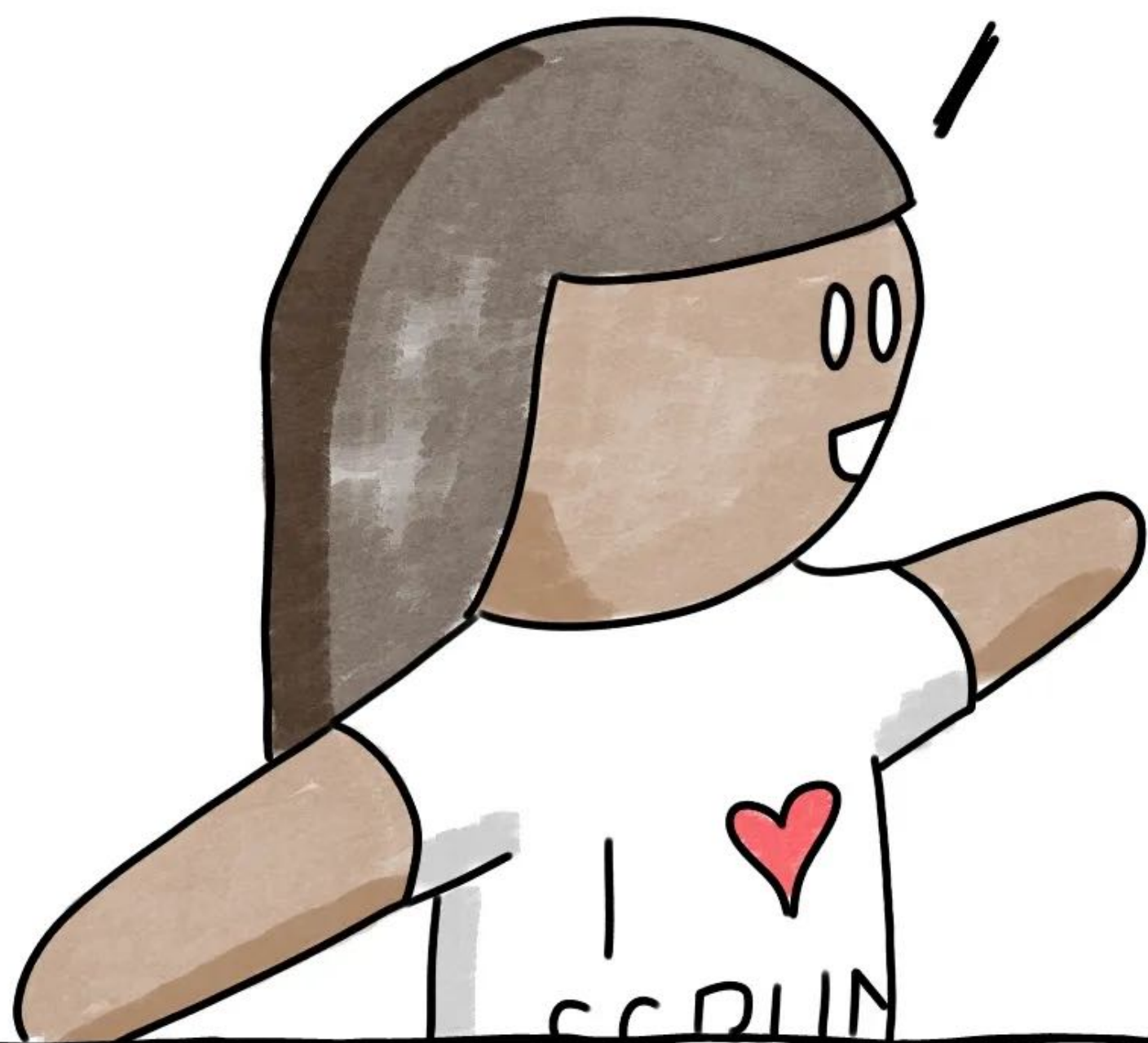
And if I have a "controversial" suggestion on improving our ways of working, they always shoot it down and say that my process change is not technically feasible.





Have you tried showing a genuine interest in how the developers build the software? You know, show that you care about their craft?

No, of course not. That's too technical for me.



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.

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

Hinweis

Die gleich gezeigten Ideen sind **nicht neu**.

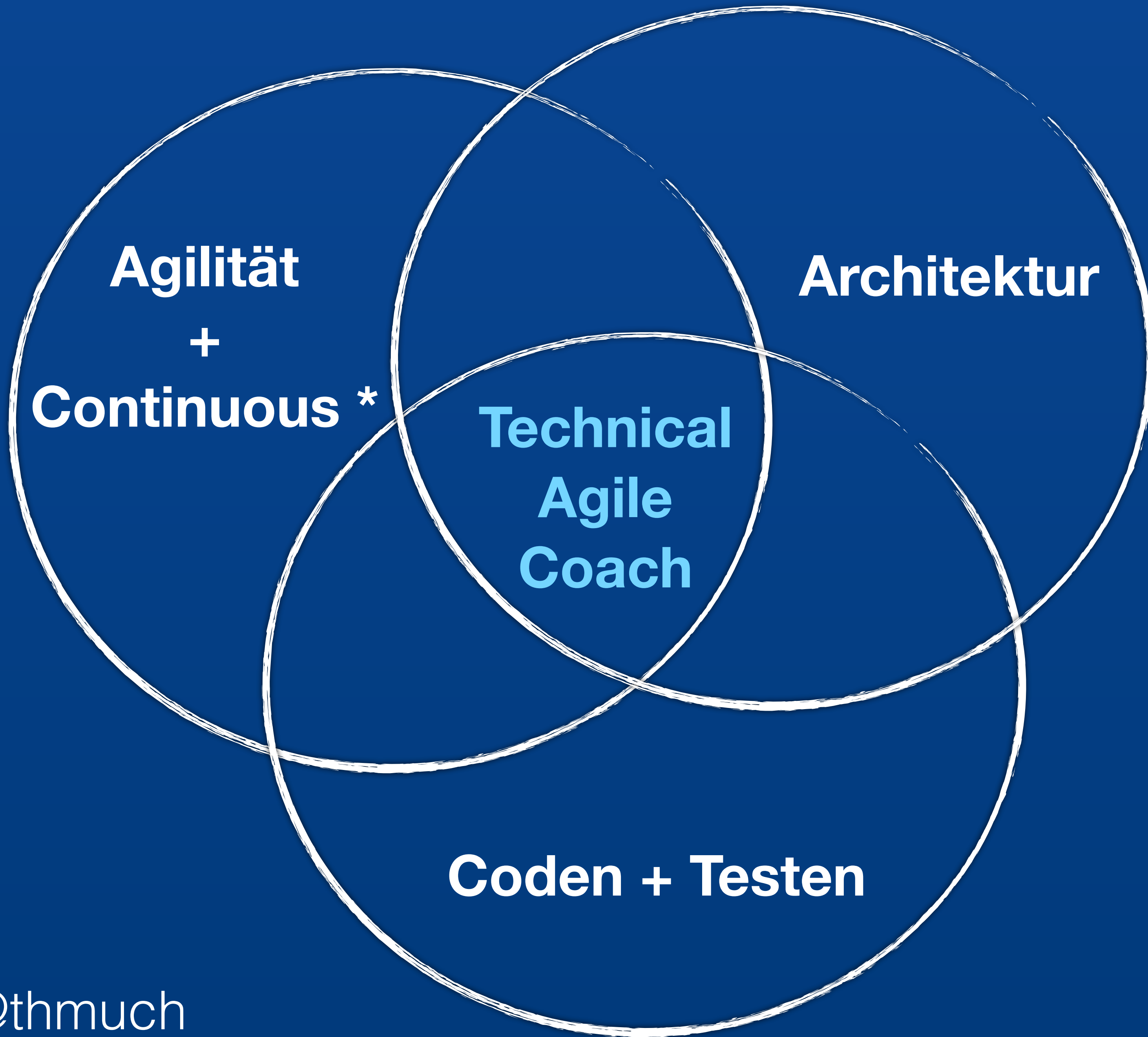
Ab und zu müssen wir über solche **Grundlagen** reden.

Diese (alten) Grundlagen sind immer noch **fundamental wichtig**.

Vielleicht sogar wichtiger als vor 20 Jahren,
weil immer mehr Unternehmen **agil** sein
und Software **kontinuierlich testen** und ausliefern wollen.



www.tk.de/IT



Coden + Testen

Typische Situationen:

Unser Team arbeitet nicht „Agile“ genug.

Kunden unzufrieden, Ergebnisse kommen zu spät.

„Wir sehen keinen Fortschritt“

Architektur

**Agilität
+
Continuous ***

Typische Symptome

Keine Tests

(nur ein bisschen Herumgeklicke hier und da)

Weitestgehend manuelle Tests

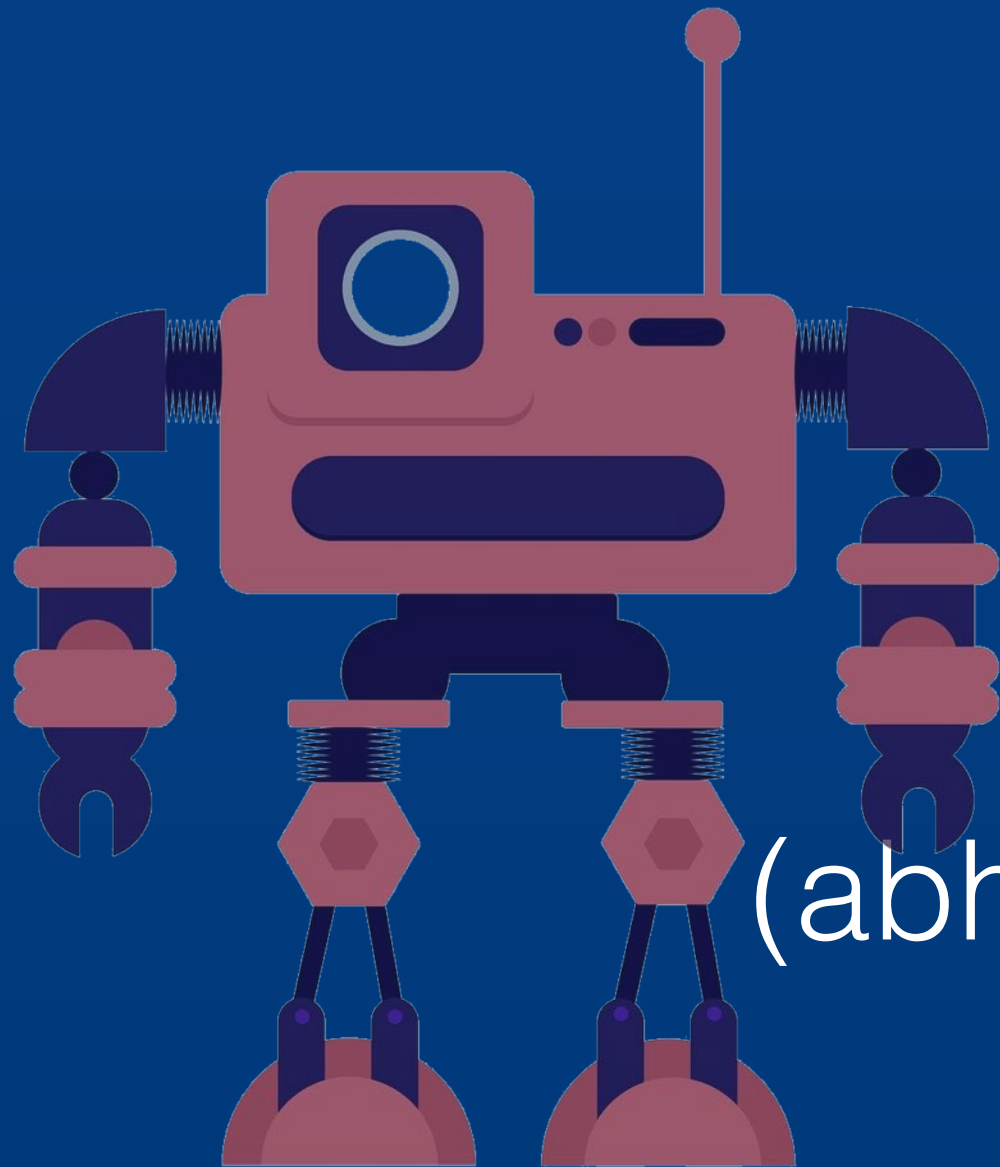
(langsam!)

Langsame automatisierte Tests

(üblicherweise End-to-end via UI)

Fragile, instabile, wackelige Tests

(abhängig von der Umgebung bzw. externen Systemen)



Software testen – warum?

Funktionieren neuer **und alter** Features sicherstellen

Änderungen 🤨

neue Features

Anpassungen bestehender Features

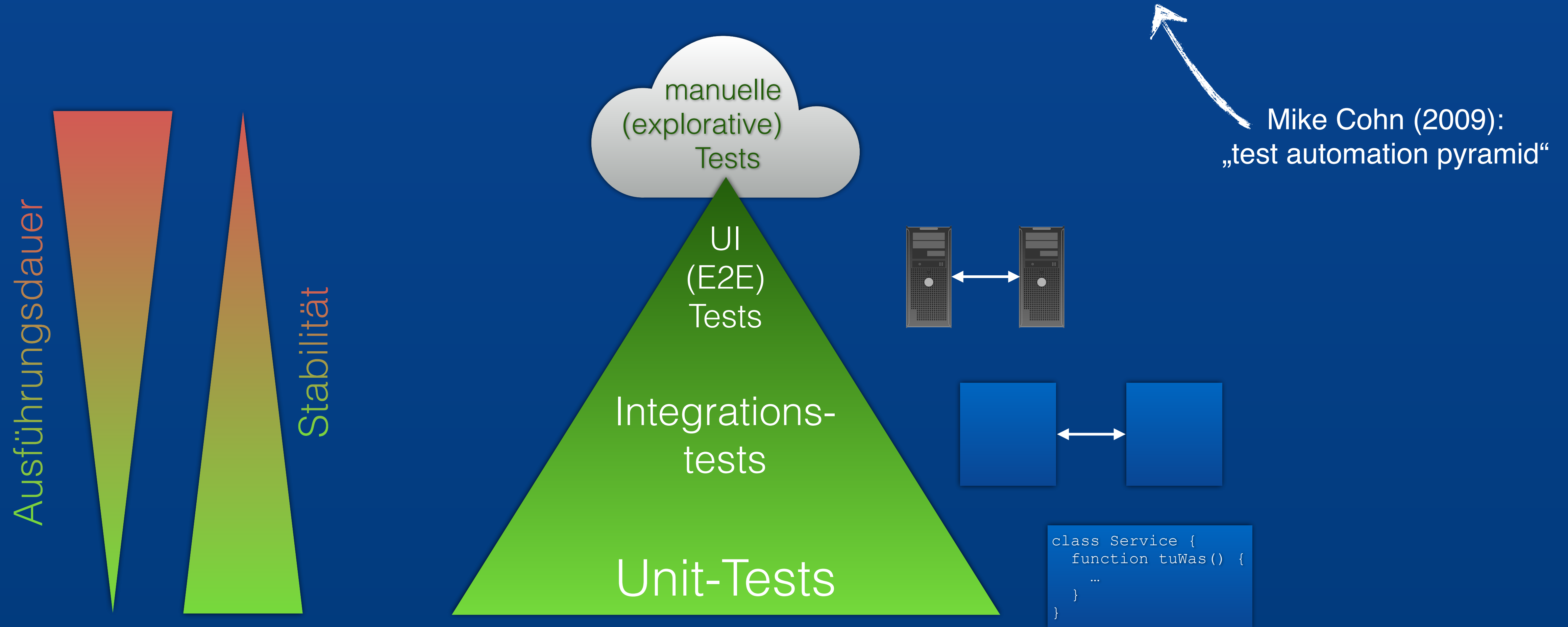
Bugfixes

Deshalb
bauen wir

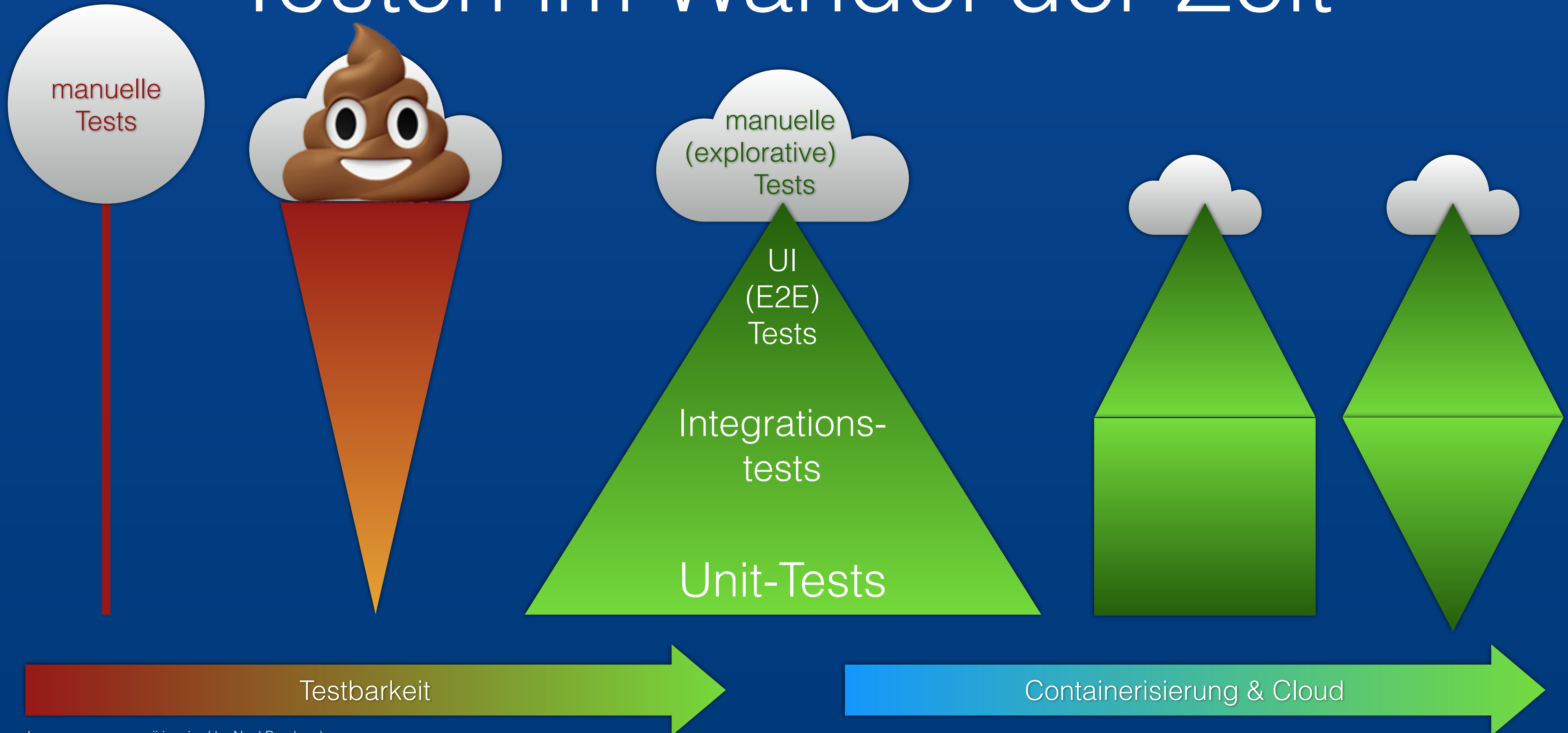
Software! 😊

Risiko managen

Klassische „agile“ Testpyramide



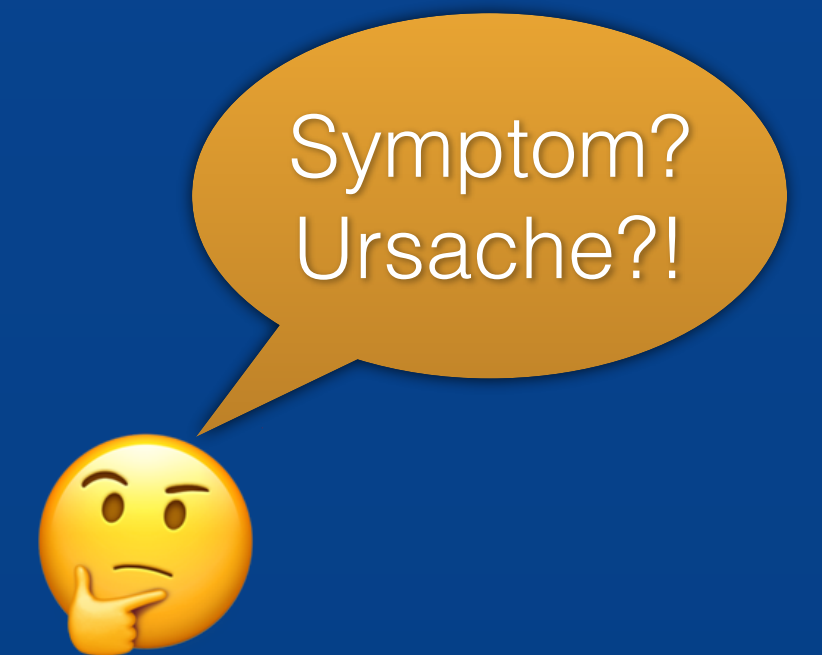
Testen im Wandel der Zeit



Meine Beobachtung

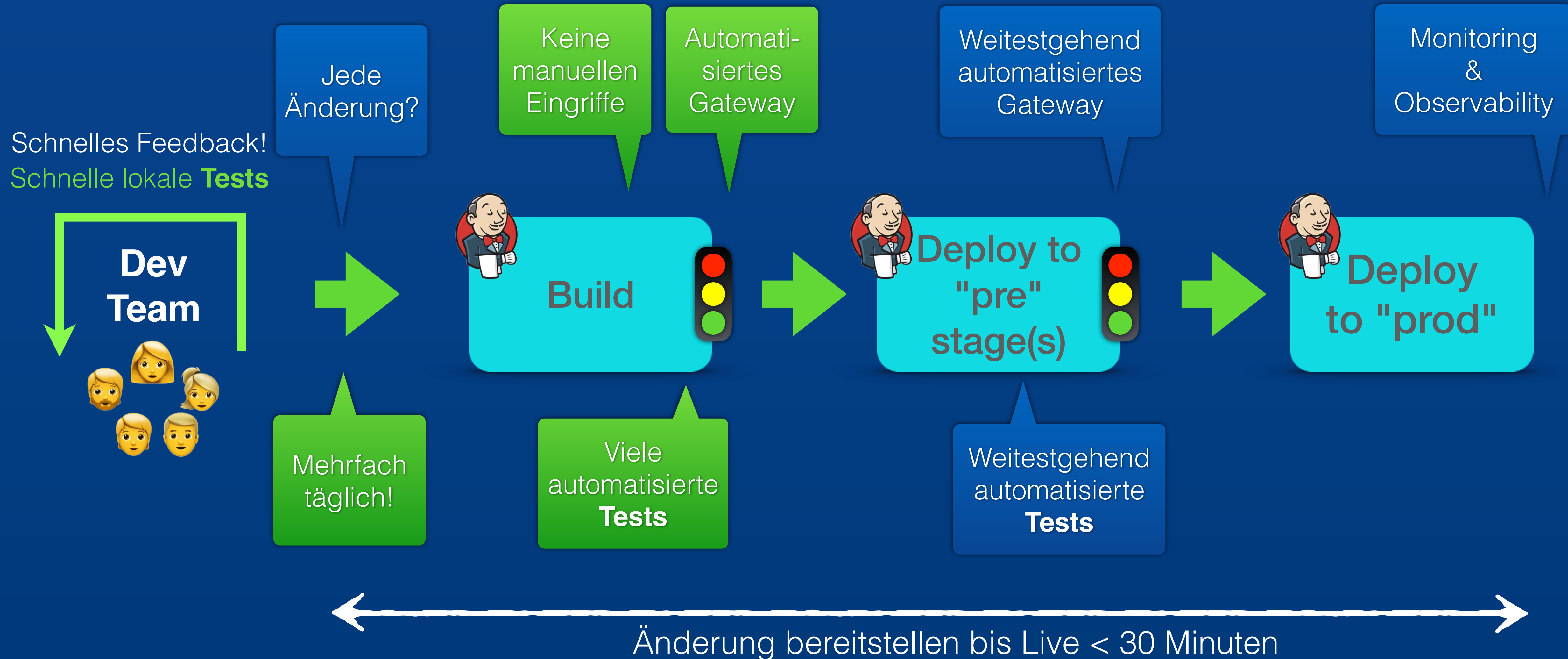
Teams mit **schneller Test-Automatisierung**
sind **agiler***

als Teams ohne oder mit nur langsamen Tests



*) liefern häufiger aus, liefern kleinere Einheiten, haben ein gemeinsames Verständnis ihrer Software, können sich gegenseitig besser unterstützen – und sie können einfacher Tests schreiben

Was macht solche Teams agiler?



Wie können die das wagen?



Schnelle, verlässliche, umfassende Tests als Sicherheitsnetz



Sie **liefern kontinuierlich Wertvolles aus**
(und falls mal was Schlimmes™ passiert, war es nur
eine kleine Änderung, einfach zurückzudrehen)

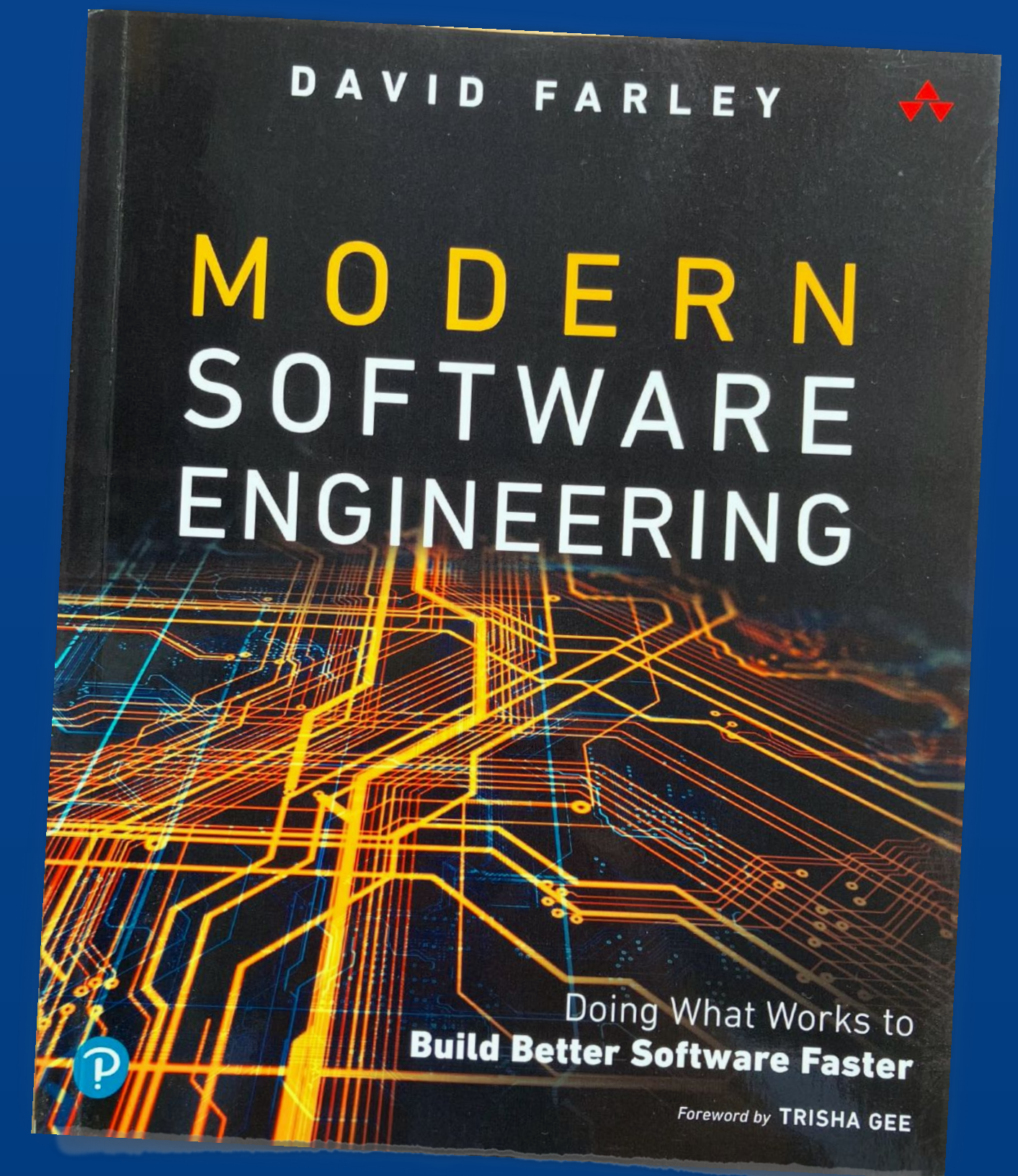
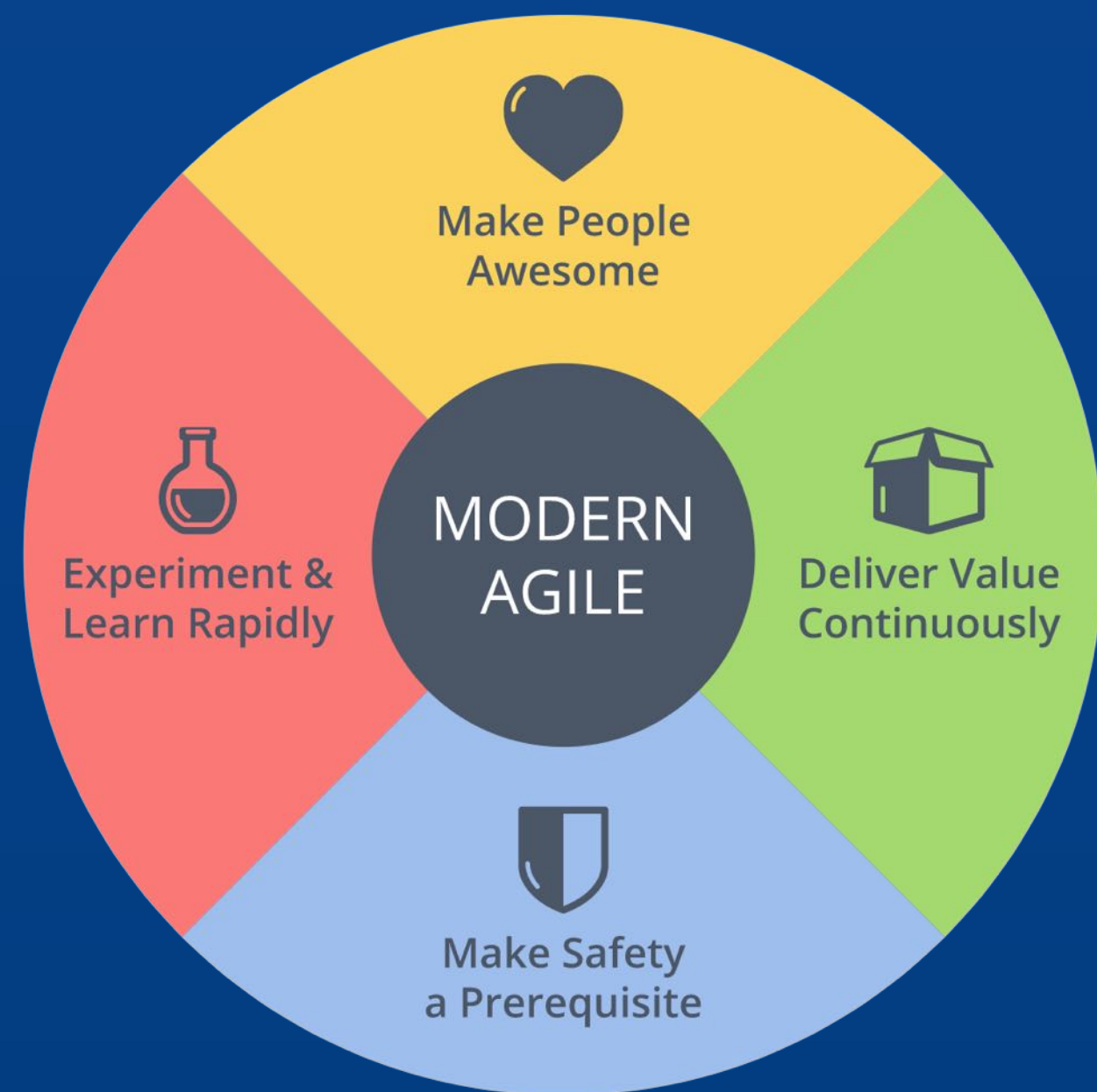


Glückliche Entwickler
(und vermutlich auch glückliche **Kunden**)



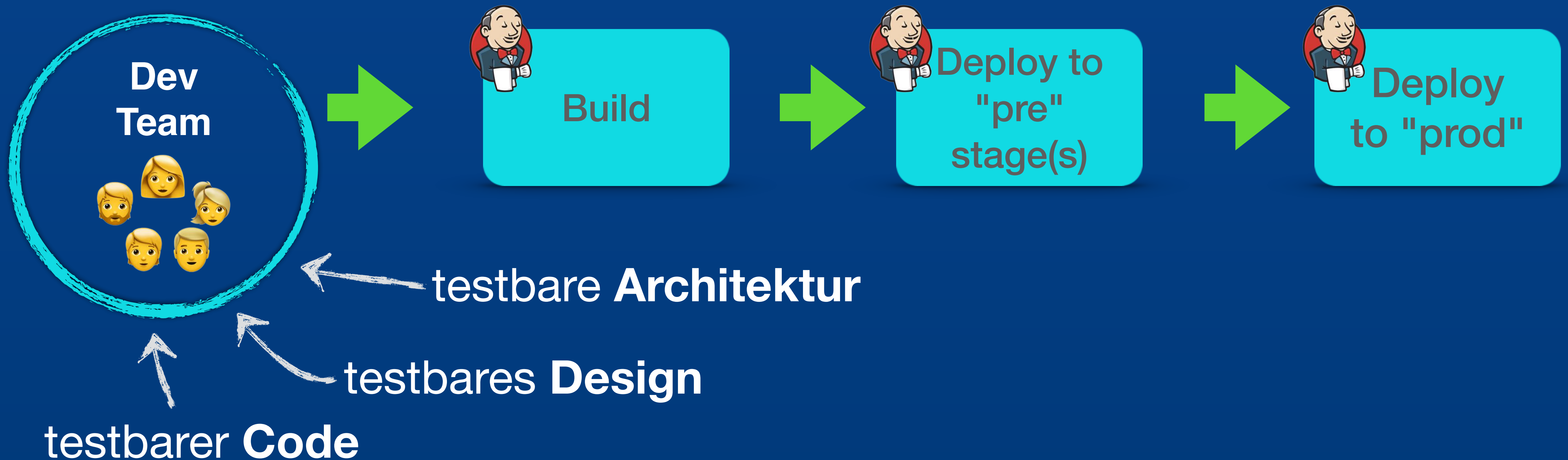
Sie trauen sich, Änderungen vorzunehmen,
trauen sich auszuprobieren, was den Kunden hilft

Das
ist
schön
kontinuierlich
und
ziemlich
agil!



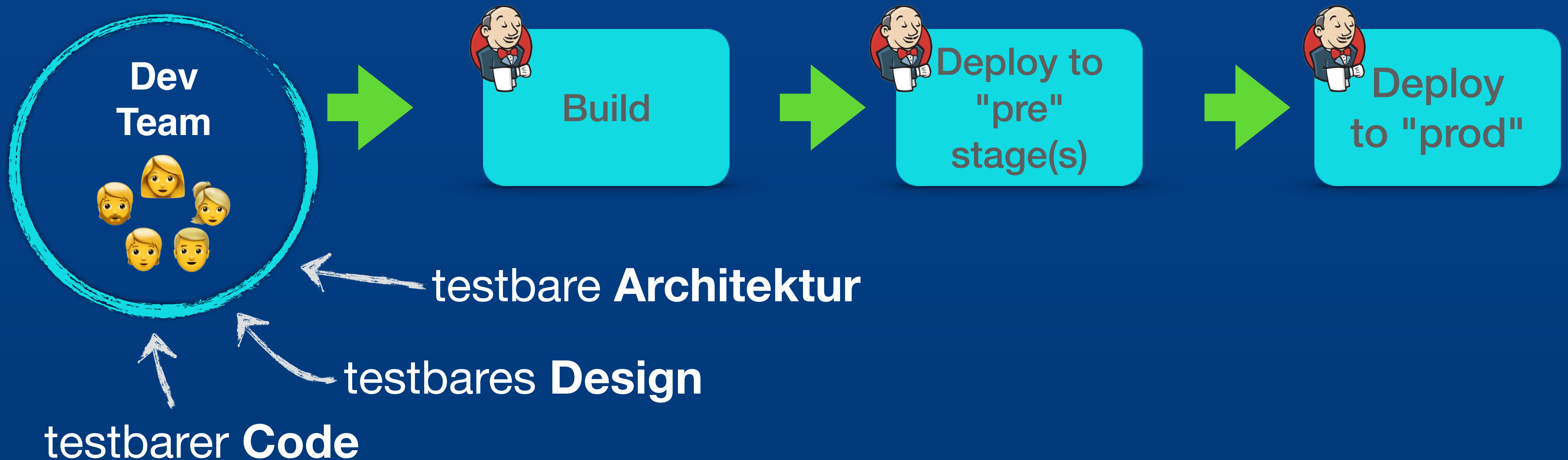
Kontinuierlich testen: Voraussetzungen?

Dev+Ops+UX+QS+...



Warum geht das **alle Rollen** an?

- Testbarkeit & kontinuierliches Testen** • ist nicht „magisch“ da
- muss **begleitet** und **moderiert** werden
 - dabei hilft **Verständnis**
- Dev+Ops+UX+QS+...





Wenn **Testbarkeit**
vernachlässigt wird

Testbarkeit

Automatisierung **praktikabel**

Testdaten **unter Kontrolle**

Schnelles Feedback

Schnelle Tests

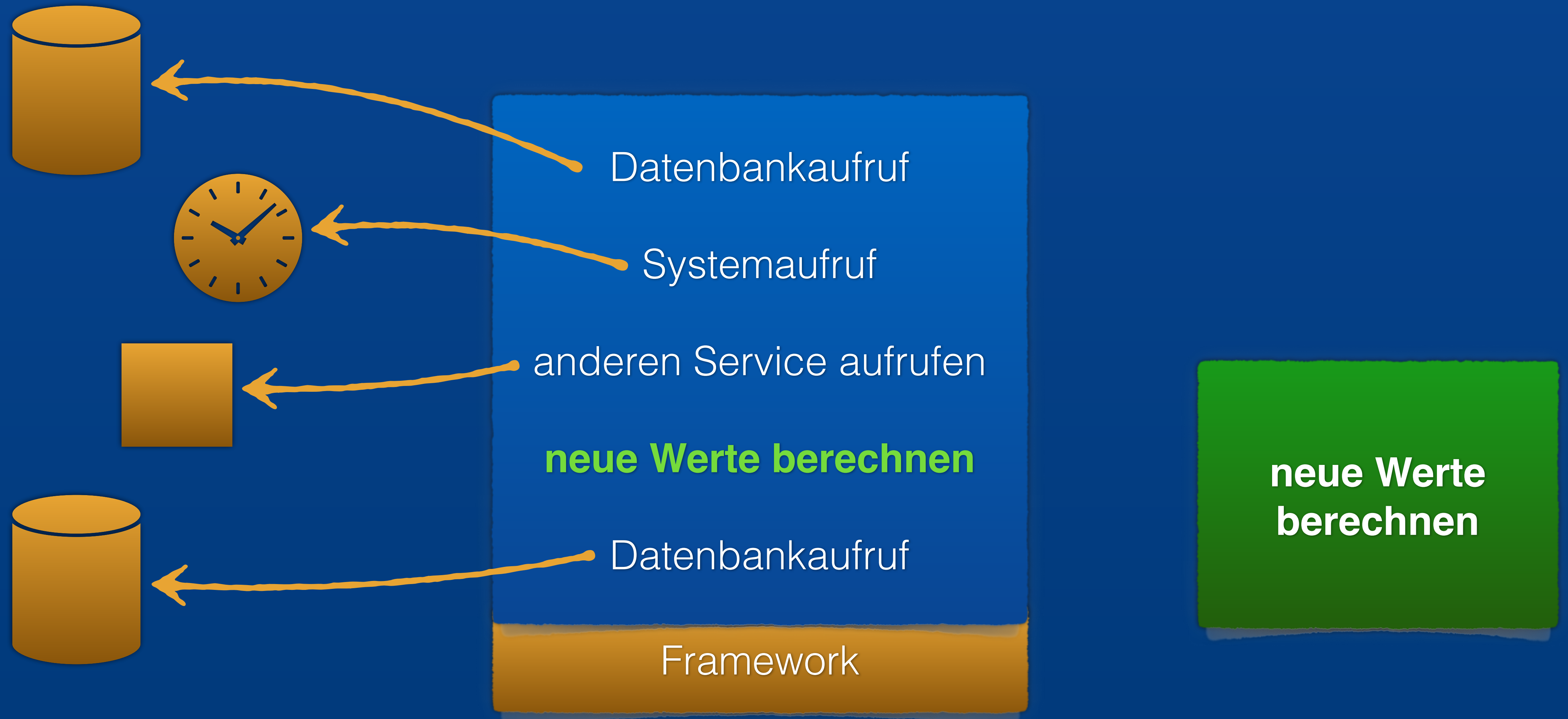
Viele
umfassende
schnelle
stabile
Tests

Schnelle Tests

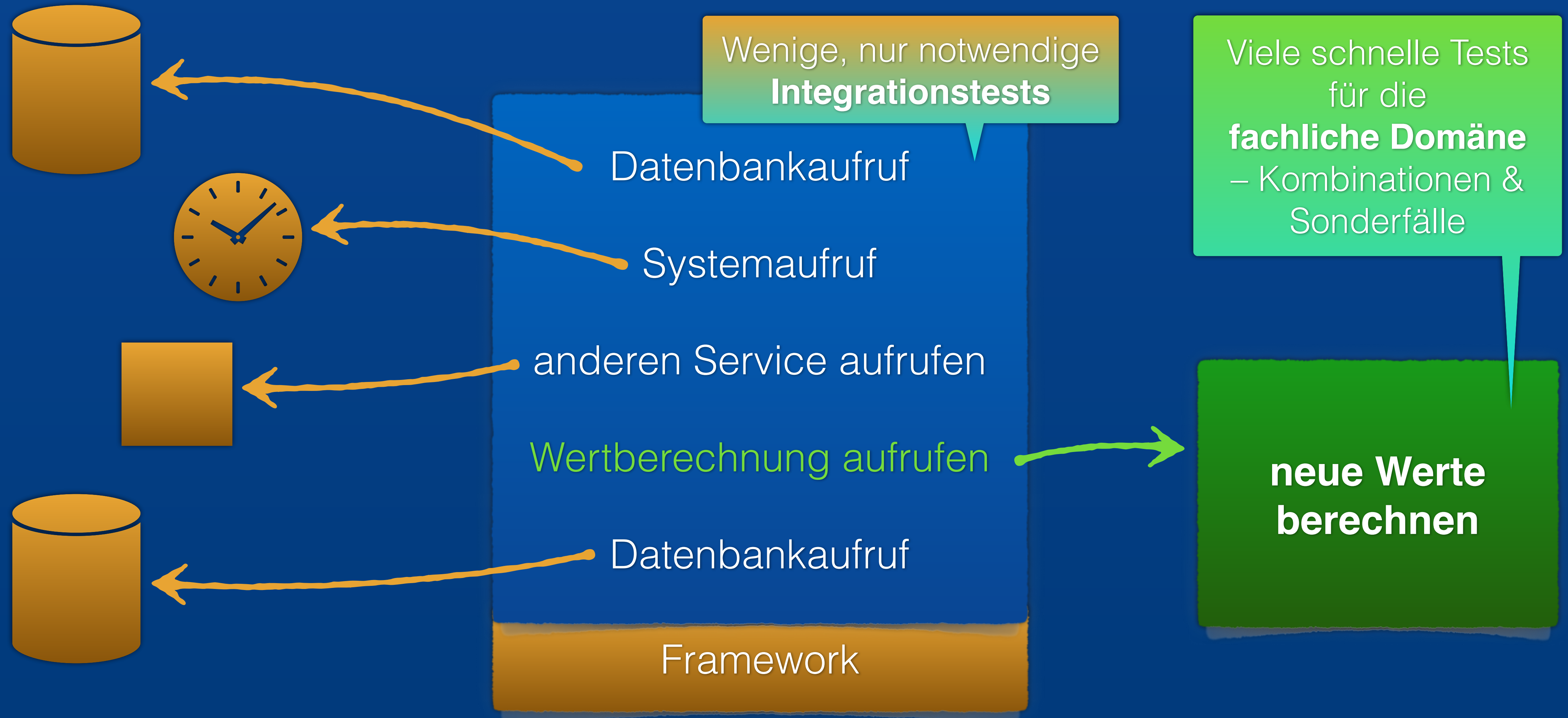
Code und **Architektur** müssen
für **Testbarkeit** designed sein

Abhängigkeiten
machen das schwierig

Abhängigkeiten



Abhängigkeiten



Abhängigkeiten erkennen

Tester:innen (Qualitätsspezialisten) können im Dev-Team **helfen**, schwierig zu testende **Abhängigkeiten zu erkennen**

Können Diskussionen der Entwickler:innen **moderieren** und mit **geeigneten Fragen** in Richtung Testbarkeit lenken

Dafür ist **Verständnis** hilfreich, wo überall Abhängigkeiten lauern und welche **Muster (Patterns)** es als mögliche Lösungen gibt

Scrum Master & Coaches können hier unterstützen!



Trennung von **Integrationscode** und **Domänencode**

... für wenige bzw. **gut beherrschbare (testbare!) Abhängigkeiten**



Code-Design

Ein
mögliches
Muster
(Pattern)



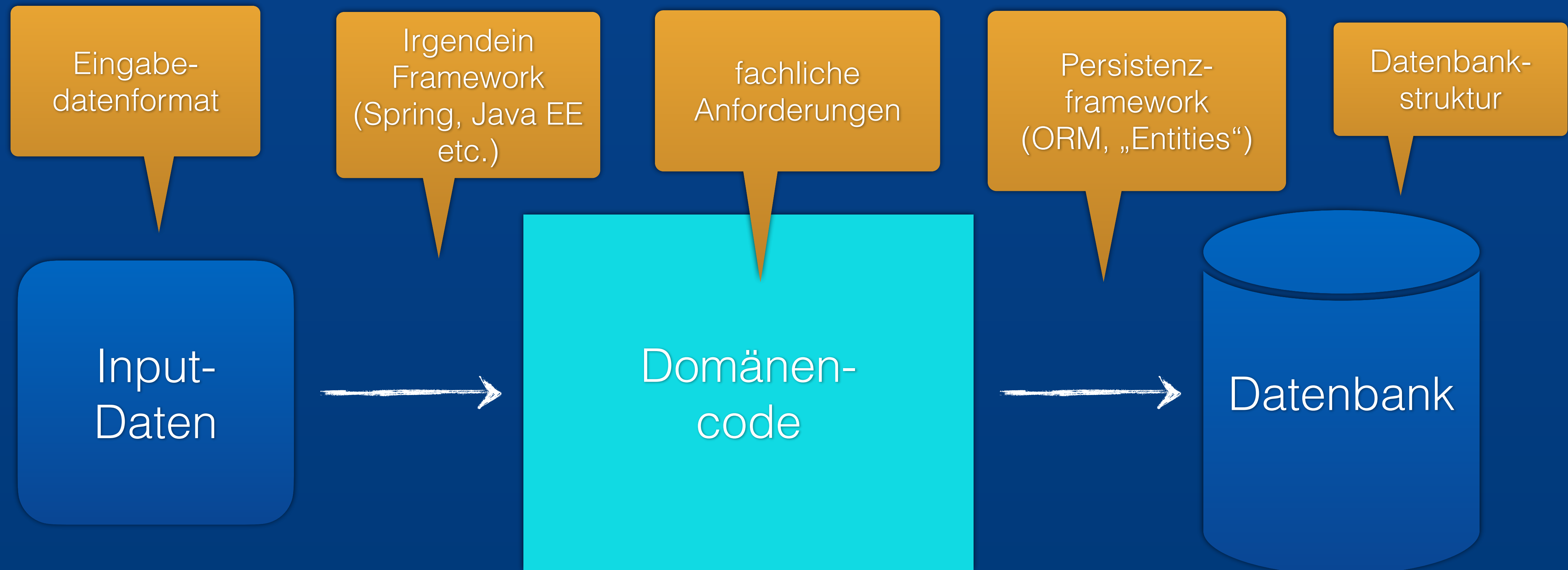
Architektur

Typisches Beispiel



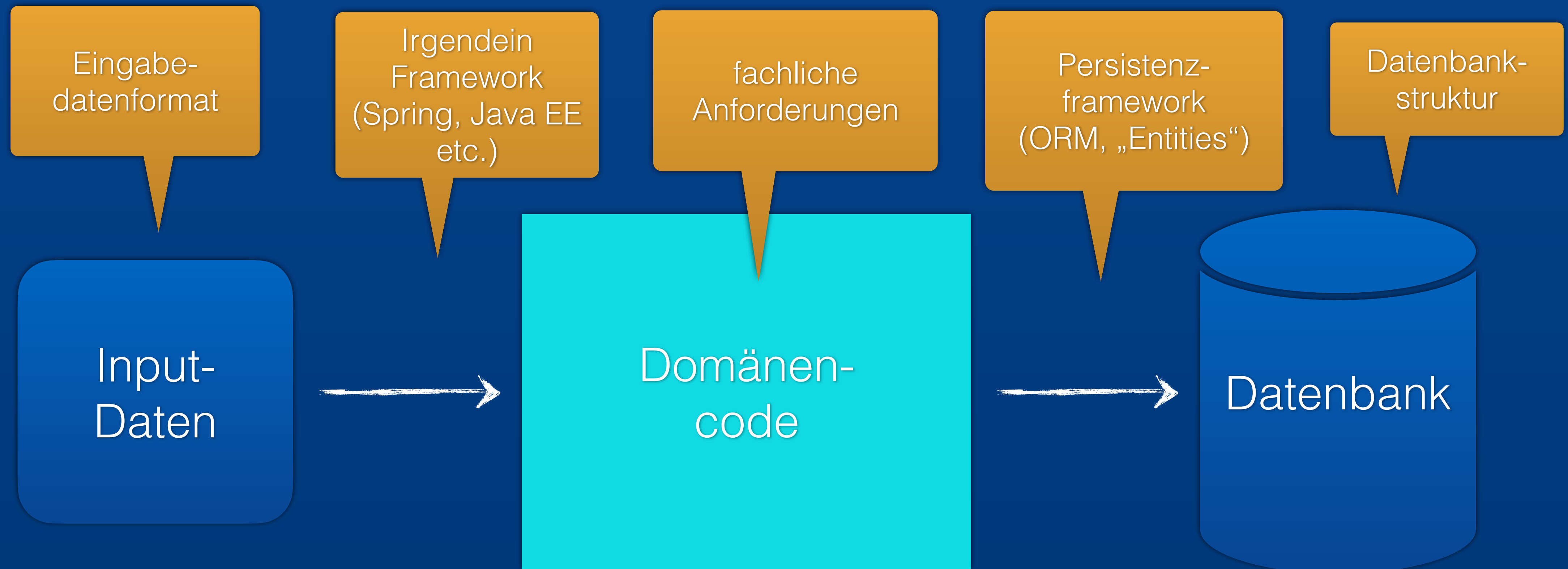
Viele, viele Abhängigkeiten

Abhängigkeiten sind oft **nicht unter unser Kontrolle**

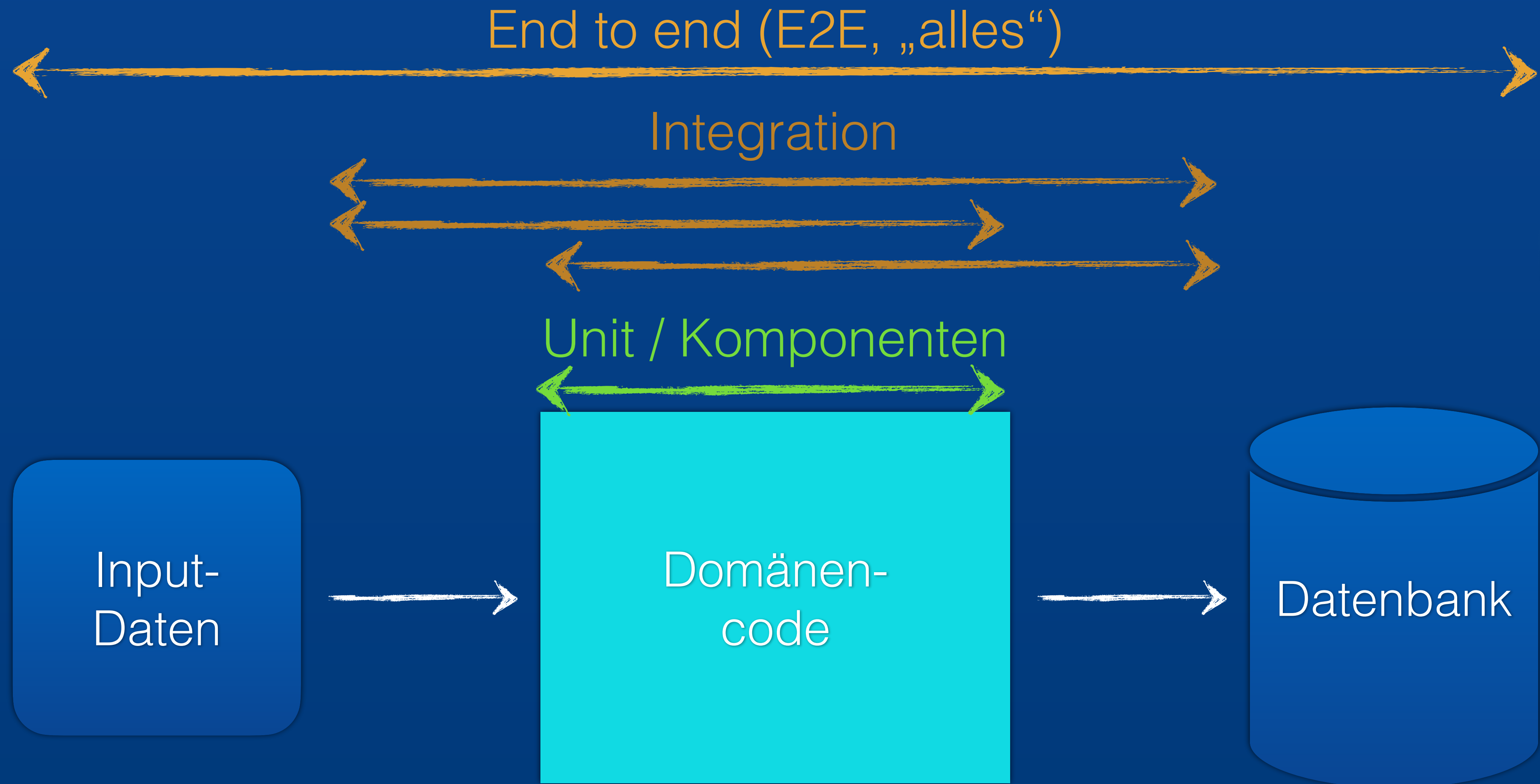


Abhängigkeiten – Gründe für Änderungen

... Abhängigkeiten, die den Code kaputtmachen
... **Abhängigkeiten, die die Tests kaputtmachen**



Was wird wo getestet?



relativ
schwer
&
langsam
zu
testen

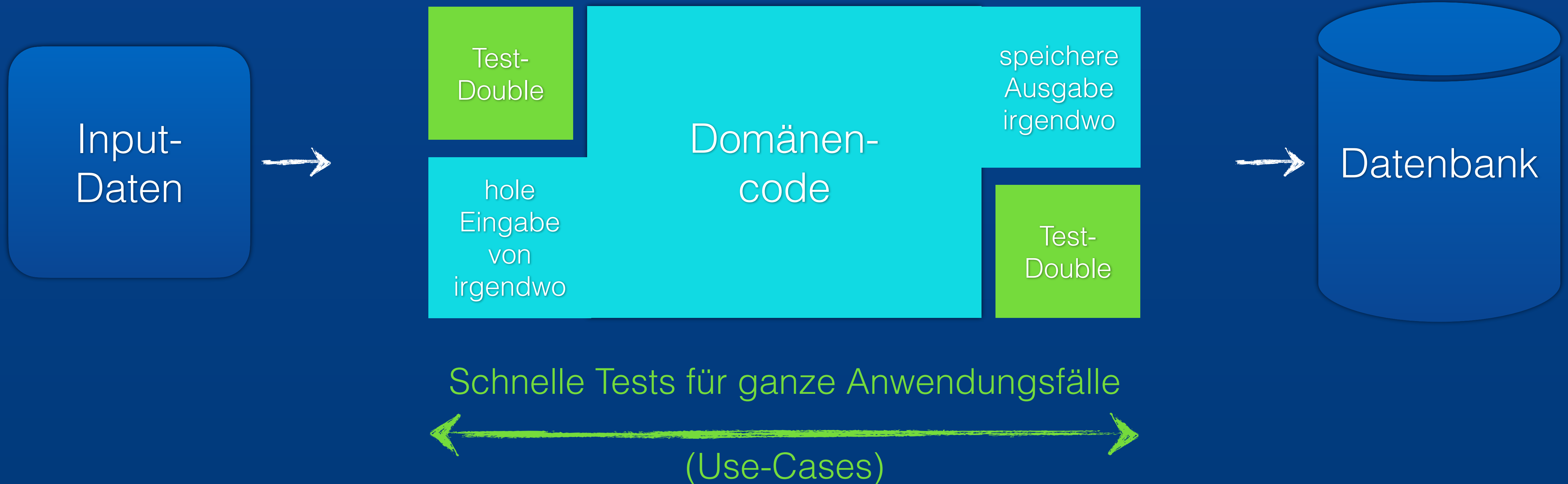
Wenige, beherrschbare Abhängigkeiten



Abhängigkeiten nach außen ...



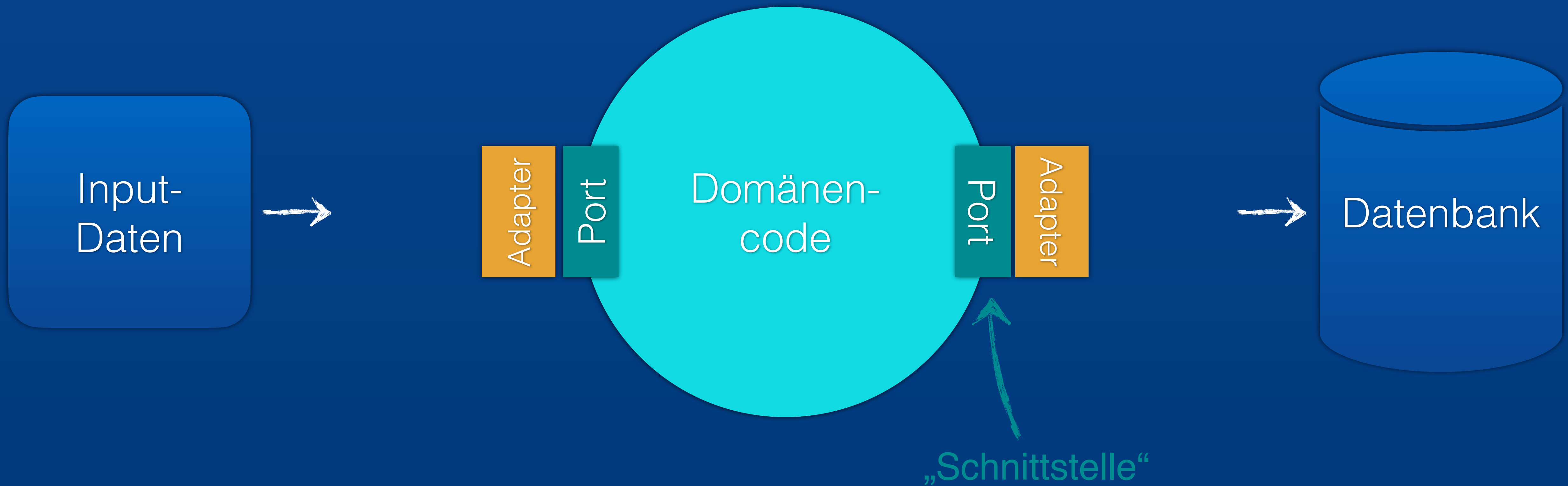
Architekturmuster



Architekturmuster



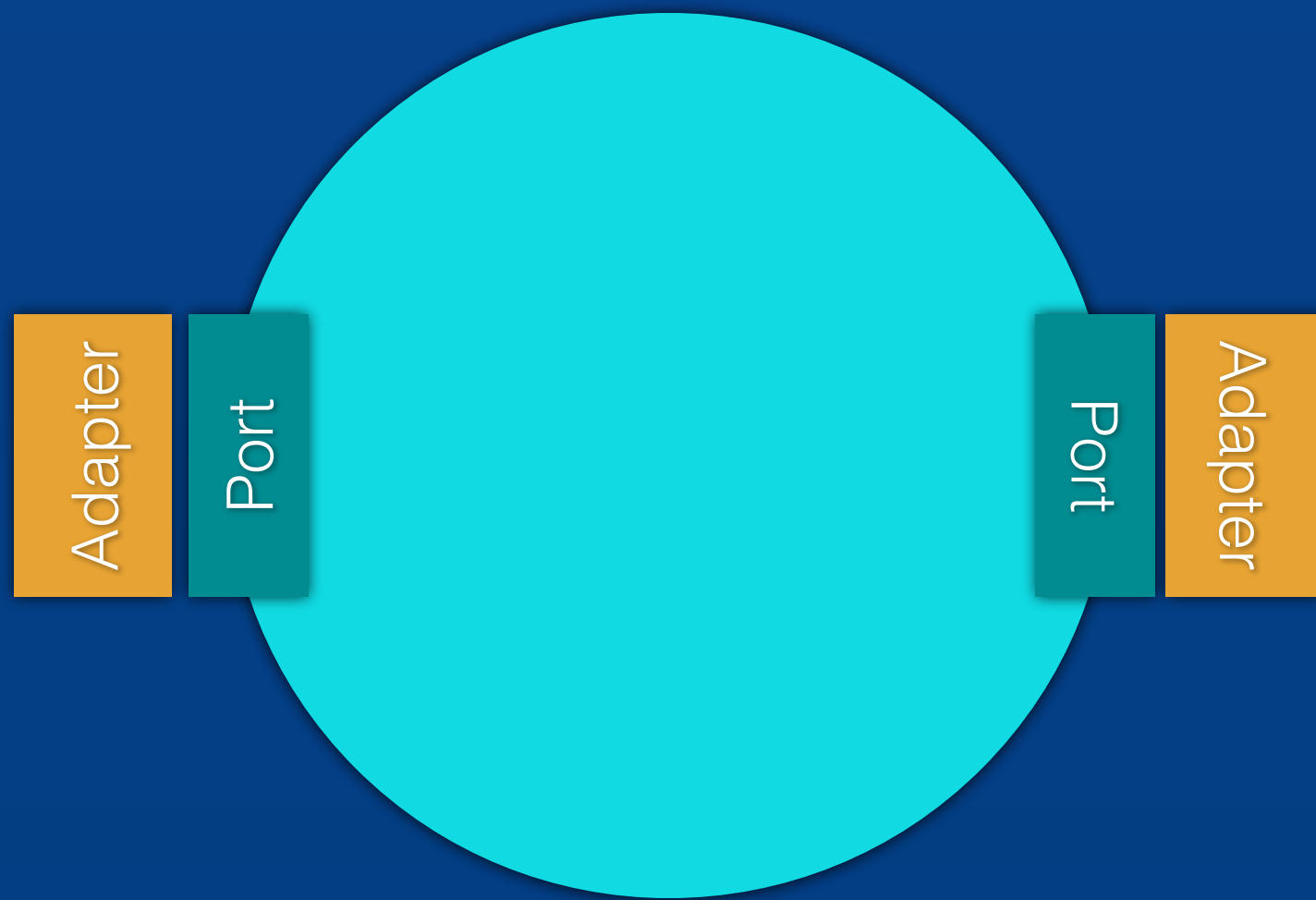
Architekturstile



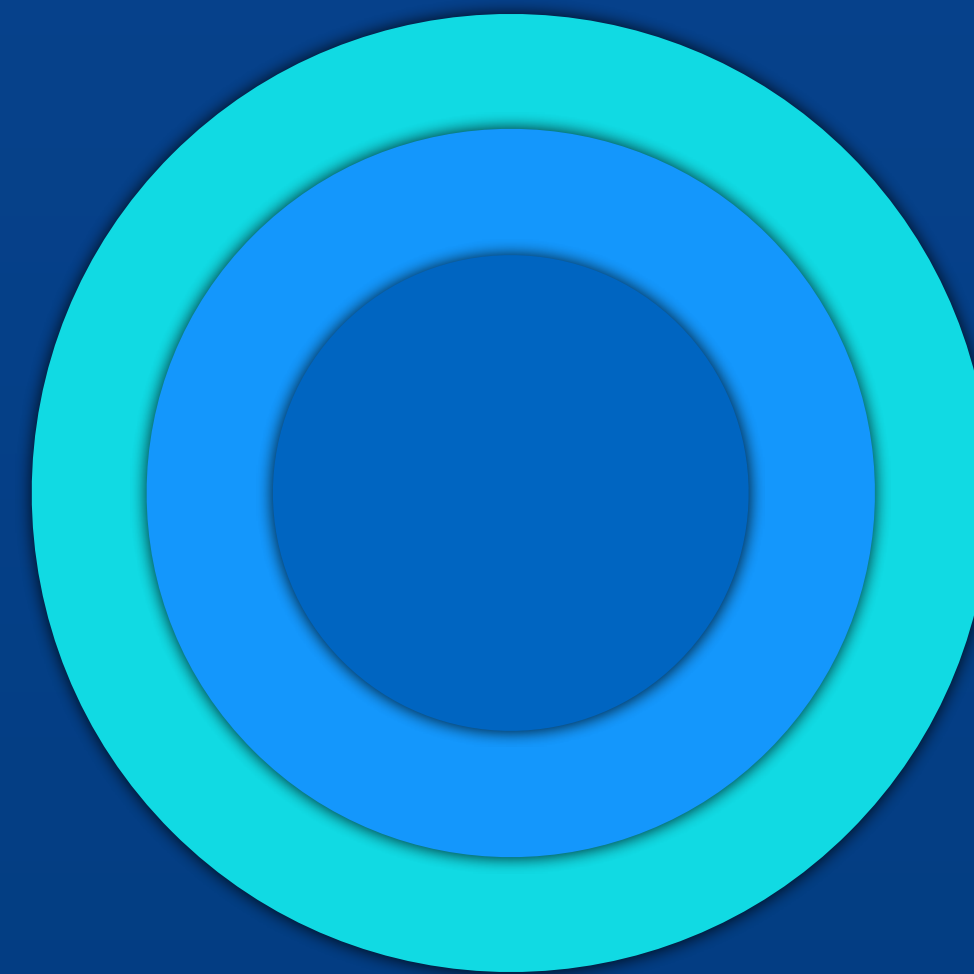
Architekturstile

Beispiele!

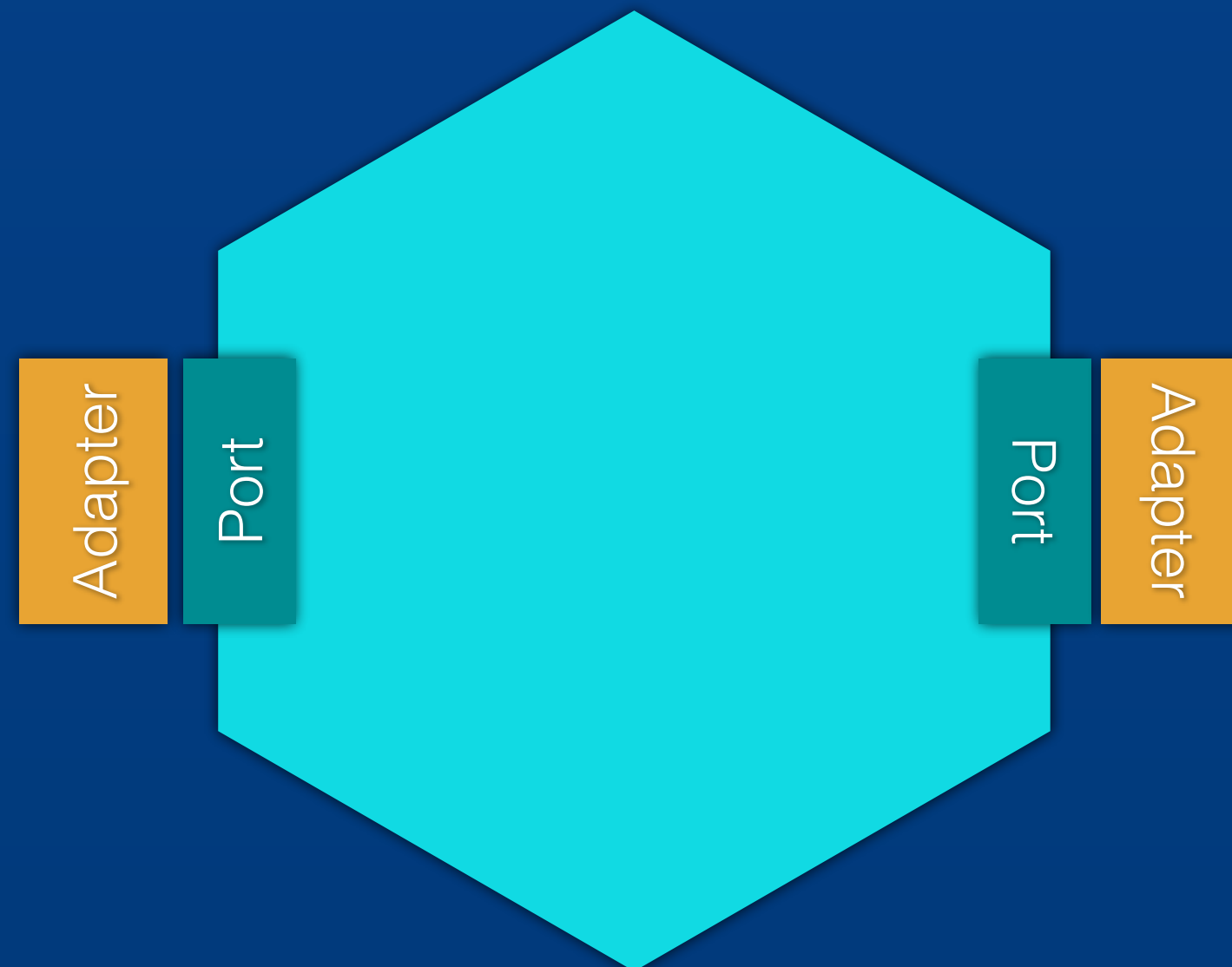
Ports & Adapters



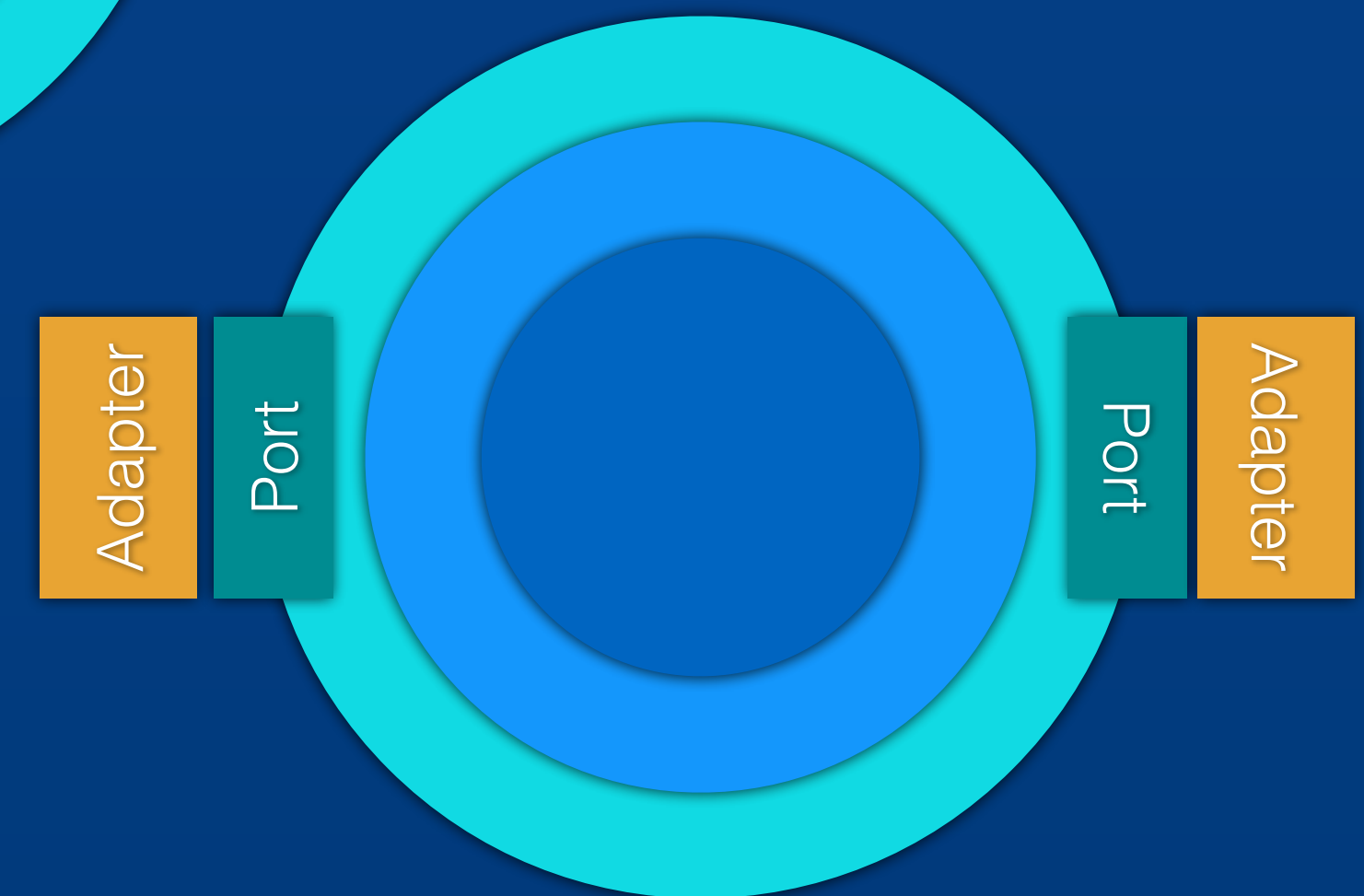
Onion (Ringe)



Hexagonal



u.a.



Innen & außen

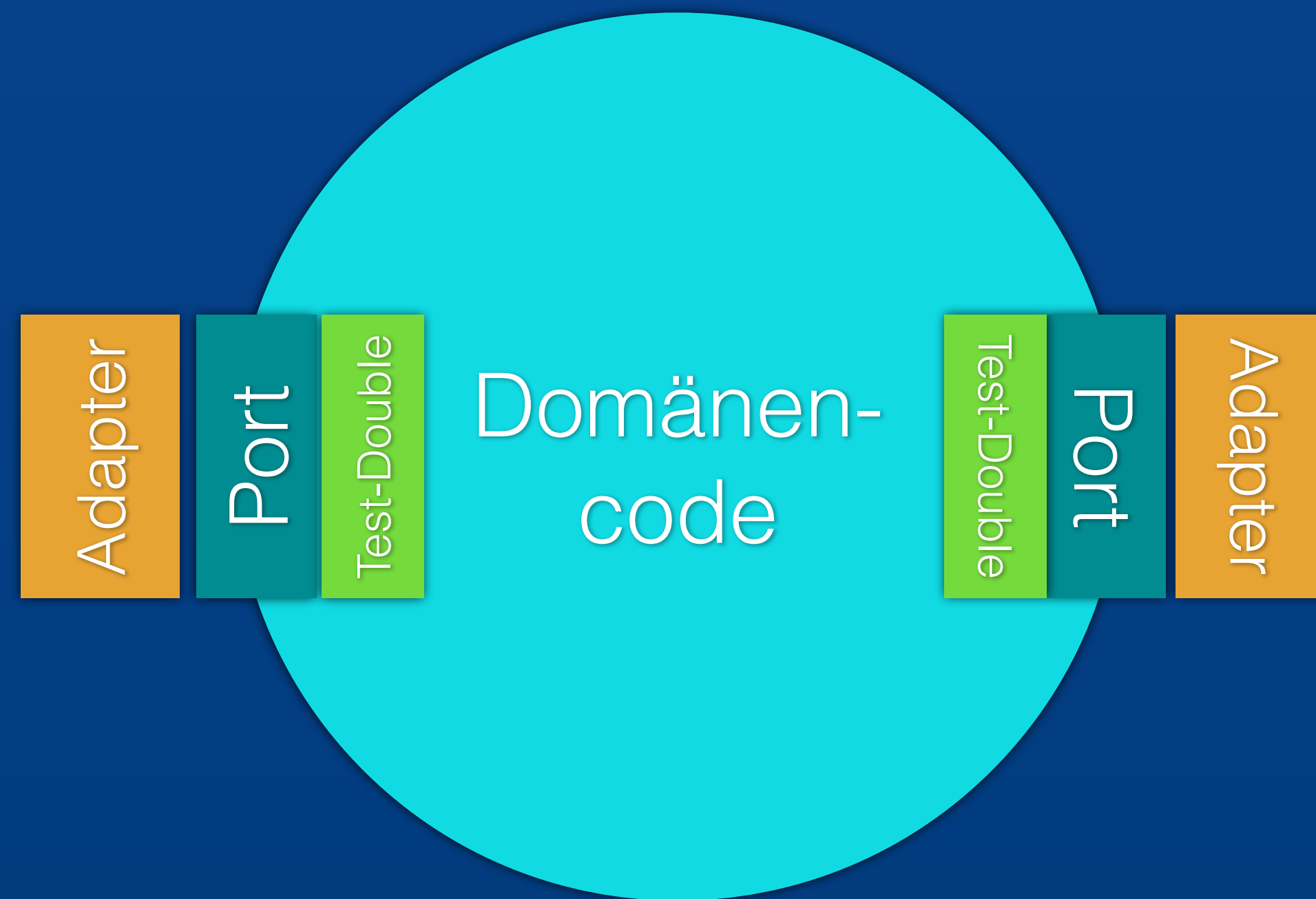
Stärker unter
unser Kontrolle

Weniger unter
unser Kontrolle



erlaubte Abhängigkeiten

Schnelle Tests



Schnelle, stabile,
umfassende **Tests**



Fazit & Ausblick

Wie Testbarkeit eure Agilität unterstützt

Schnelle Tests liefern
kontinuierlich schnelles Feedback

Je **umfassender** die schnellen Tests sind,
desto **sicherer** lassen sich Änderungen vornehmen und ausliefern.

Dafür müssen Code-Design und Architektur auf **Testbarkeit**
ausgelegt sein, z.B. durch das **Beherrschen von Abhängigkeiten**.

Wie Testbarkeit eure Agilität unterstützt

Moderation gefragt

Welche Abstraktionen lohnen sich? Was ist **angemessen**?

Wie können **schnelle Tests** erreicht werden?

Alle Rollen sind **von Anfang an** beteiligt

Testbarkeit wird **von Anfang an** mitgedacht

Die richtigen **Fragen** fragen

„**Wie** werden wir das testen?“

„**Warum** ist das schwer zu testen?“

In unterschiedlichen Rollen:
Entwickler. Architekt. Tester. **Coach.**

„Was müssten wir **ändern**,
um bessere Testbarkeit zu erreichen?“

„Wäre es **gut genug**, ... zu testen,
wenn wir ... separieren würden?“

Testbarkeit **von Anfang an**

User Story & Akzeptanzkriterien

Aufgabengröße

Aufgabenschnitt

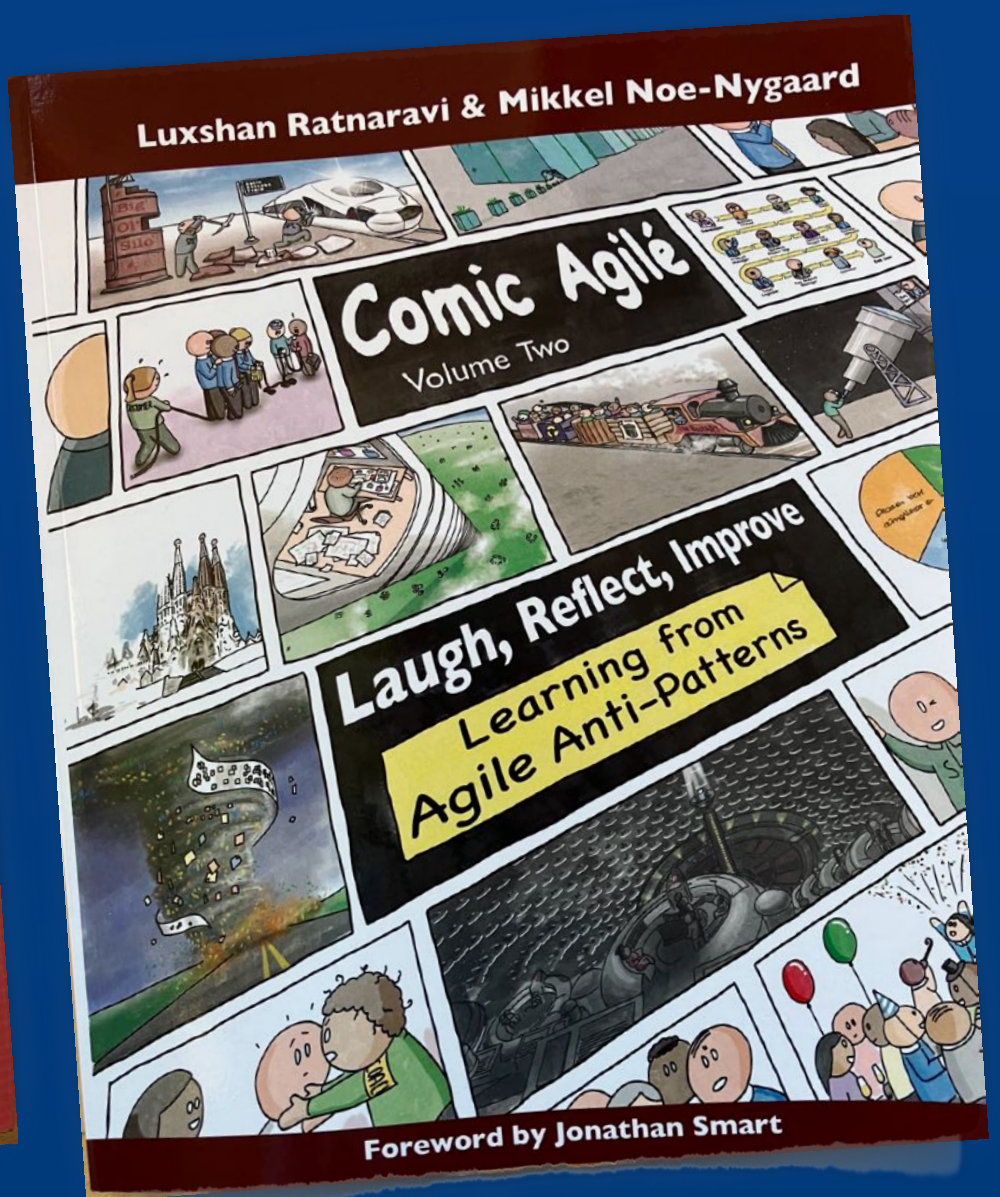
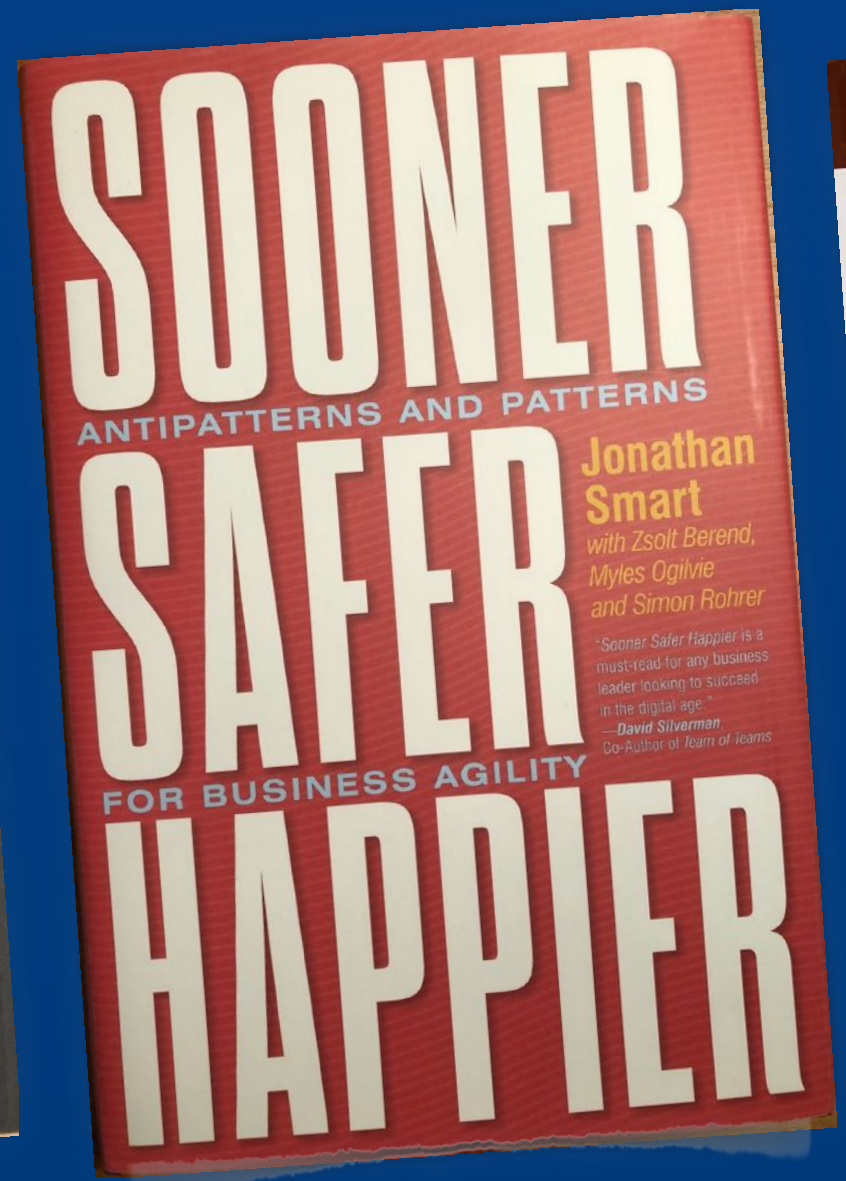
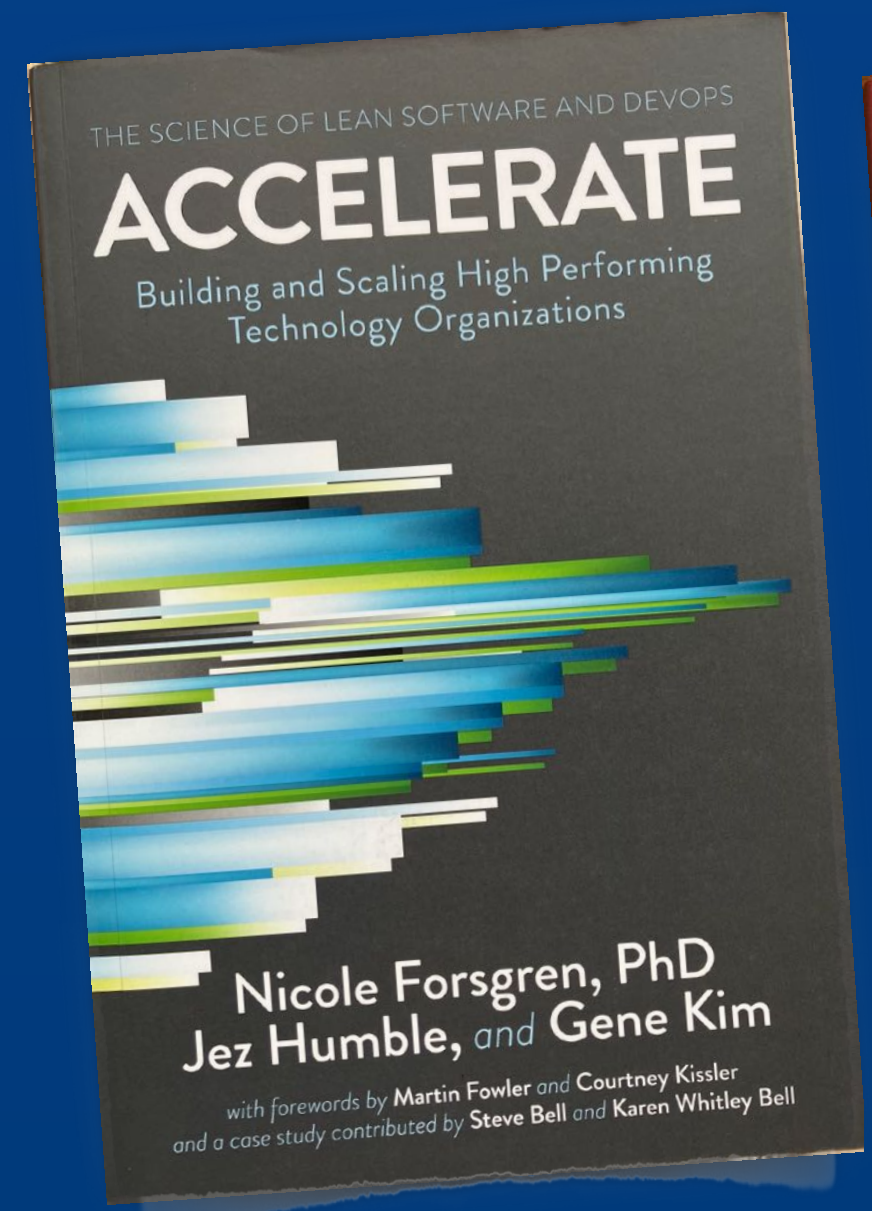
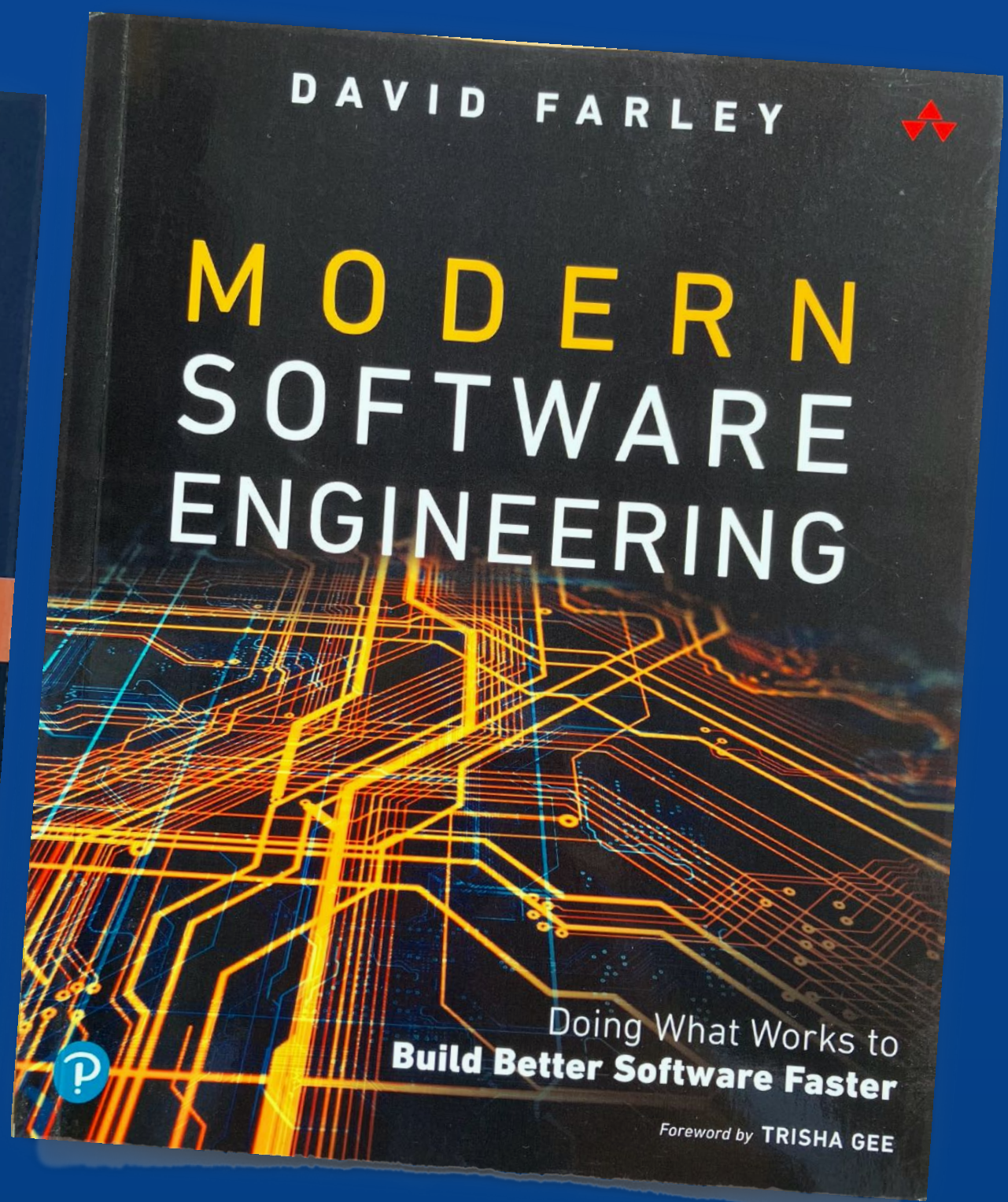
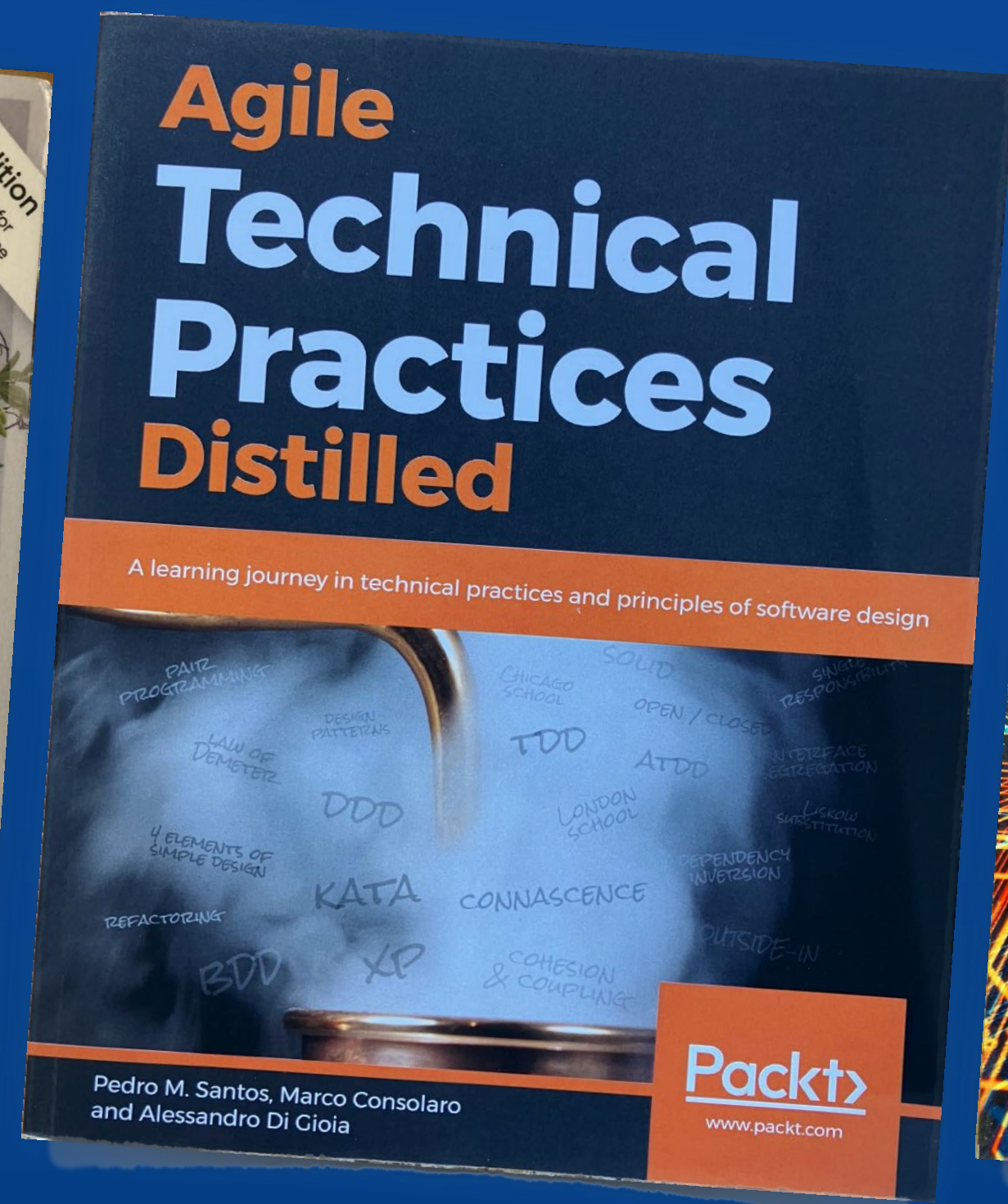
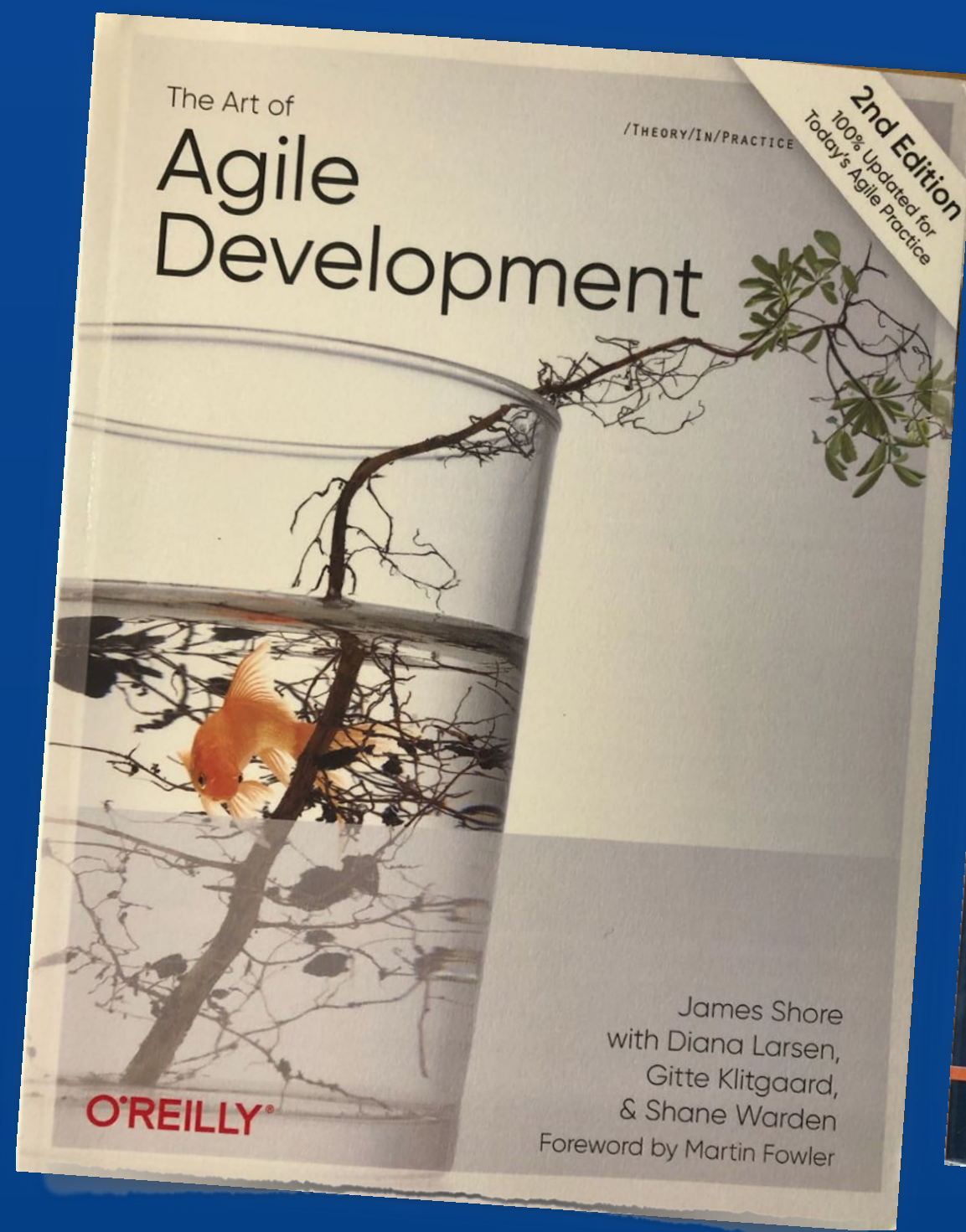
...

Zusammen arbeiten.

Zusammen coden.

Zusammen testen.

Voneinander lernen.



SCRUM MASTERS HH

Vielen Dank 😊



www.tk.de/IT

   @thmuch

Testgeschwindigkeitspyramide?!

