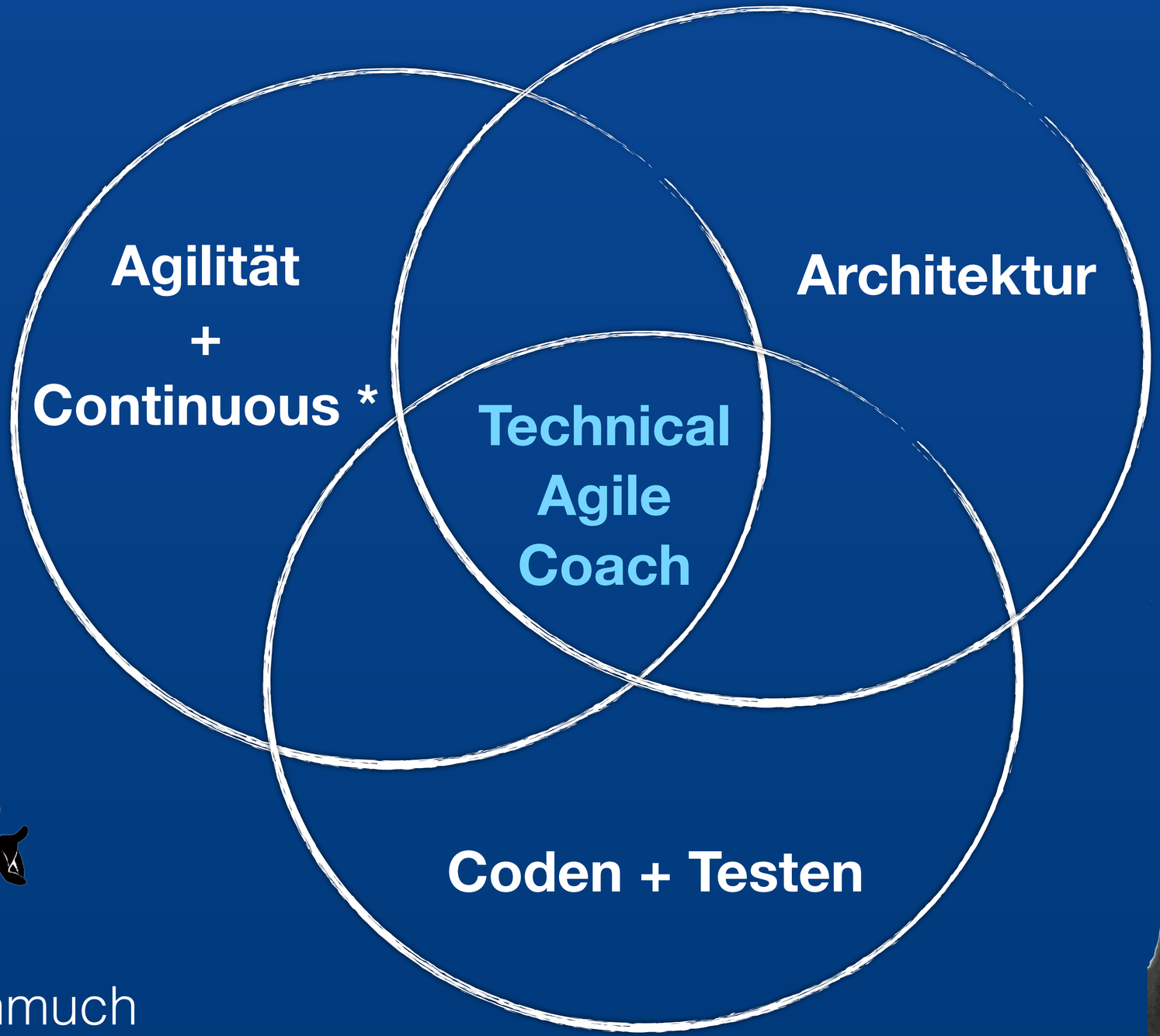
Einfach gut testbar

Grundlagen von Code-Design und Architektur für gute Testbarkeit

Thomas Much
@thmuch

21. März 2023



Die Techniker

www.tk.de/IT

@thmuch

Testbarkeit

Automatisierung **praktikabel**

Testdaten **unter Kontrolle**

Schnelles Feedback

Testpyramide & Co.

manuelle Tests

manuelle (explorative) Tests

UI (E2E) Tests

Integrations-tests

Unit-Tests

Testbarkeit

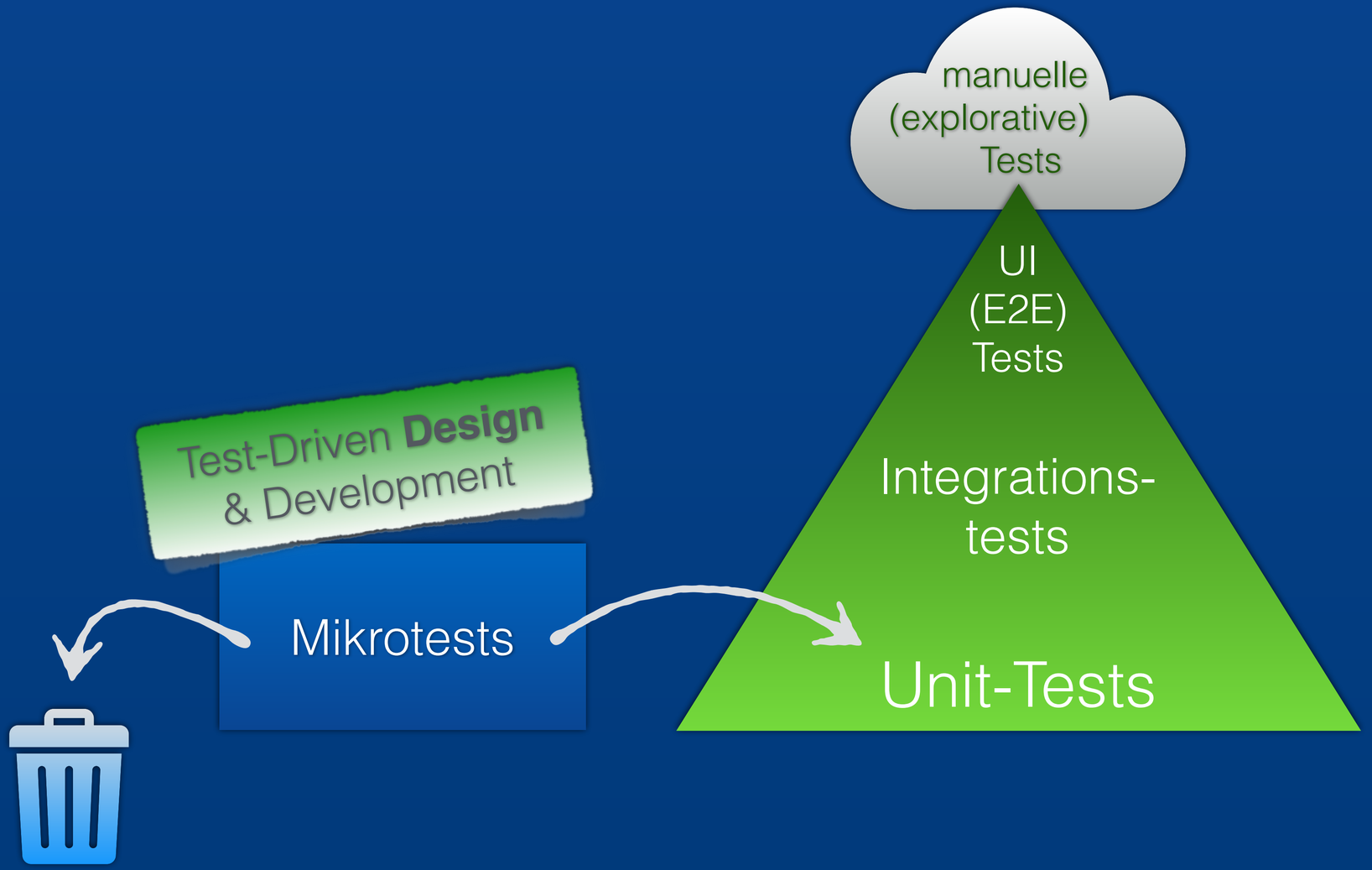
Containerisierung

Ice cream cone emoji inspired by Neal Brooks ;-)

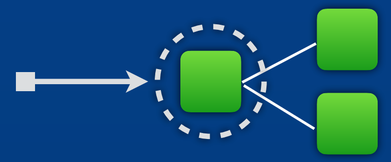
Schnelle Tests

Viele
umfassende
schnelle
stabile
Tests

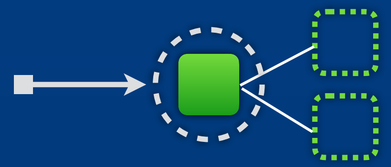
Units oft größer als Mikrotests!



„Sociable Tests“



„Solitary Tests“



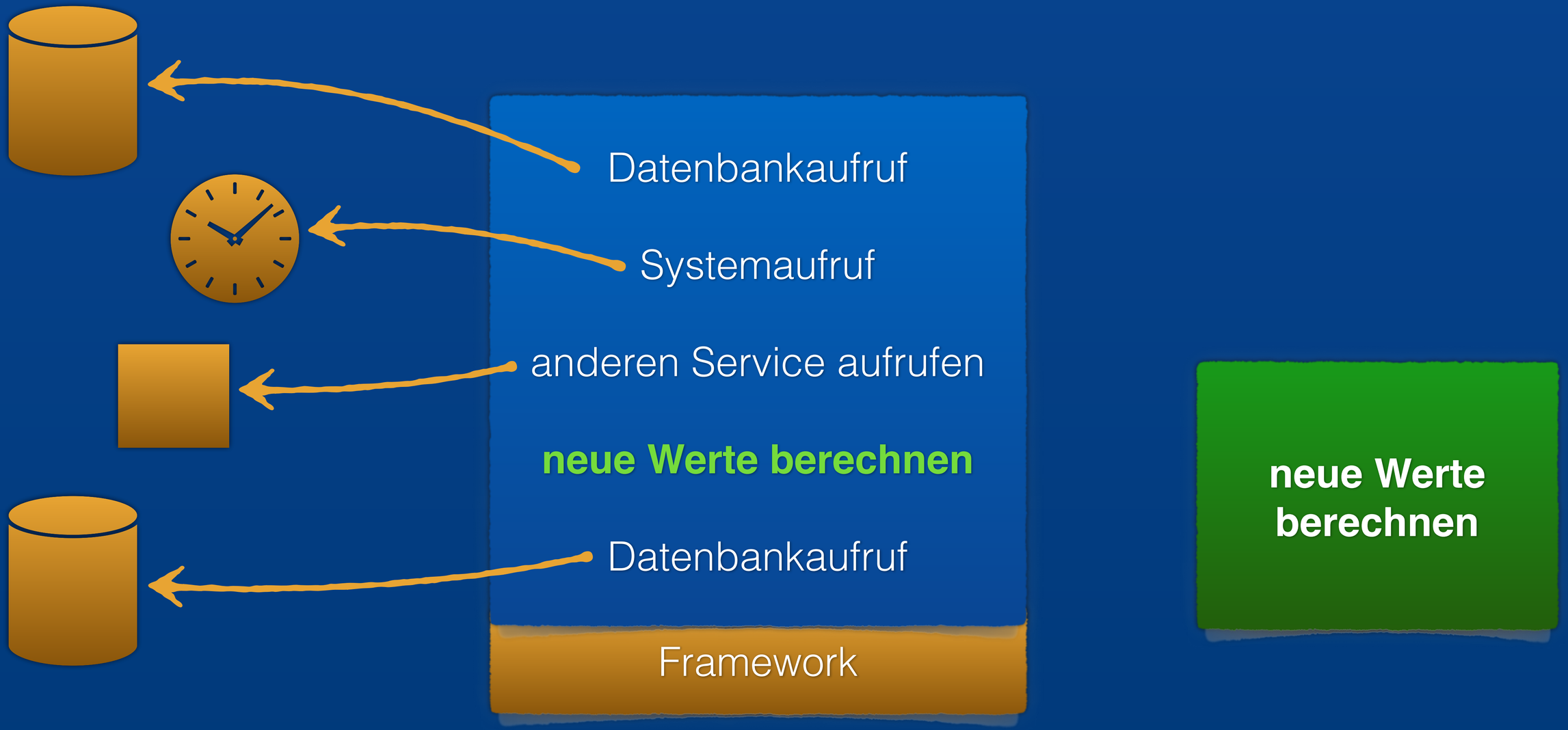
https://martinfowler.com/articles/2021-test-shapes.html

Schnelle Tests

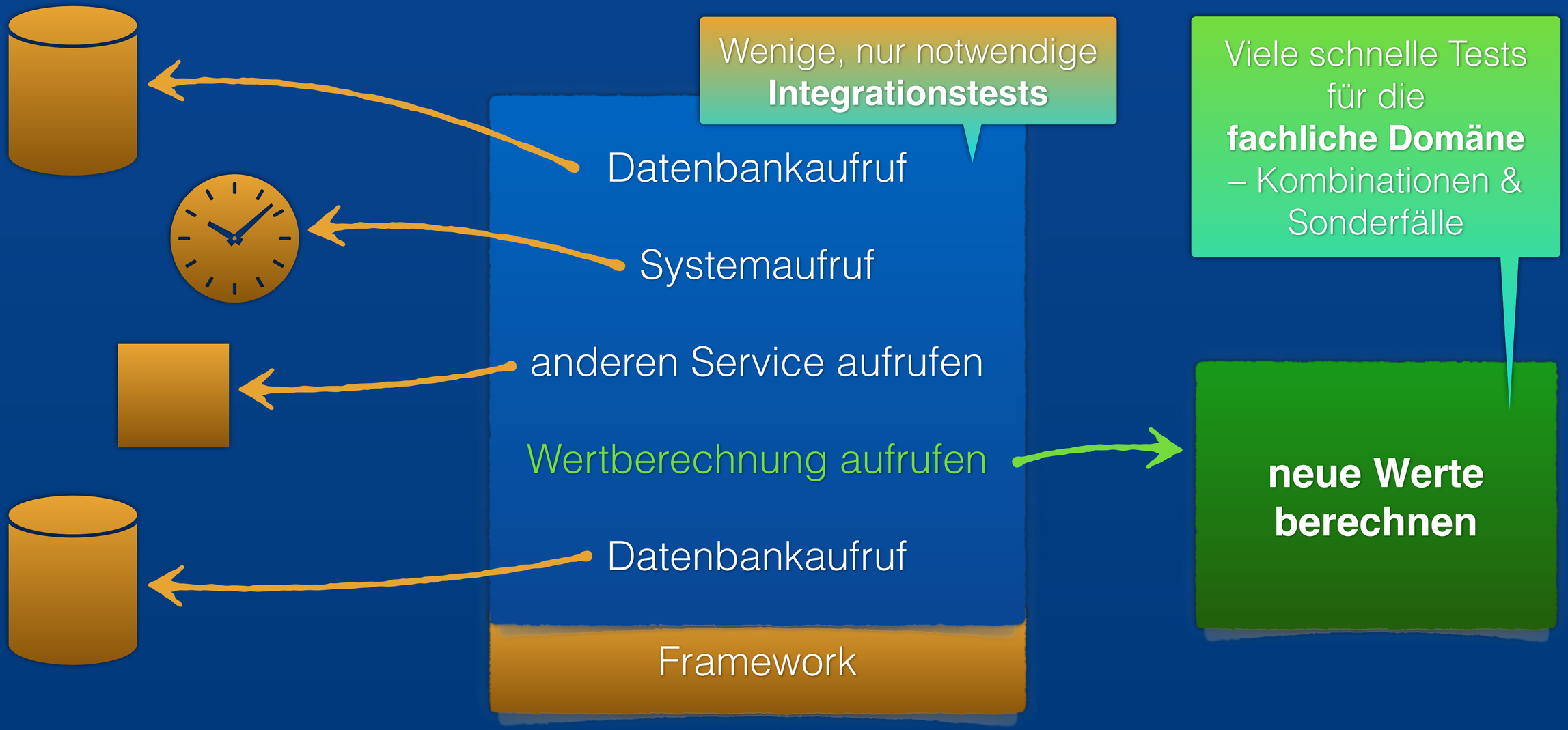
Code und **Architektur** müssen
für **Testbarkeit** designed sein

Abhängigkeiten
machen das schwierig

Abhängigkeiten



Abhängigkeiten



Trennung von
Integrationscode
und
Domänencode

Teil 1

Live Demo

Muster (Patterns) & Stile

Code-Design-Patterns

"Integration Operation Segregation Principle" (IOSP)

"Single Layer of Abstraction" (SLA)

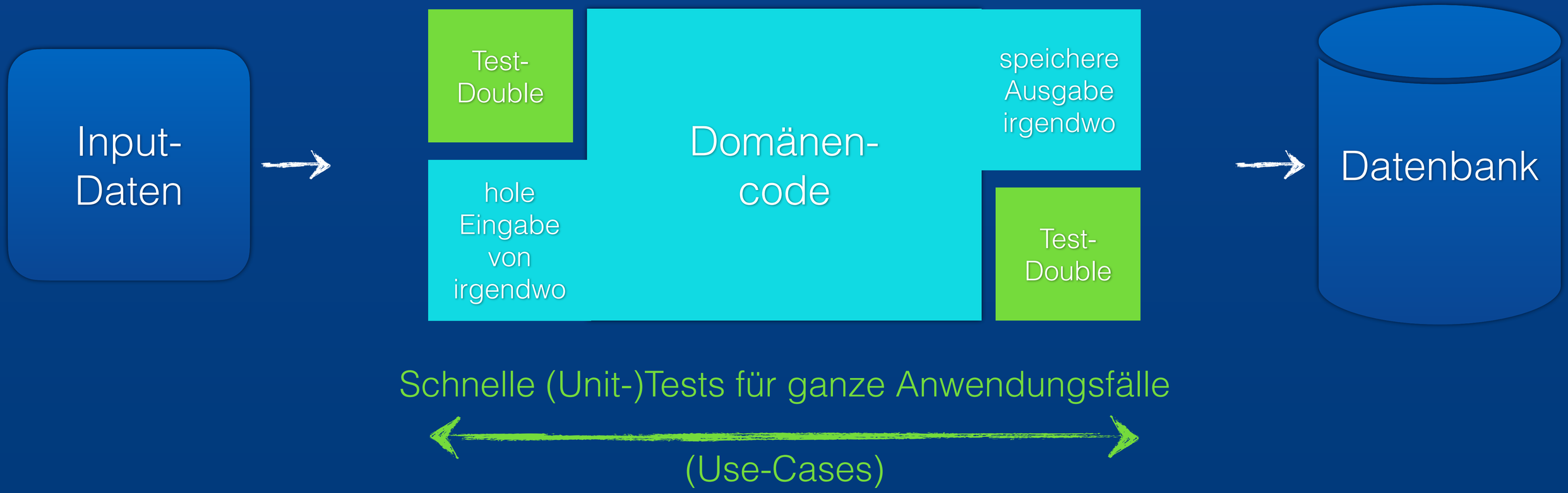
etc.

Vergleichbare **Muster & Stile** für die **Architektur**!

Typisches Beispiel

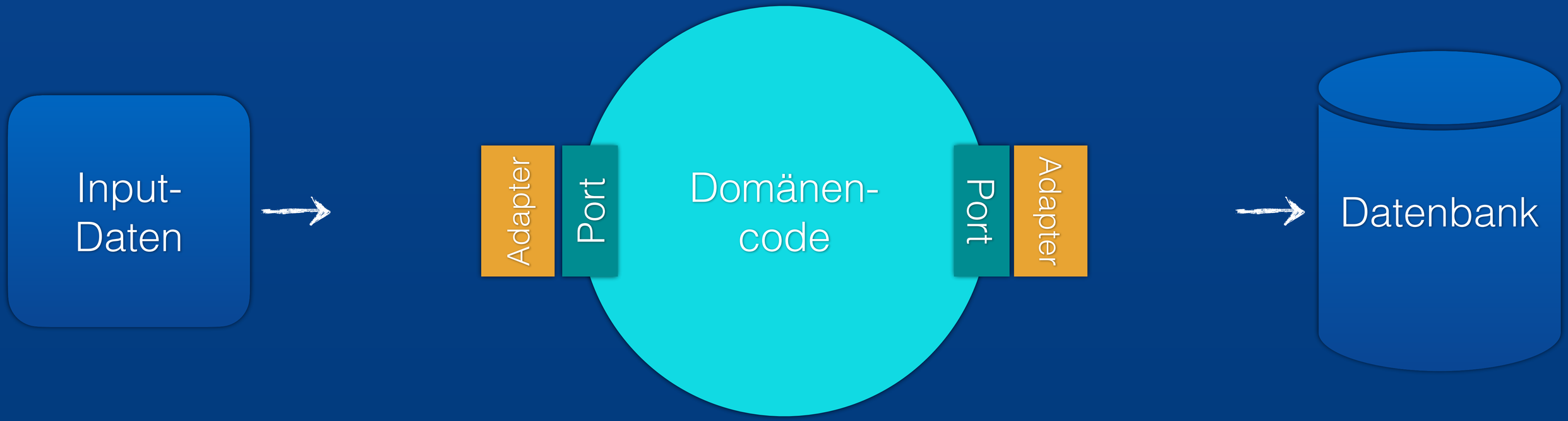
Abhängigkeiten nach außen …

Architekturmuster



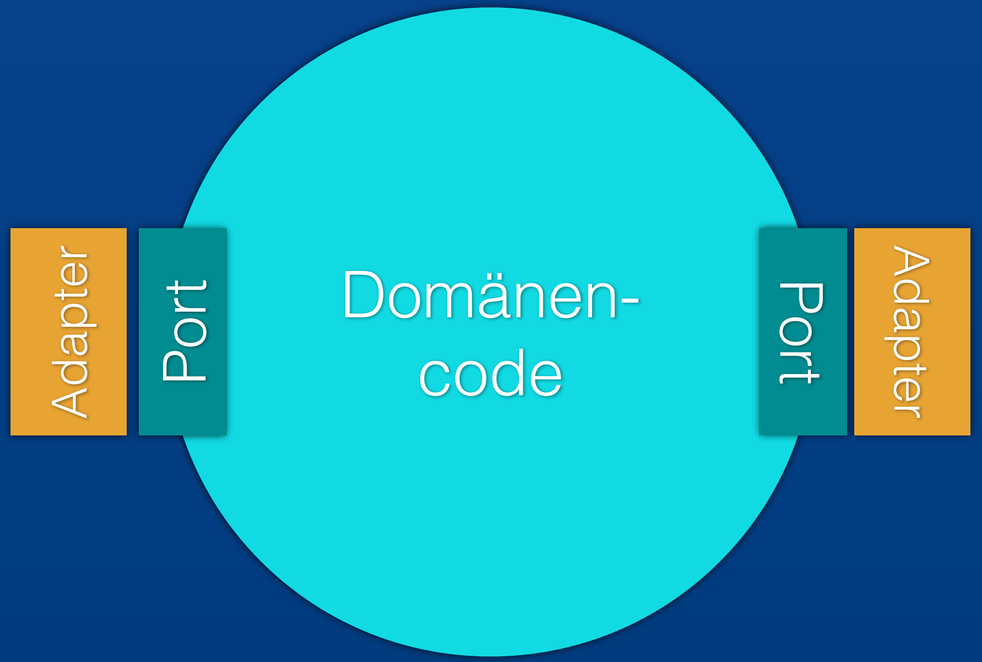
Architekturmuster

Architekturstile

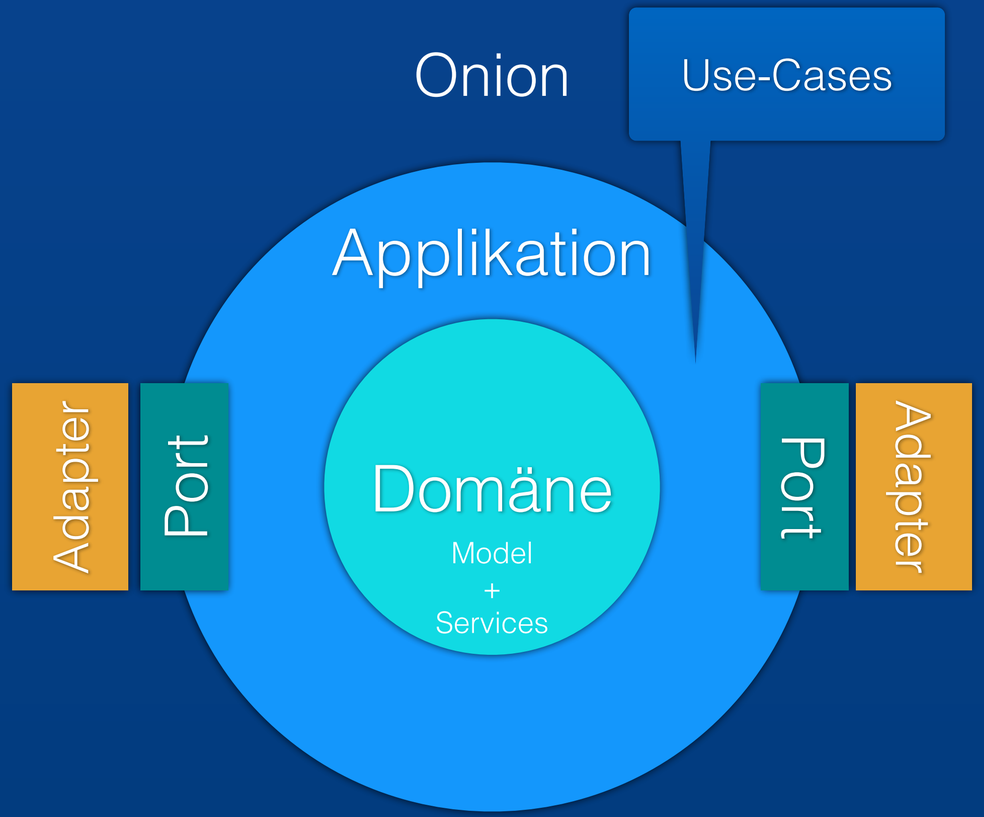


Architekturstile

Ports & Adapters ("Hexagonal")

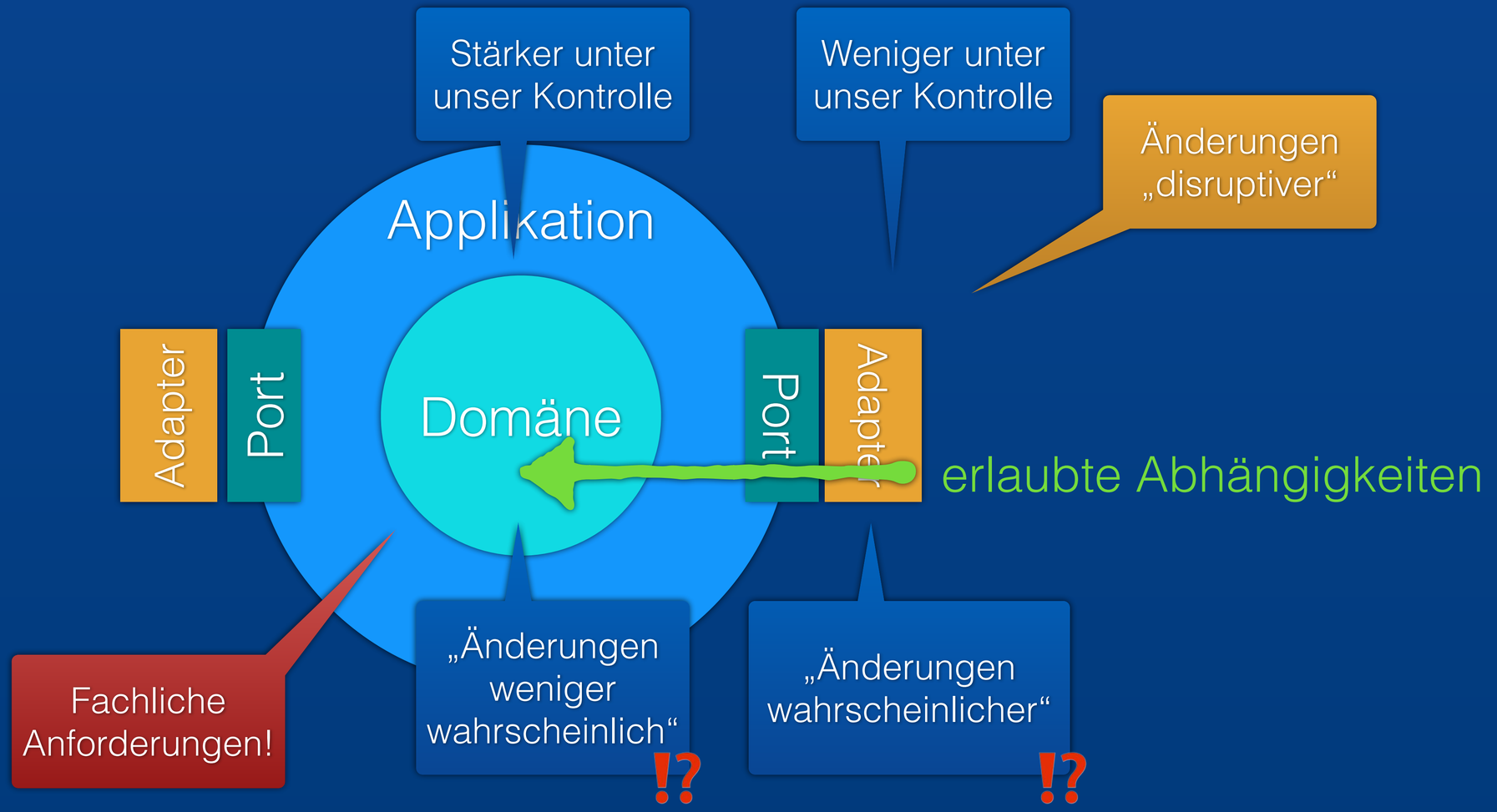


Onion

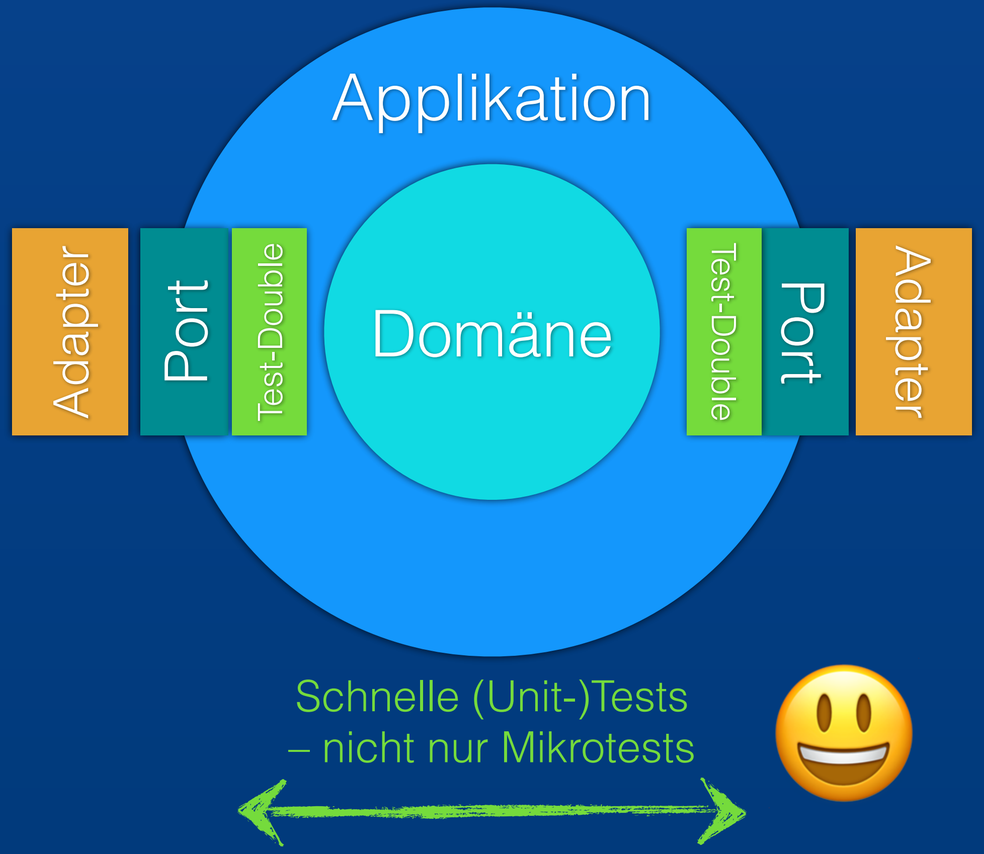


„Clean“ (irgendwo dazwischen)

Statt Schichten: Innen & außen!



Schnelle Tests für ganze Use-Cases



Teil 2

Live Demo

Testbaren Architekturstil prüfen

```
@ArchTest
void Onion_Architektur_wird_beachtet(JavaClasses analyzedClasses) {

    ArchRule rule = onionArchitecture()
            .domainModels("..domain.model")
            .domainServices("..domain.service")
            .applicationServices("..application")
            .adapter("REST-API", "..adapter.api")
            .adapter("Oracle-DB", "..adapter.datenbank")
            .adapter("SAP-Nummernkreis", "..adapter.zaehler");

    rule.check(analyzedClasses);
}
```

ArchUnit

*https://github.com/**thmuch**/**einfach-gut-testbar***

Fazit & Ausblick

Einfach gut testbar

… wenn **Code** und **Architektur** für **Testbarkeit** entworfen sind

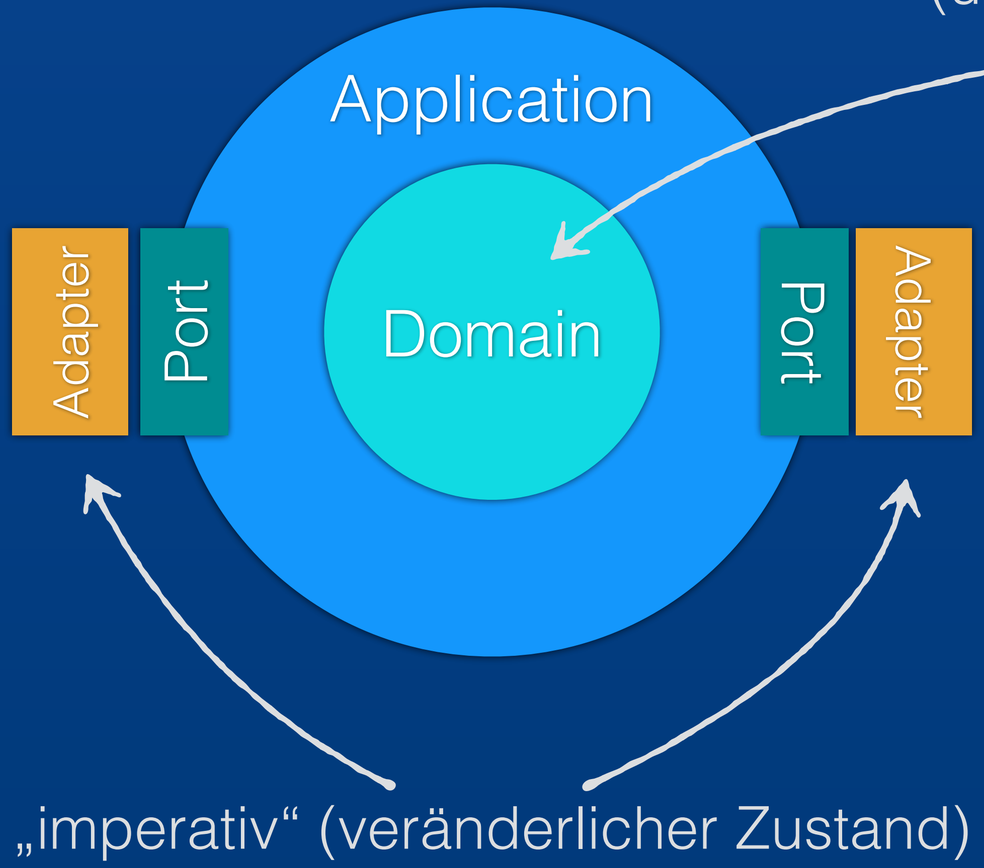
… wenn **Integrationscode** und **Domänencode** klar **getrennt** sind

… wenn **schnelle Tests** möglich sind

… auch für **größere Units** (> 1 Methode, > 1 Klasse)

Funktionaler Kern, imperative Hülle

Ausblick



Weiterdenken – Architekturarbeit!

Lohnen sich die Strukturen und Abstraktionen?

Ist **schnell zu Testendes** anders von
langsam zu Testendem unterscheidbar?

Wichtig sind

umfassende, **schnelle**, **stabile** Tests


Agile
Technical
Practices
Distilled
A learning journey in technical practices and principles of software design
Pedro M. Santos, Marco Consolaro
and Alessandro Di Gioia
Packt
www.packt.com
WORKING
EFFECTIVELY
WITH
LEGACY CODE
Michael C. Feathers
The Addison Wesley Signature Series
A Kent Beck Signature Book
Growing
Object-Oriented
Software,
Guided by Tests
Steve Freeman
Nat Pryce
DAVID FARLEY
MODERN
SOFTWARE
ENGINEERING
Doing What Works to
Build Better Software Faster
Foreword by TRISHA GEE
Robert C. Martin Series
PRENTICE HALL
Clean Architecture
A Craftsman's Guide to
Software Structure and Design
Robert C. Martin
Foreword by Kevlin Henney
Afterword by Jason Gorman
MANNING
Effective
Software
Testing
A developer's guide
Maurício Aniche
Forewords by Arie van Deursen and Steve Freeman

Testbarkeit

Schnelle Tests

Architektur

Code-Design

Fragen?

Unit-Tests

Abhängigkeiten

Mikrotests

Größe einer Unit

Fast Feedback Loops

@thmuch

Vielen Dank 😊

Die Techniker

www.tk.de/IT

@thmuch