



Bessere **Testbarkeit**

durch geeignete Architekturmuster und Code-Designs

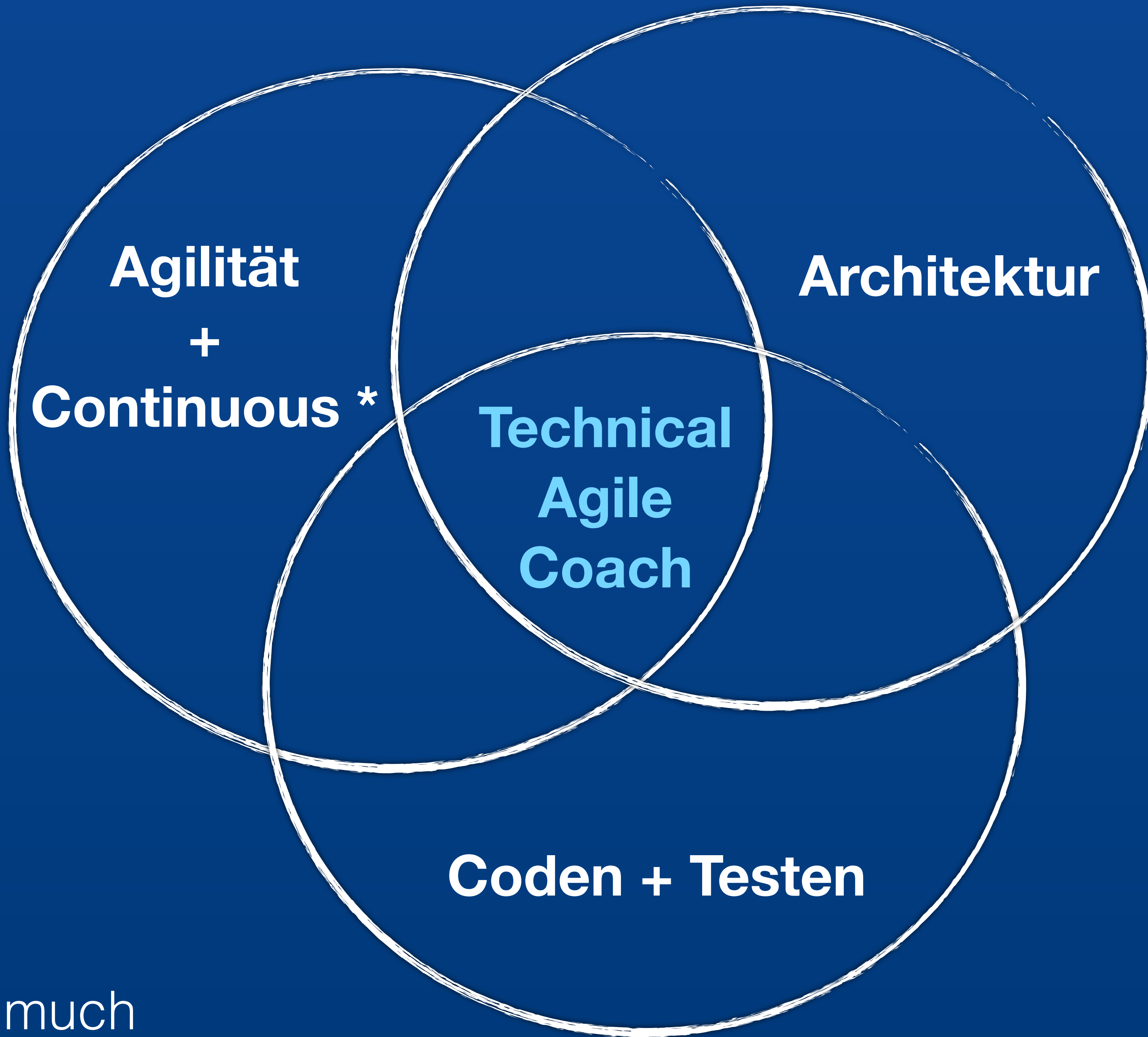
Thomas Much

  @thmuch

5. Juli 2023



www.tk.de/IT



  @thmuch

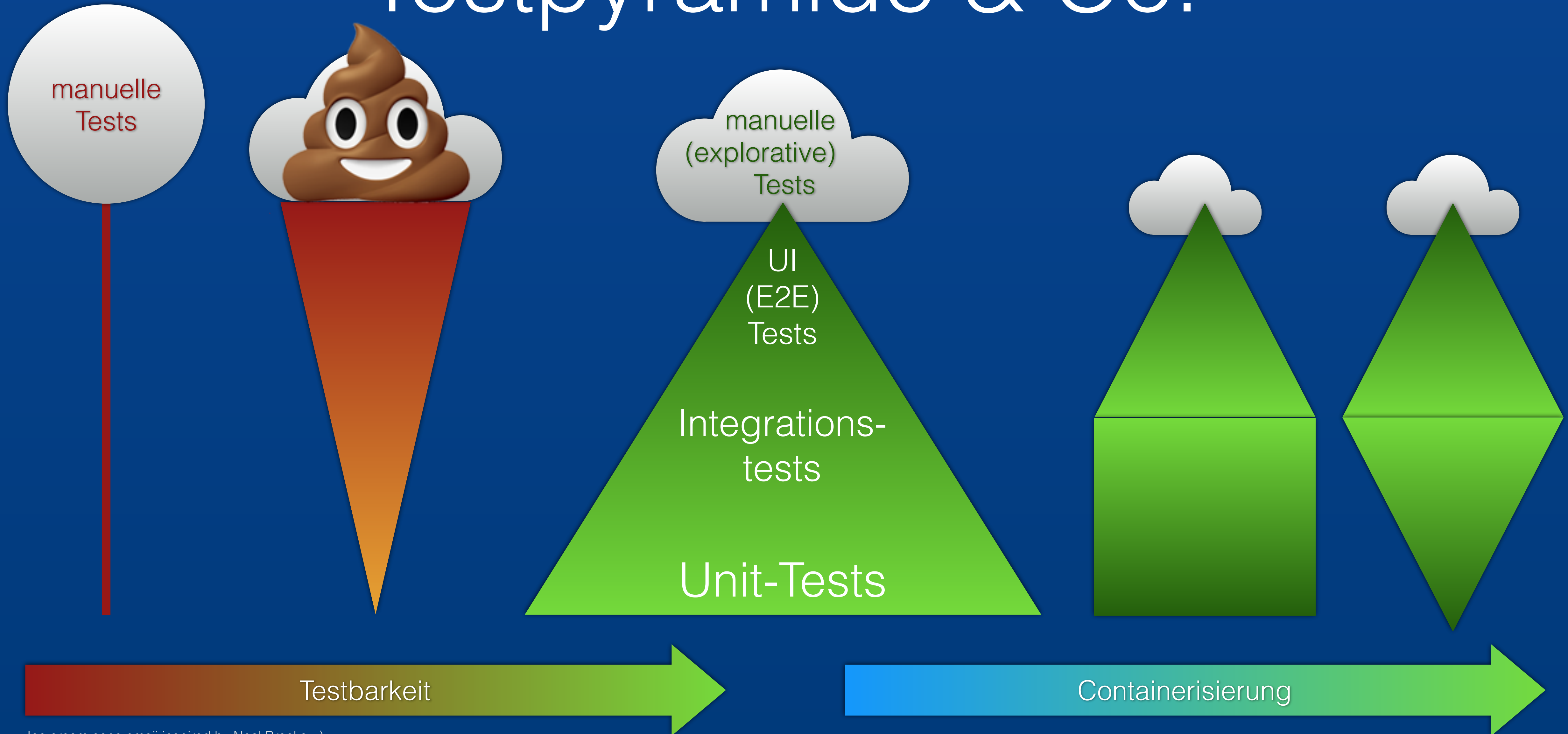
Testbarkeit

Automatisierung **praktikabel**

Testdaten **unter Kontrolle**

Schnelles Feedback

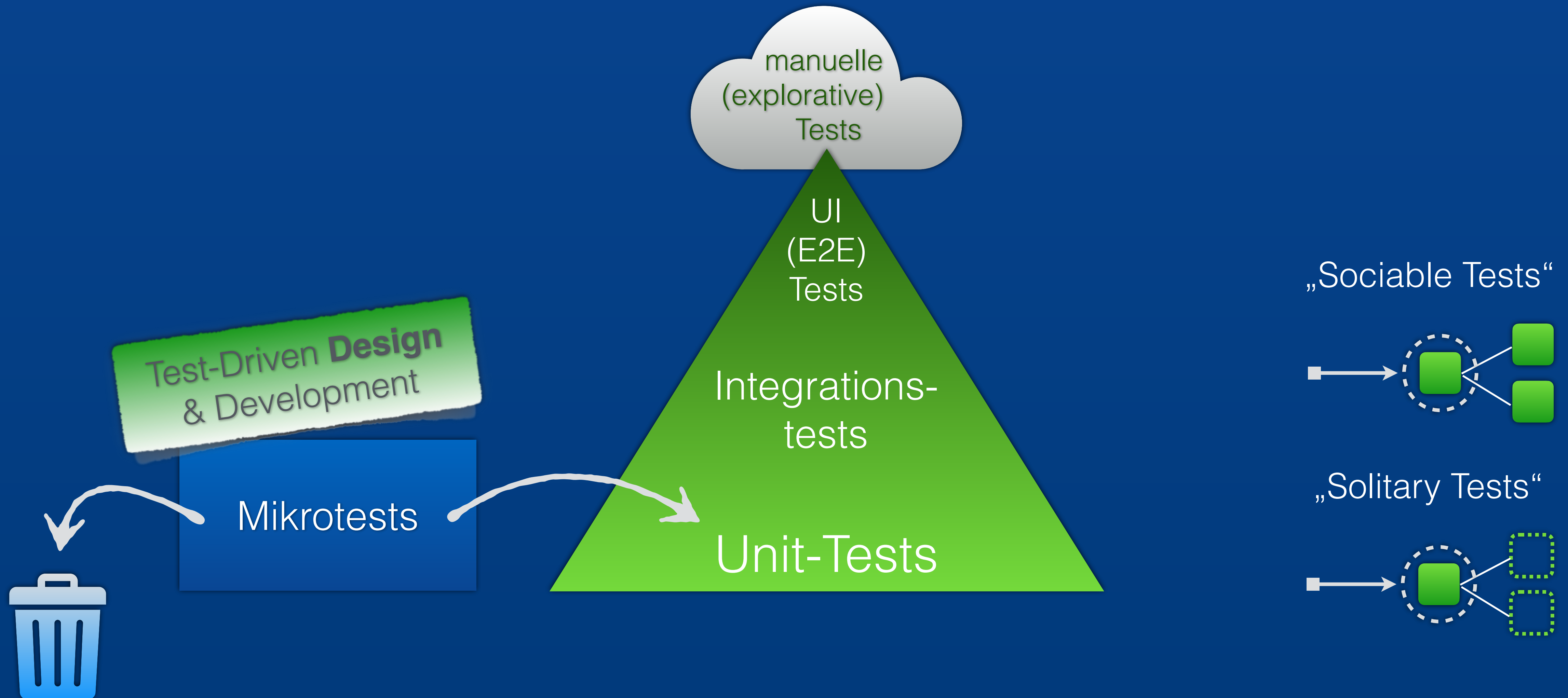
Testpyramide & Co.



Schnelle Tests

Viele
umfassende
schnelle
stabile
Tests

Units oft größer als Mikrotests!

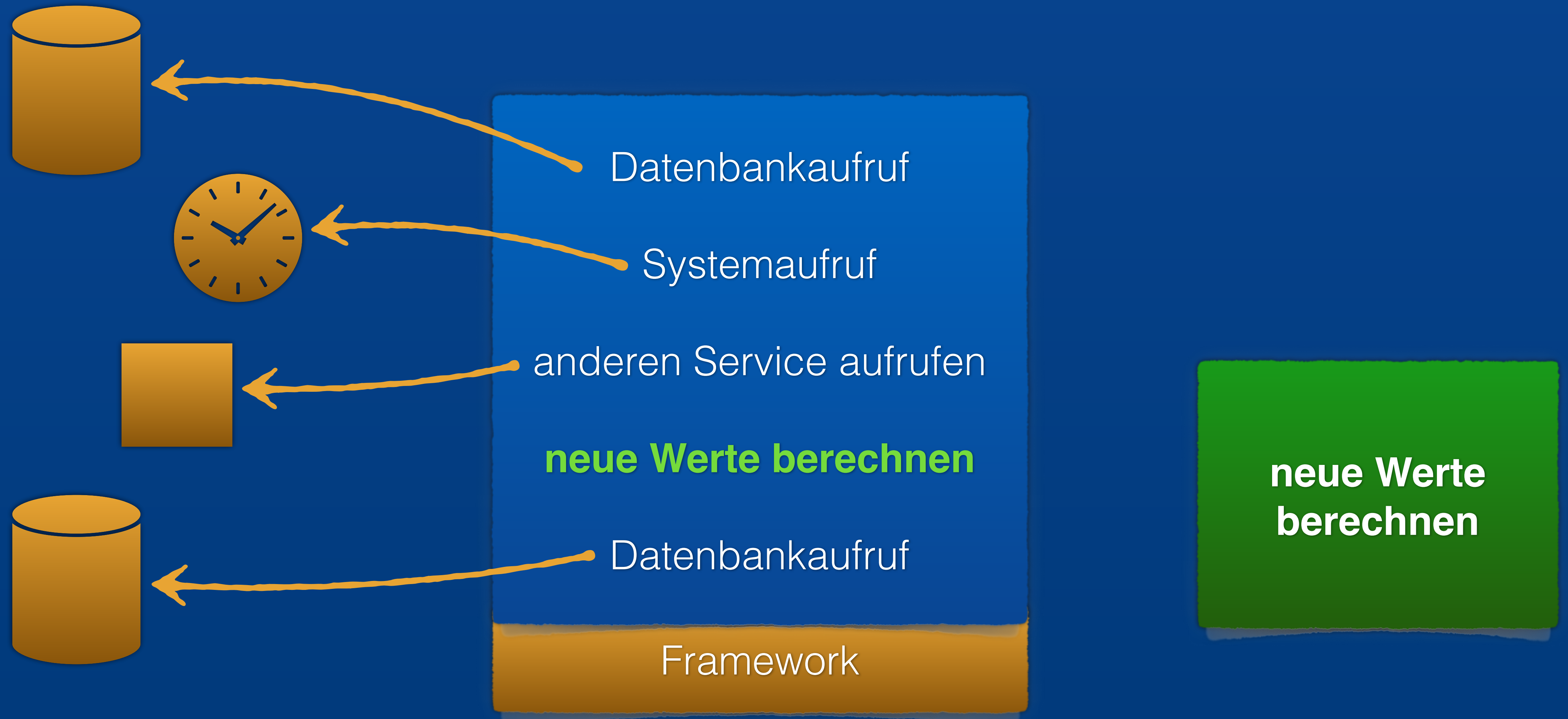


Schnelle Tests

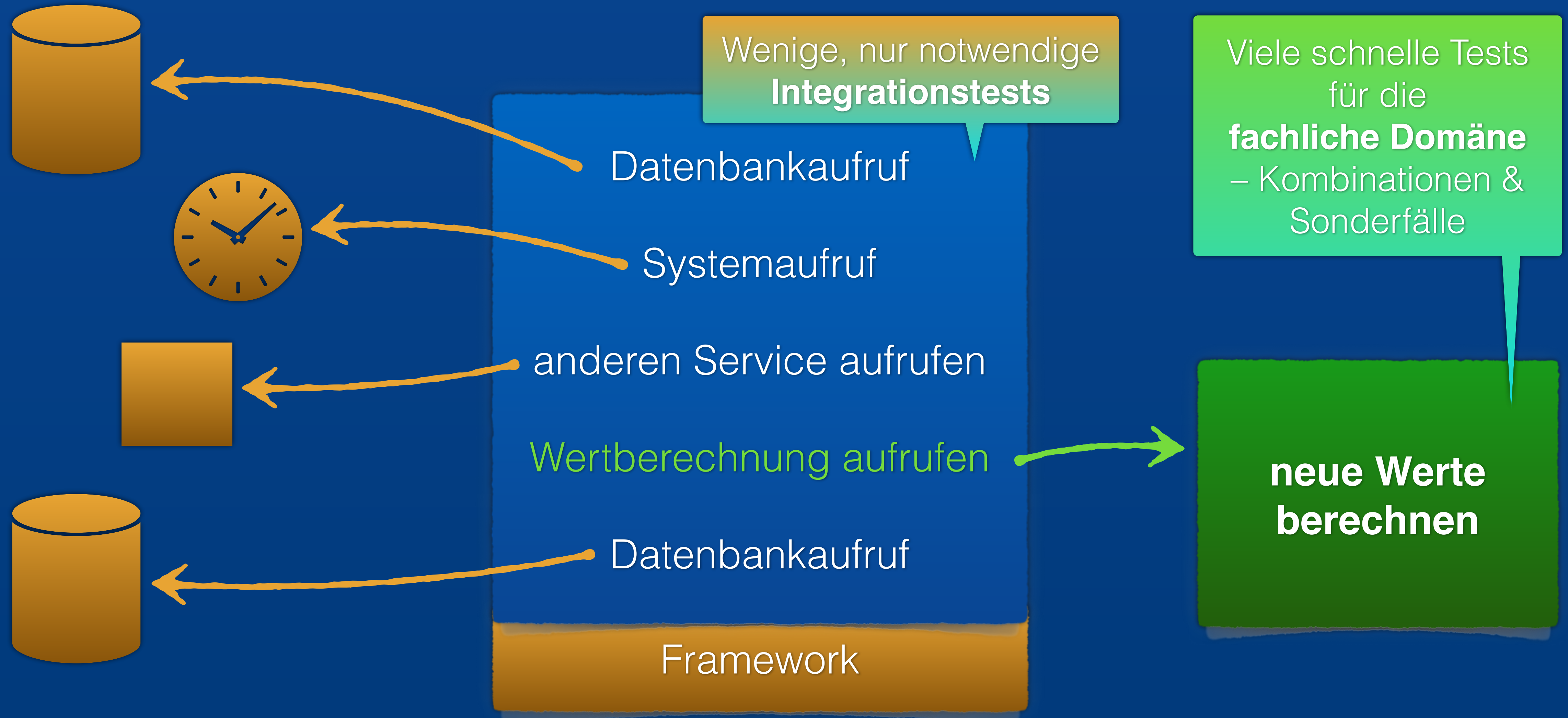
Code und **Architektur** müssen
für **Testbarkeit** designed sein

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



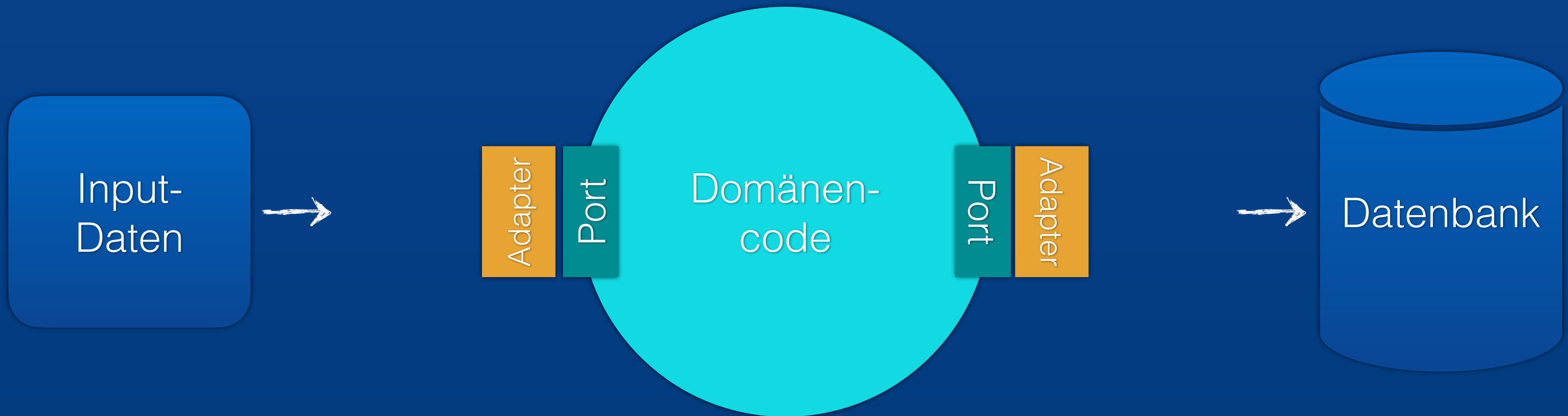
Schnelle (Unit-)Tests für ganze Anwendungsfälle



Architekturmuster

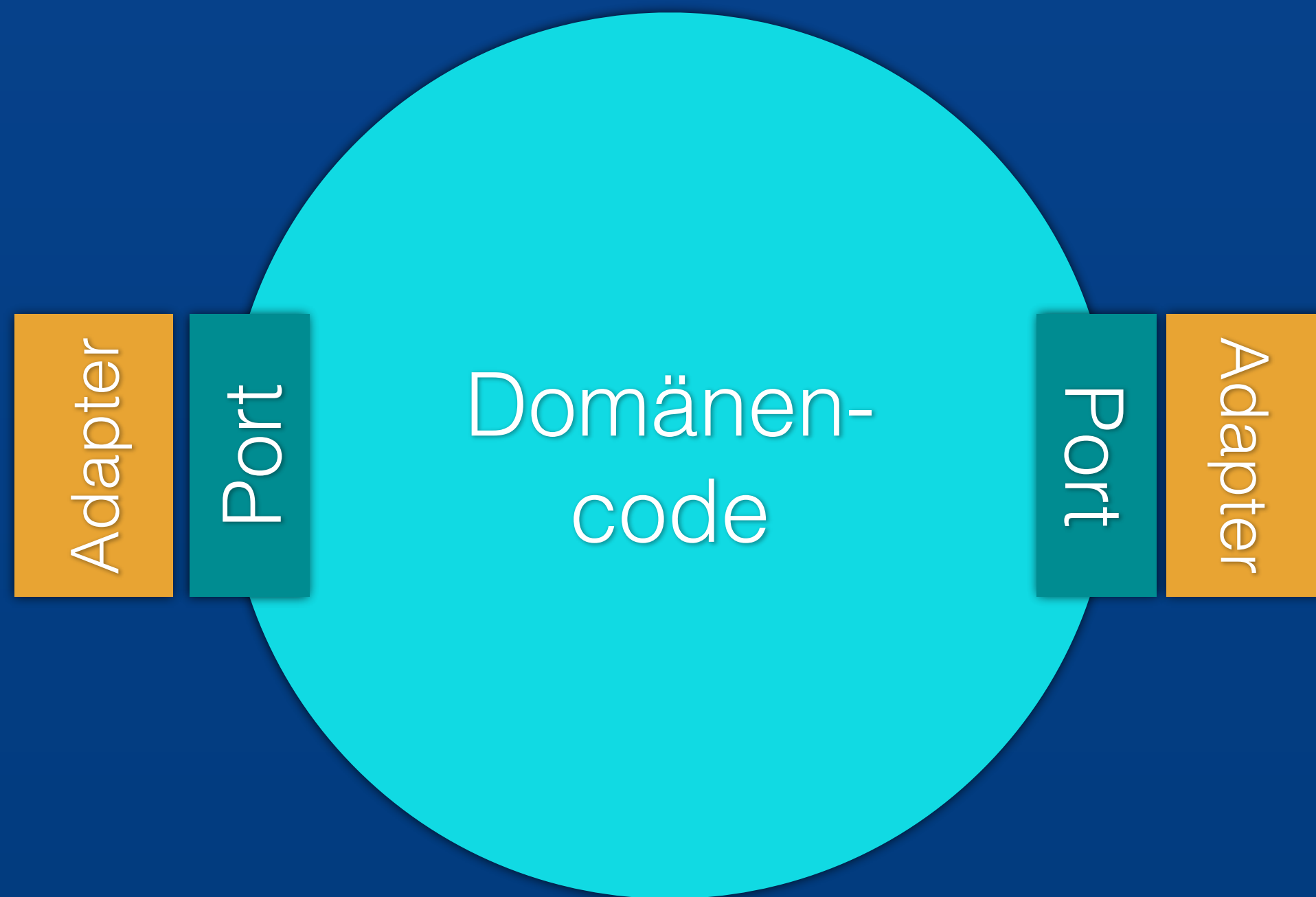


Architekturstile



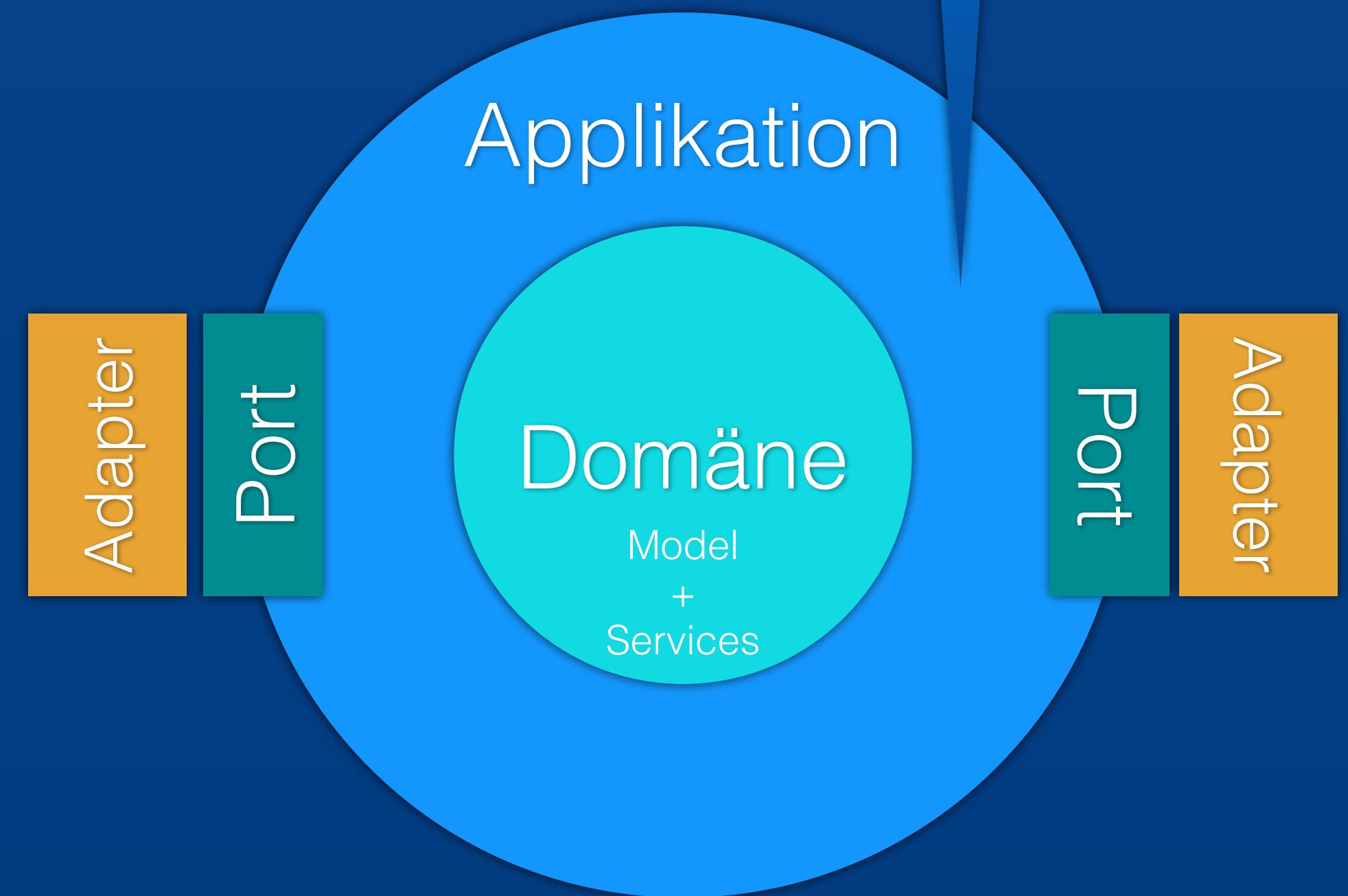
Architekturstile

Ports & Adapters, „Hexagonal“



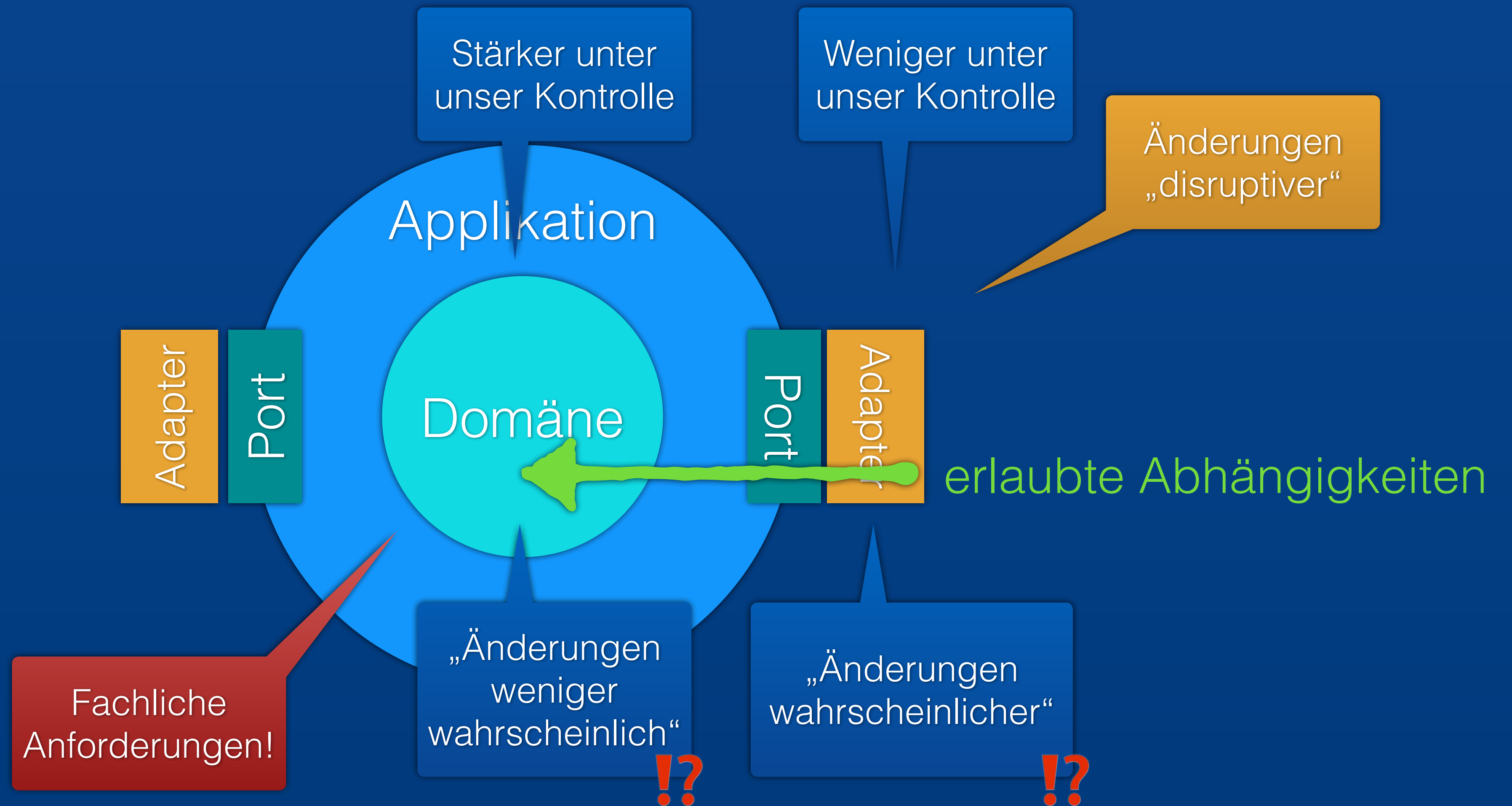
„Onion“

Use-Cases

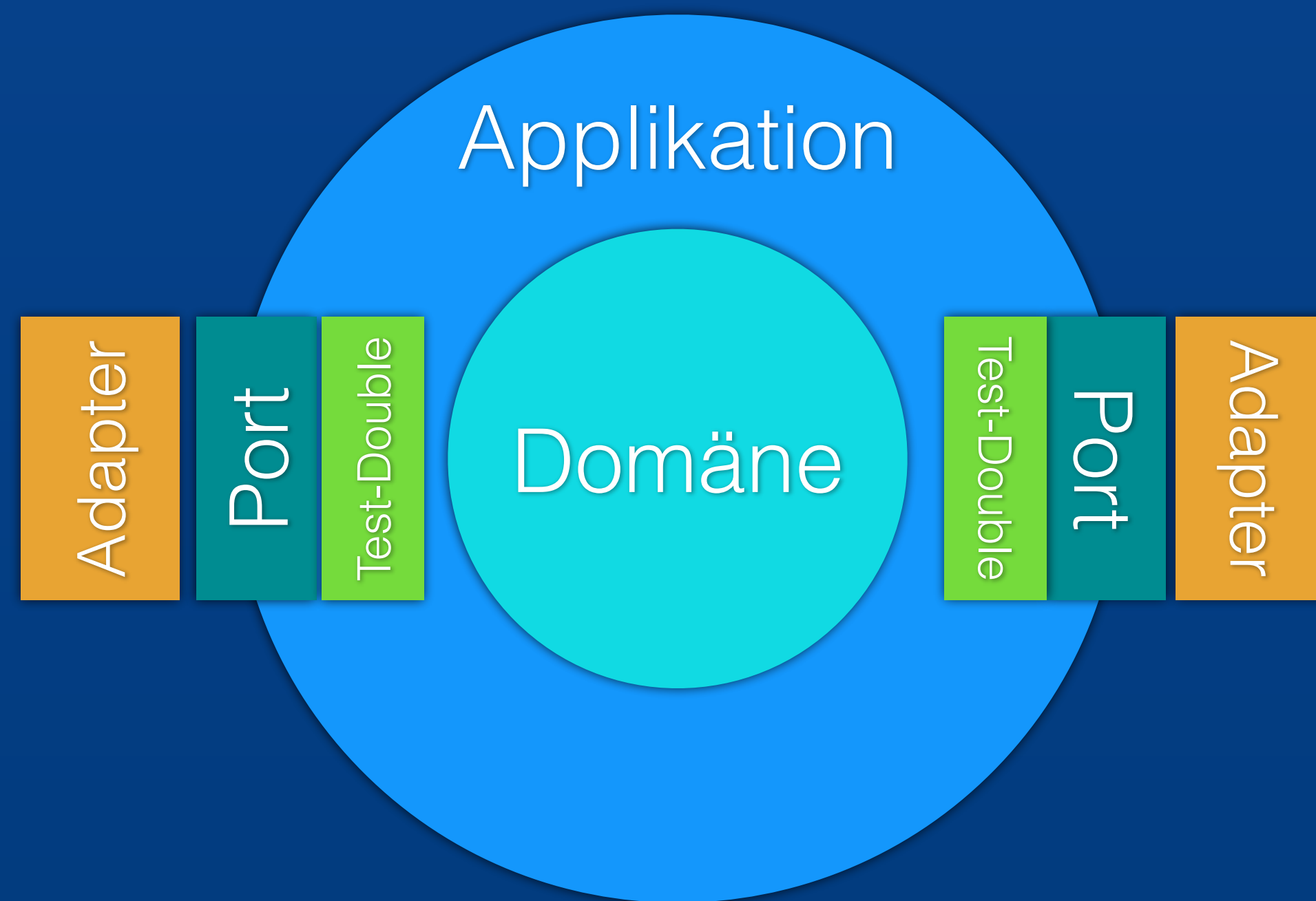


„Clean“ (irgendwo dazwischen)

Statt Schichten: Innen & außen!



Schnelle Tests für ganze Use-Cases



Schnelle (Unit-)Tests
– nicht nur Mikrotests



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);  
}
```



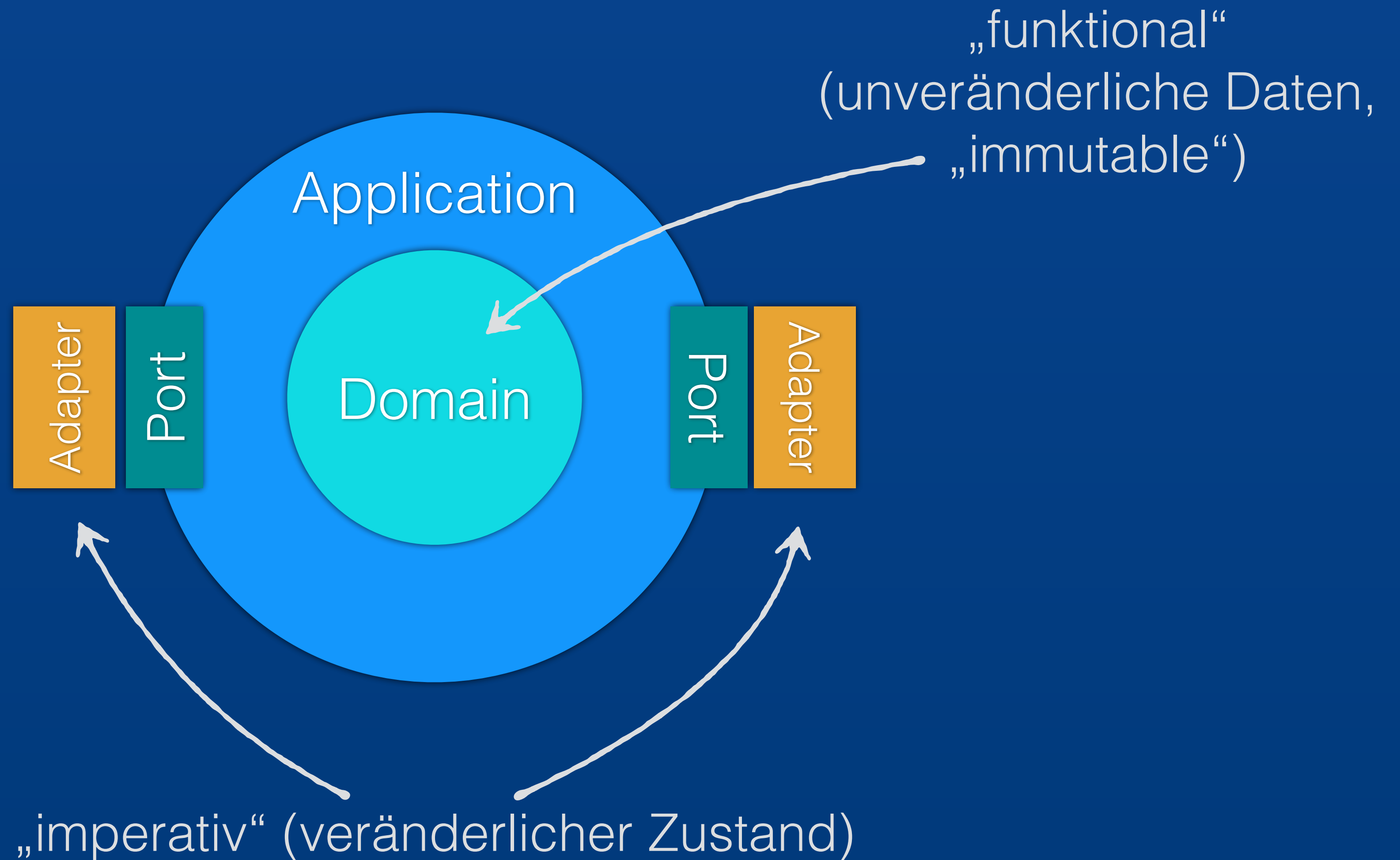
<https://github.com/thmuch/einfach-gut-testbar>

Fazit & Ausblick

Bessere Testbarkeit

- ... 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



engl.
„Functional Core,
Imperative Shell“

Weiterdenken – Architekturarbeit!

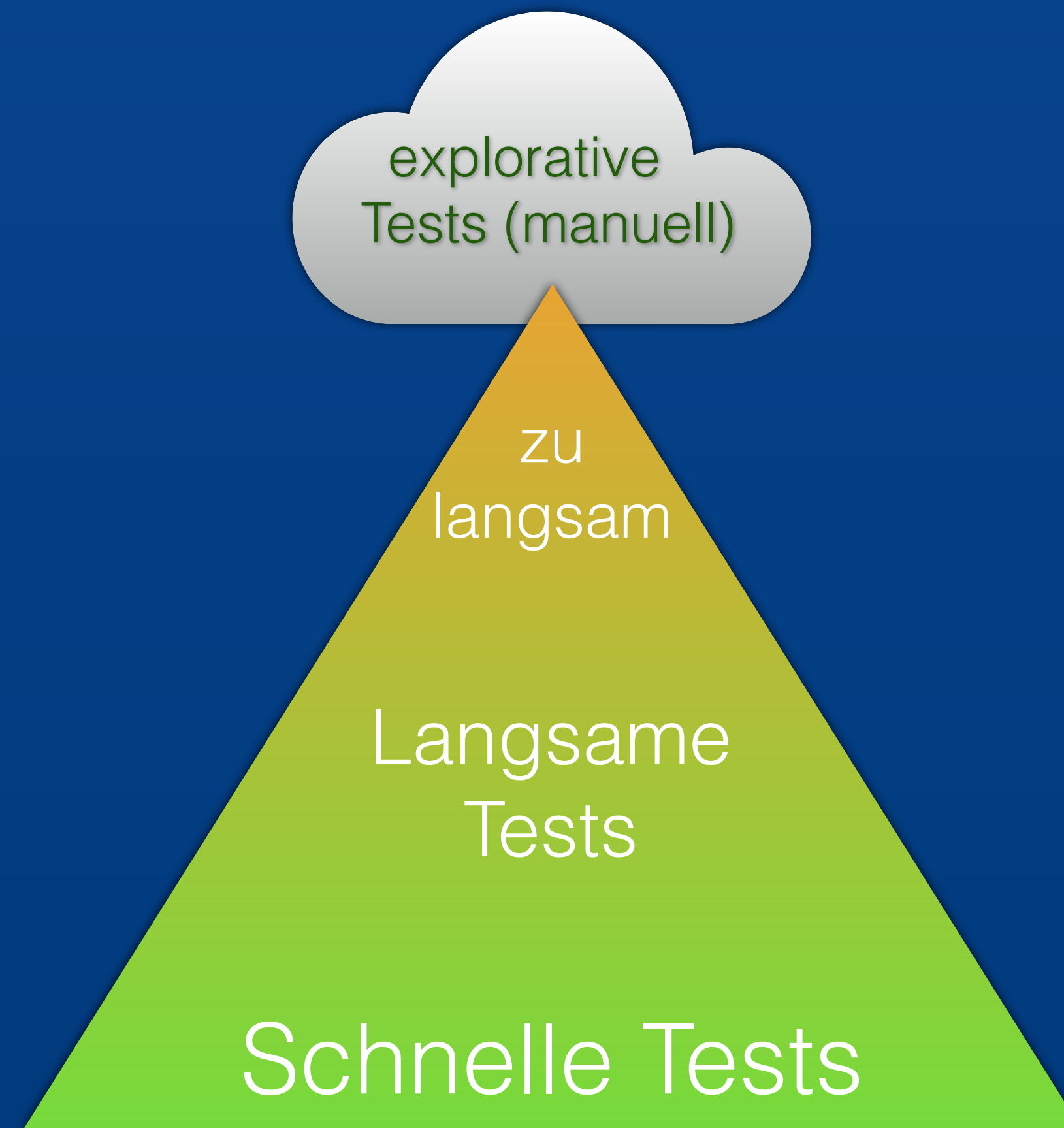
Lohnen sich die Strukturen und Abstraktionen?

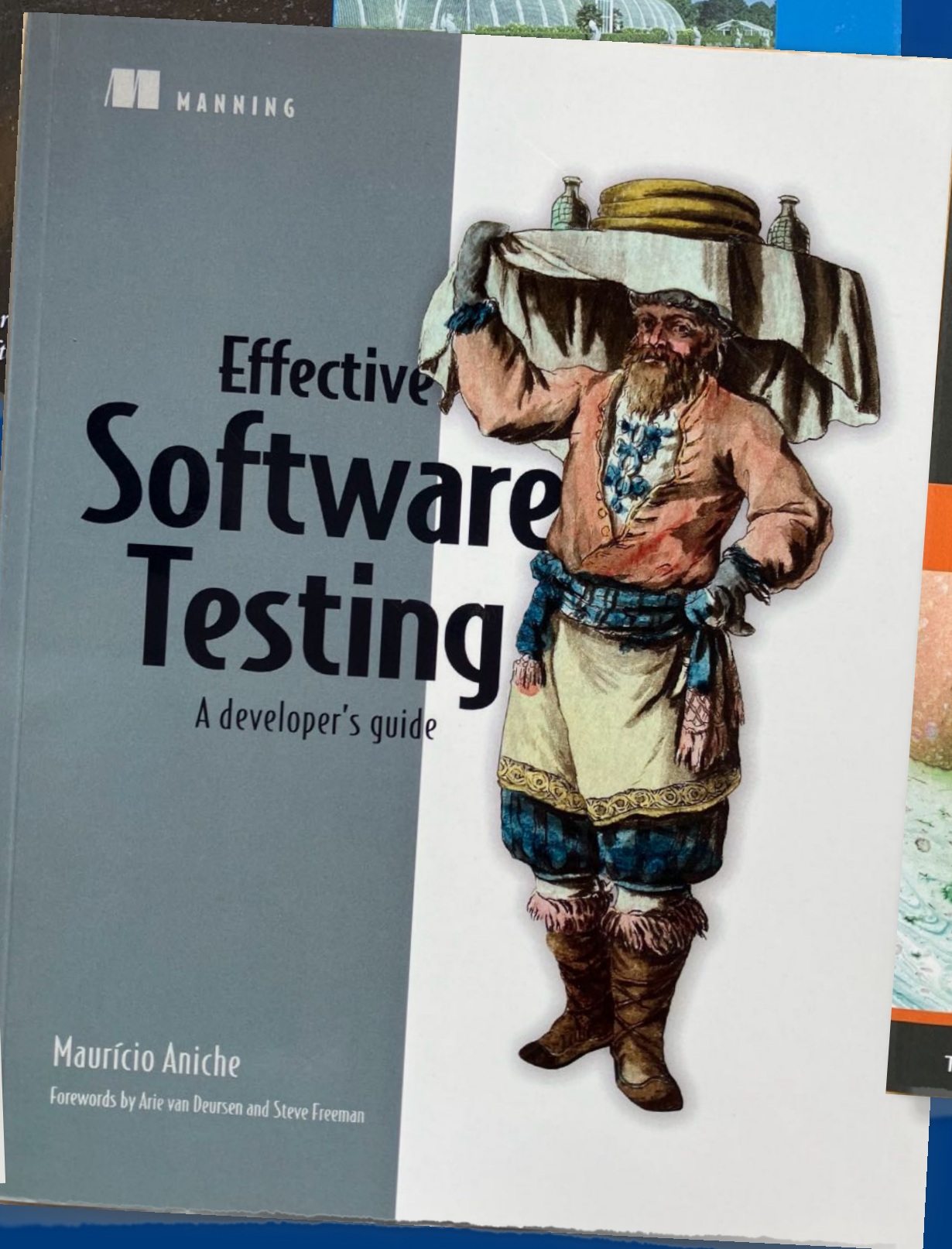
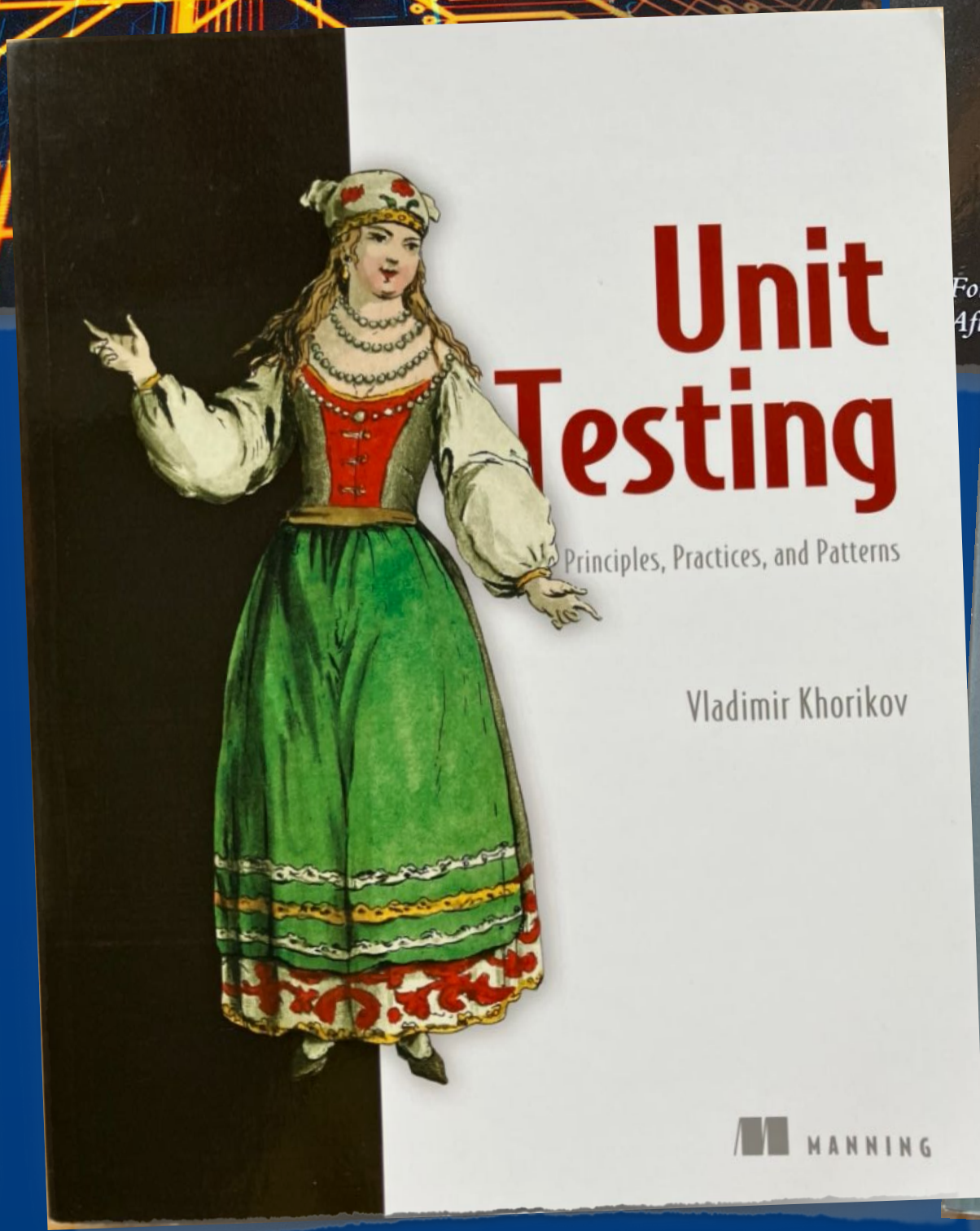
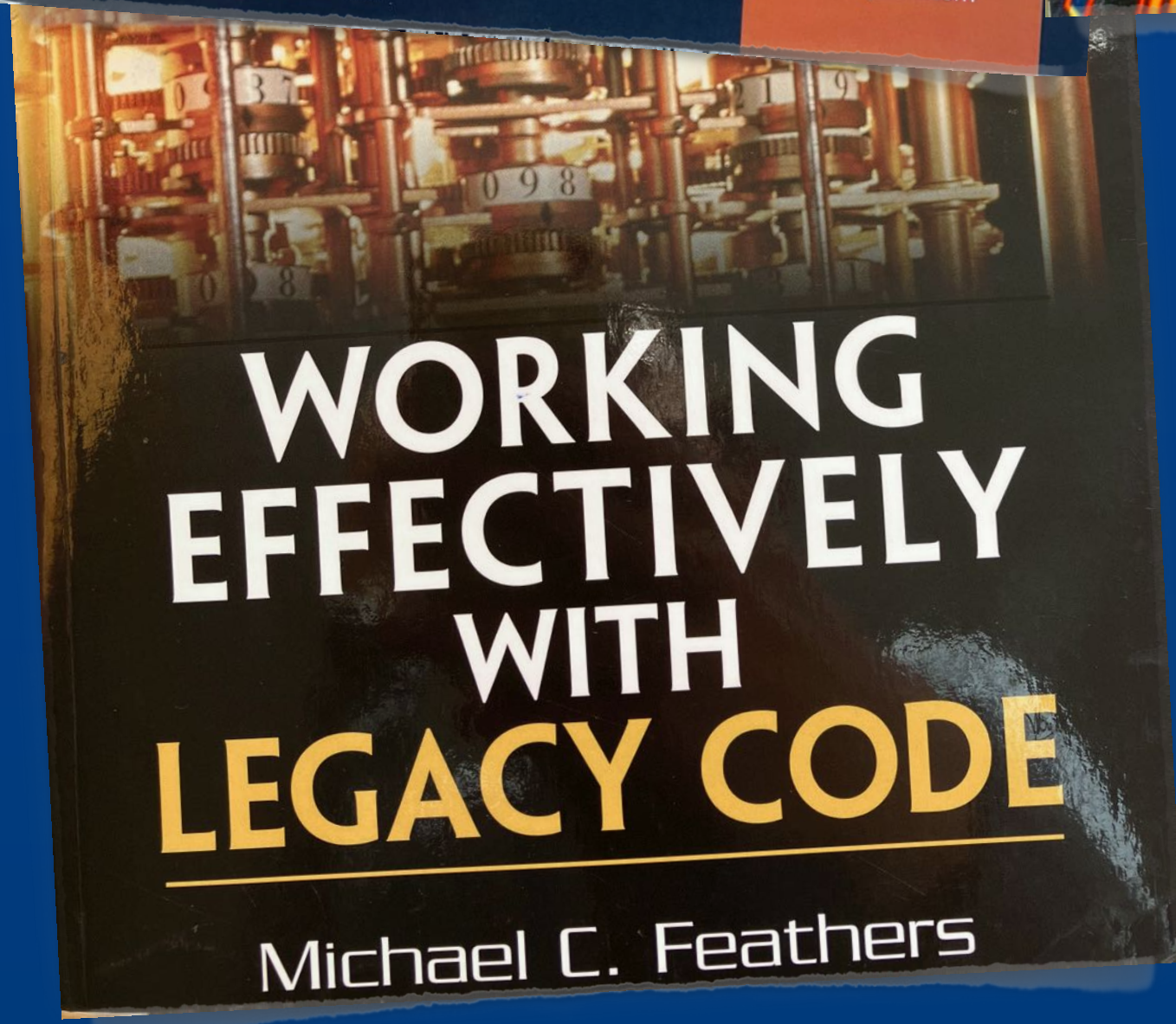
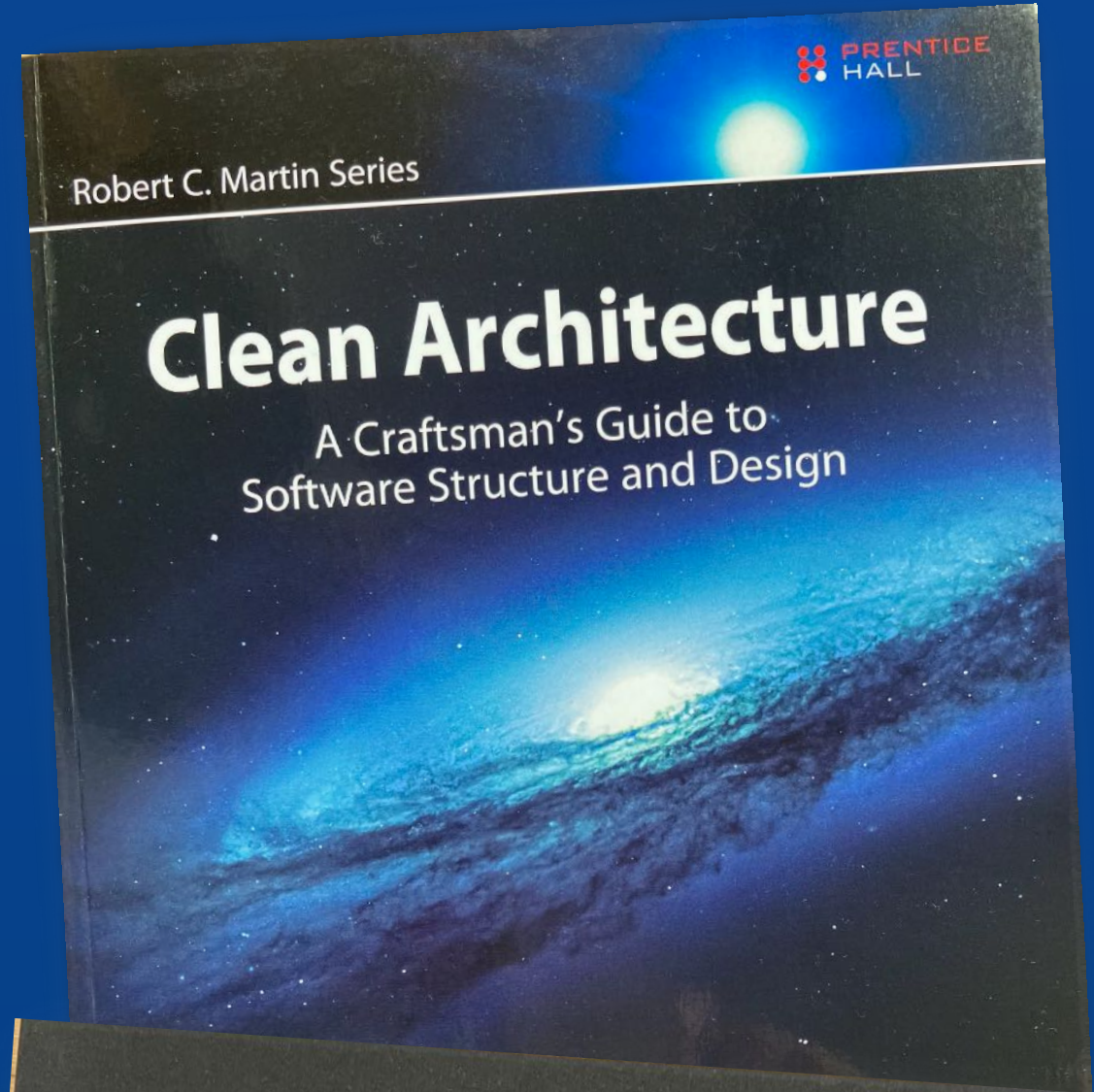
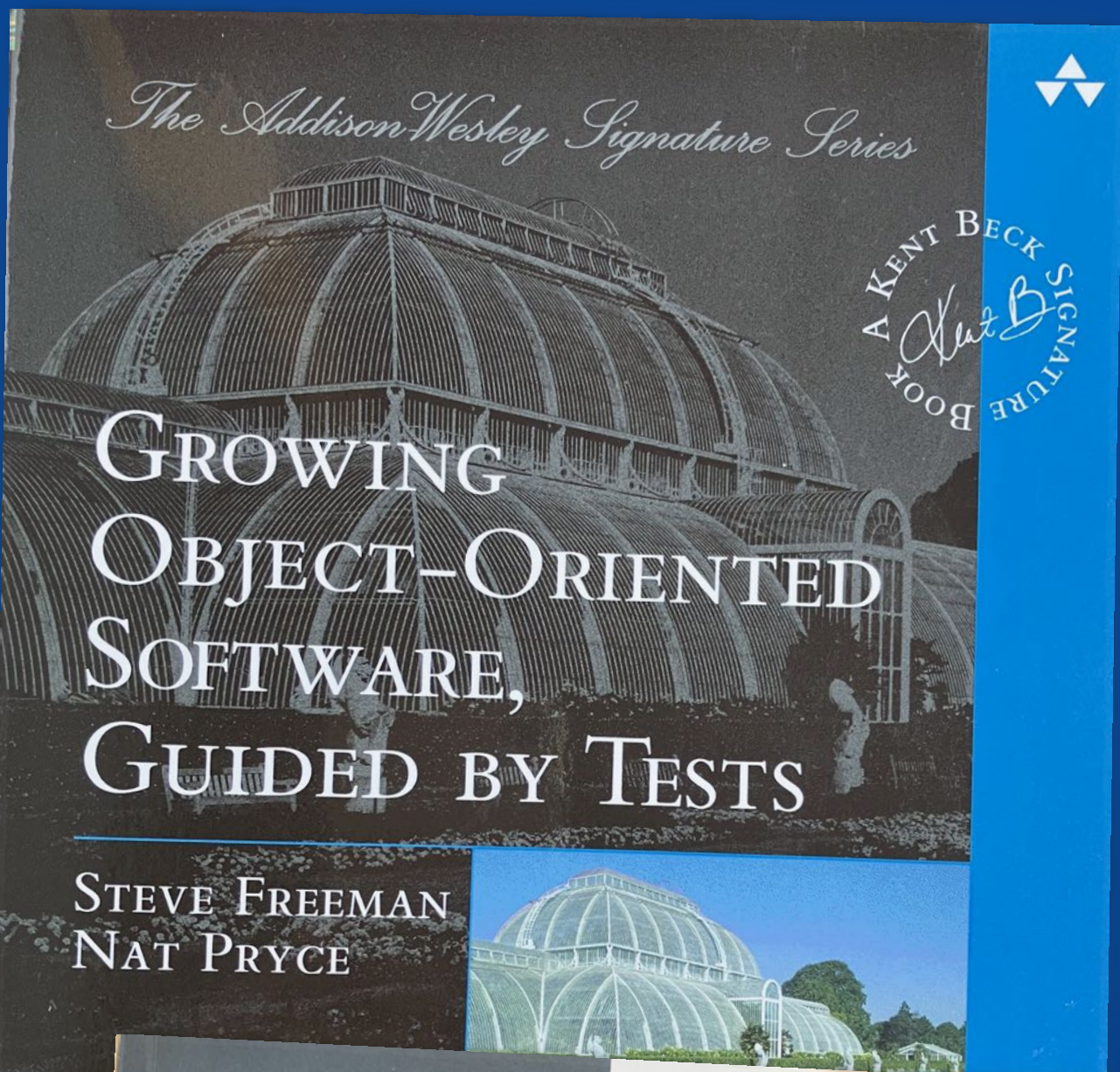
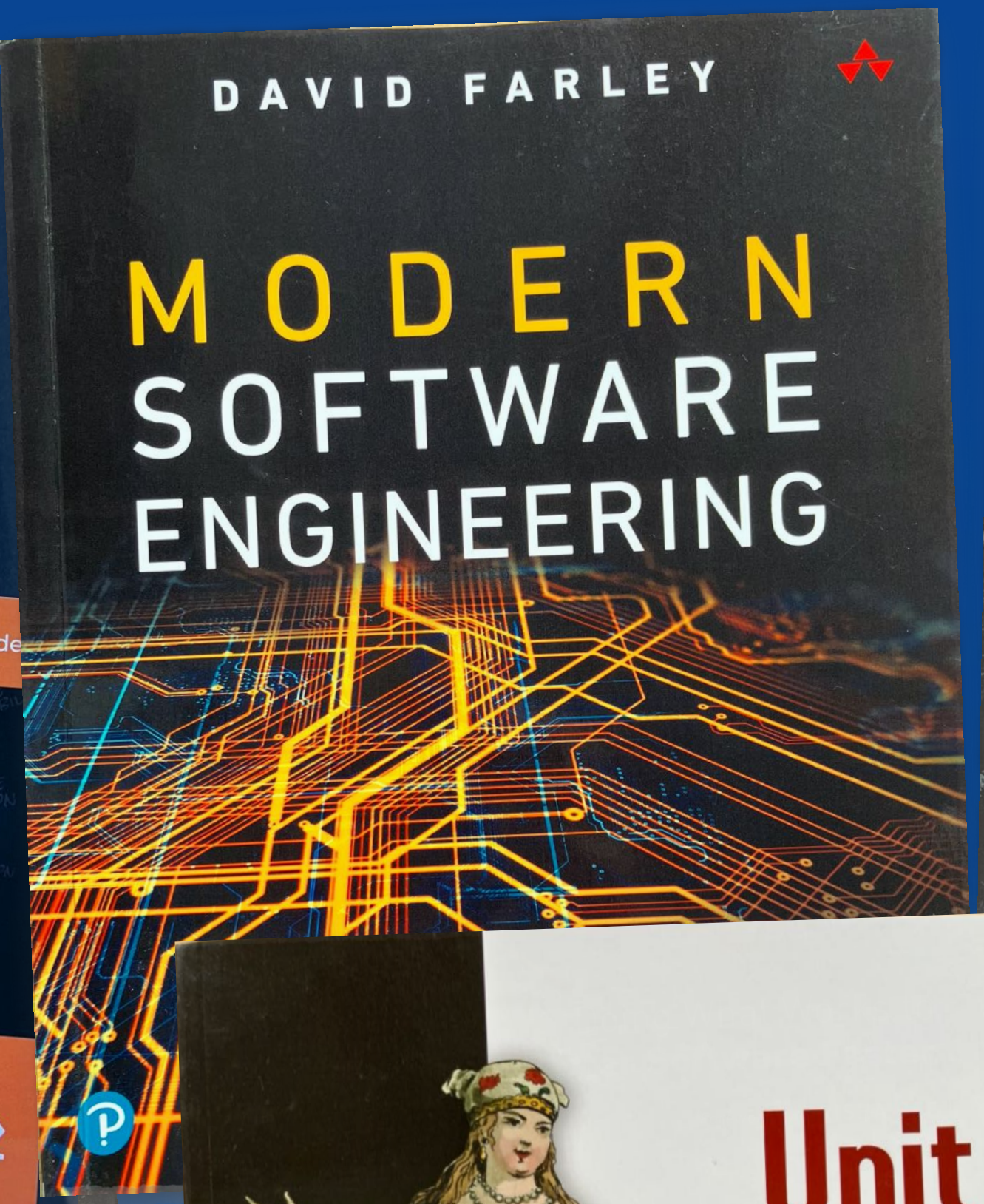
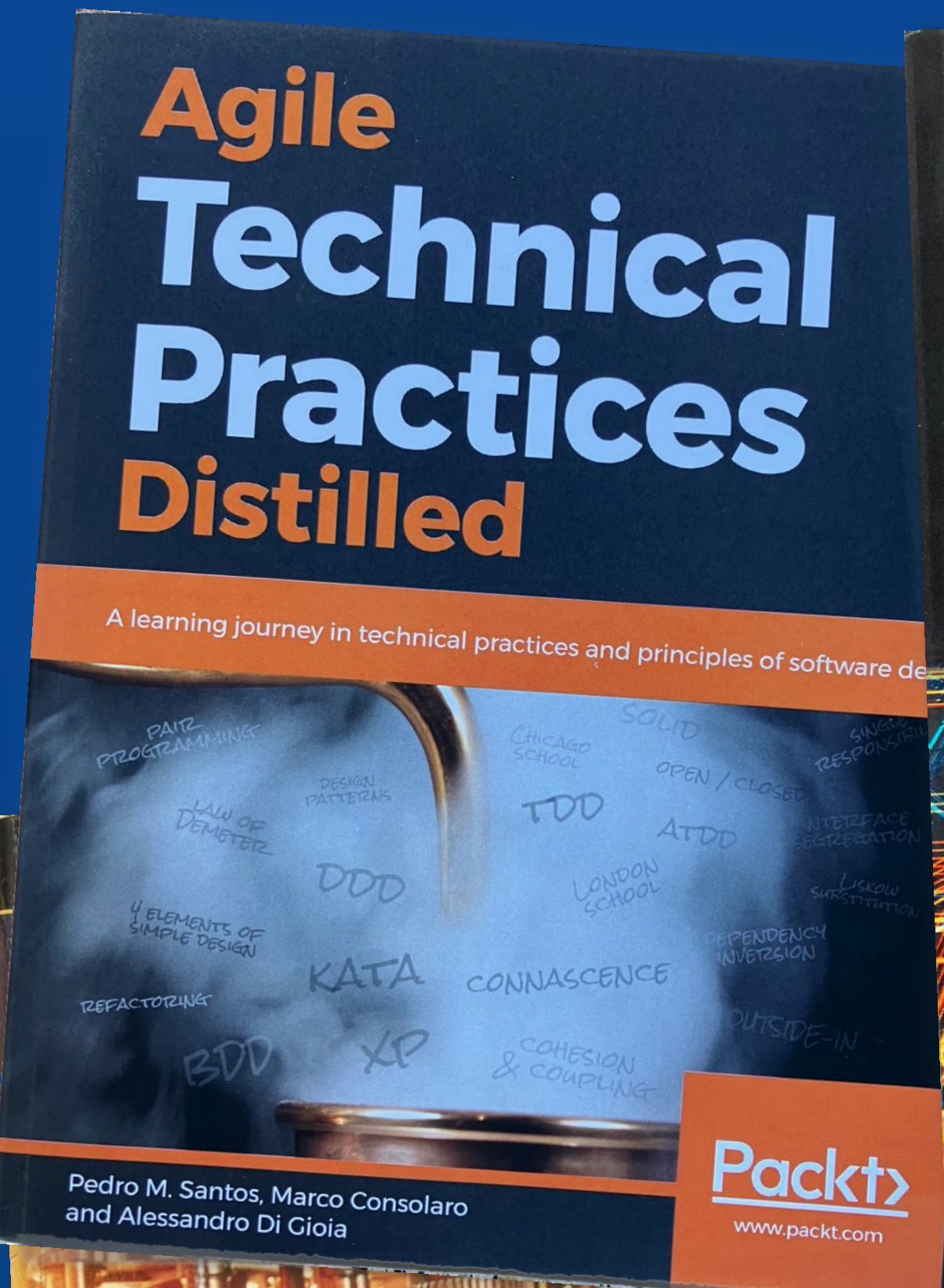
Ist **schnell zu Testendes** anders von
langsam zu Testendem unterscheidbar?

Ziel:

Umfassende, schnelle, stabile Tests

Testgeschwindigkeitspyramide?!







Testbarkeit

Unit-Tests

Schnelle Tests

Abhängigkeiten

Architektur

Fragen?

Mikrotests

Code-Design

Größe einer Unit

Fast Feedback Loops

  @thmuch



Vielen Dank 😊



www.tk.de/IT

  @thmuch

"Do not depend on volatile things"
(Robert C. Martin)

**"Make the change easy (this can be hard!),
then make the easy change"**
(Kent Beck)

"Many More Much Smaller Steps"
(GeePaw Hill)