

Microsoft  
Learn

STUDENT AMBASSADOR



# Test-Driven Development

Gino Messmer



# Agenda

- Vorstellung
- Manuelle Tests & Software-Tests
- Arten von Tests
- Mocks, Fakes & Stubs
- TDD und Herausforderungen
- Demo: TDD in der Praxis
- Q&A
- Nützliche Ressourcen



# Gino Messmer

Microsoft Learn Student Ambassador  
DHBW Karlsruhe  
medialesson GmbH

@ginomessmer  
<https://linktr.ee/ginomessmer>



# Wie testet ihr eure Projekte?

Abstimmen auf <https://sli.do/tdd>



- 3/3 ✓ Pass
- 4/5 ⚠ Partial
- 2/8 🔥 Fail
- 3/3 ✓ Pass
- 3/3 😄 Pass

# Warum testen?

- Fehler minimieren
- Sicherheitslücken und Risiken bekämpfen
- Unterstützt Dokumentation
- Oft gesetzliche/vertragliche Voraussetzung





[https://de.wikipedia.org/wiki/Starship\\_\(Rakete\)](https://de.wikipedia.org/wiki/Starship_(Rakete))



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)



# Manuelles Testen

- Teil der Quality Assurance
- Menschen testen
- Basierend auf Testplänen
- Hoher Aufwand (Erstellung + Updates)
- Zeitaufwendig und Teuer
- Fehleranfälliger

# Test Plans

The screenshot displays the Azure DevOps Test Plans interface for the 'Dane BABOK Project'. The left sidebar contains navigation links: Overview, Boards, Repos, Pipelines, Test Plans (selected), Parameters, Configurations, Runs, and Artifacts. The main area shows the 'Test Plan for Iteration 1' with a tree view containing three test cases: '306 : Add user (1)', '307 : Edit user (3)' (selected), and '308 : Delete user'. The right pane shows the 'Test suite: 307 : Edit user (Suite ID: 2257)' with a table of tests. A context menu is open over the third test, listing actions such as 'Assign tester', 'Run for web application', and 'Pass test'.

Outcome	Order	ID	Title	Configurat...	Tester
Active	1	1342	Test foor for sample FR	Windows ...	vstsadmin
Active	2	1578	TC new	Windows ...	vstsadmin
Active	3	2213		Windows ...	vstsadmin

<https://43dj4uavp3670rku1dte2310-wpengine.netdna-ssl.com/wp-content/uploads/2019/12/Test-Runs.png>



# Was sind Software-Tests?

Code, die die Anforderungen beschreiben und den Production Code validieren.

Validieren: *Gültigkeit von etwas bestimmen*  
**das heißt:** Entspricht X den Anforderungen?



# Testautomatisierung

- Write once
- Unbeaufsichtigte Ausführung
- Schnellere Ausführung
- Günstiger
- Teil von Continuous Integration
- Test Coverage



# Was macht einen guten Test aus?

## Erfüllen der **A-TRIP** Eigenschaften

**A**utomatic

**T**horough

**R**epeatable

**I**ndependent

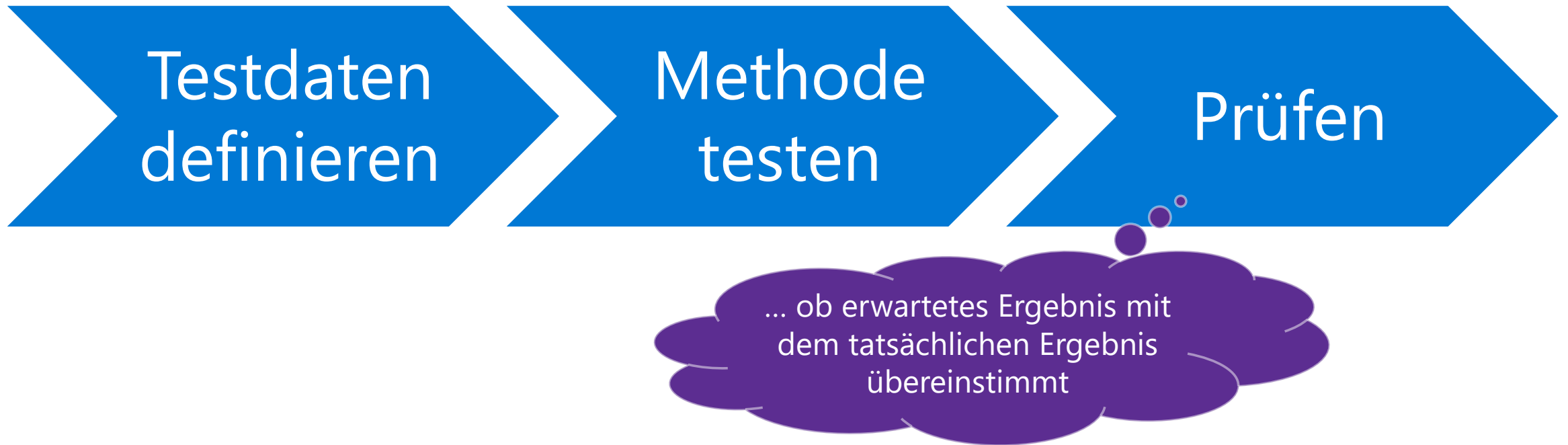
**P**rofessional

# Herangehensweise



Tipp:  
xUnit

# Struktur

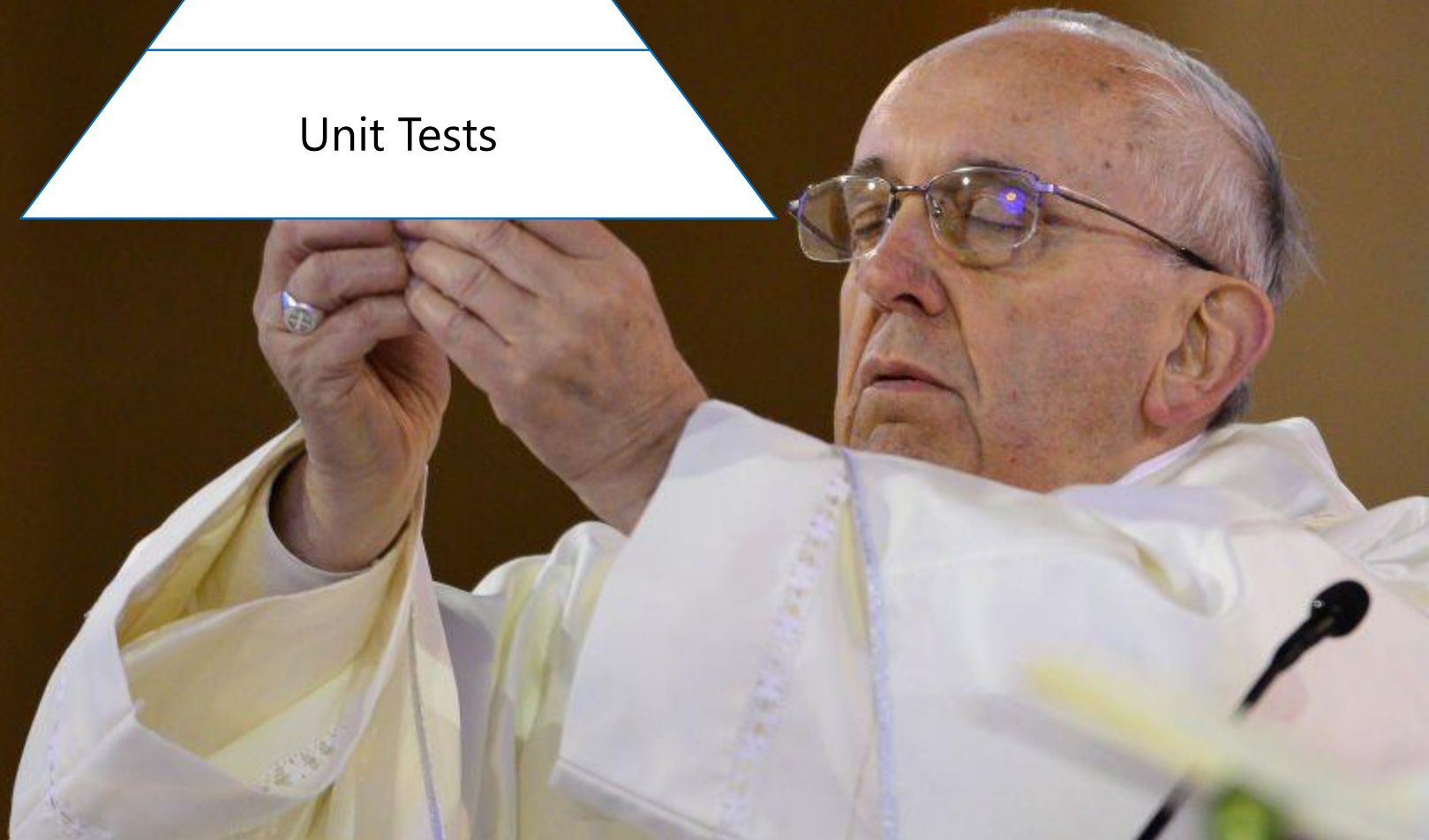




UI  
Tests

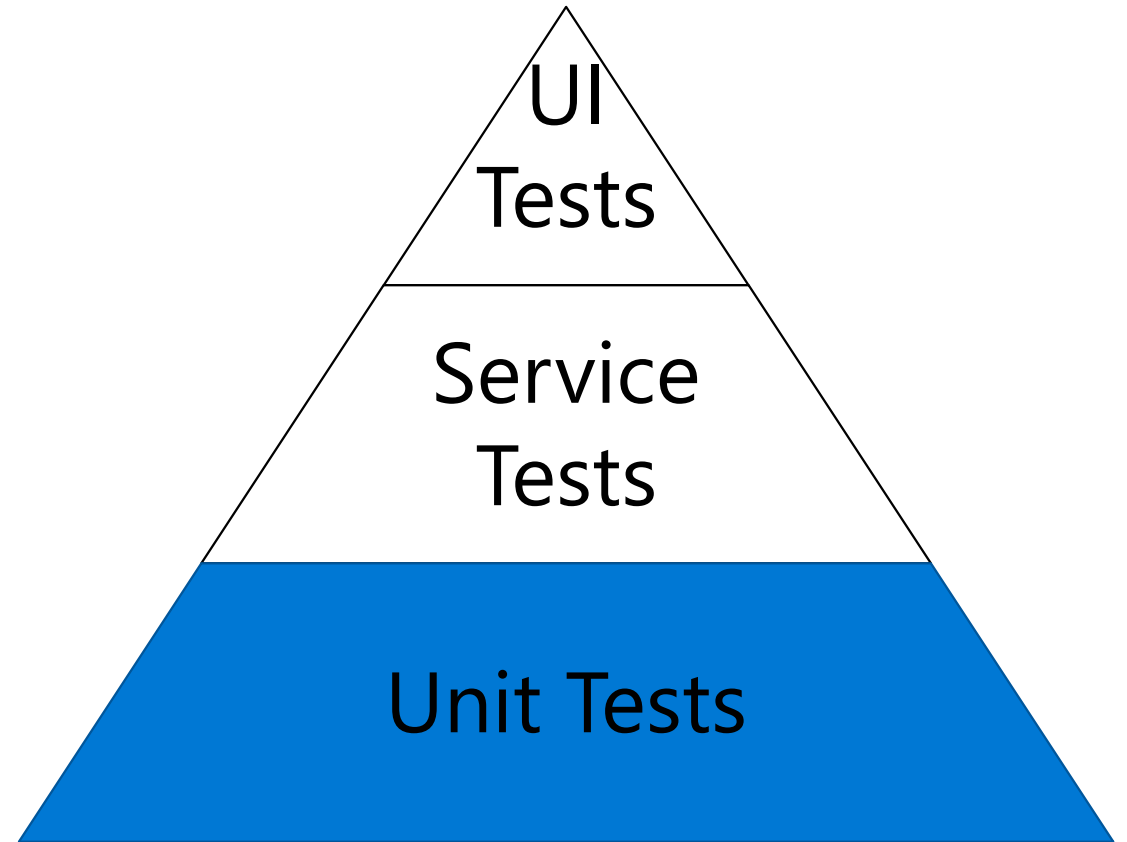
Service Tests

Unit Tests



# Unit Tests

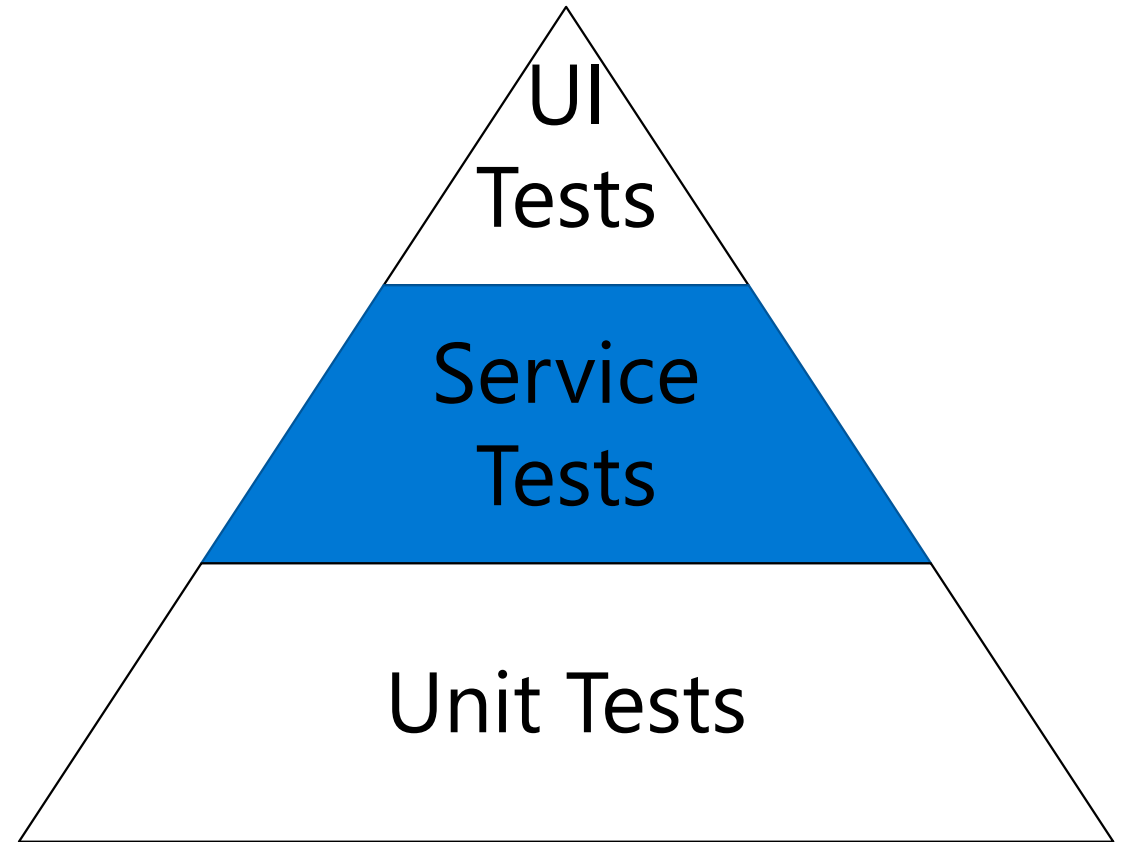
- Software besteht aus mehreren Einheiten (Units)
- Testen die kleinstmögliche Unit
- Was ist eine Unit? It depends:
  - Methode
  - Klasse
- Benötigt ggf. Mocking und Stubbing (dazu später mehr)



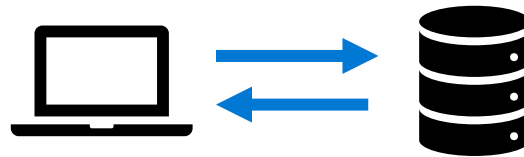
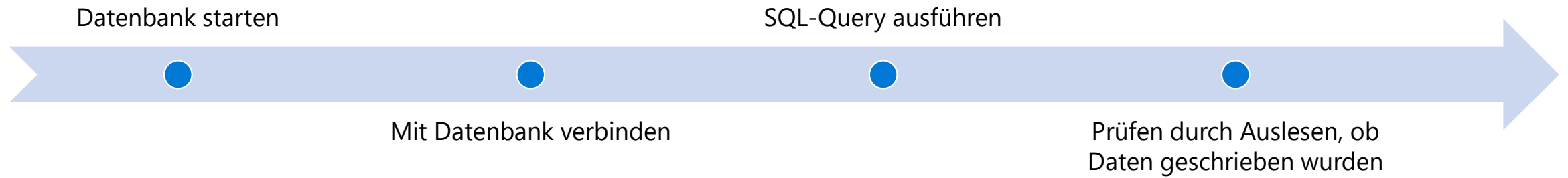


# Service Tests

- u.a. Integration Tests, Contract Tests
- Testen der *Integration* zwischen zwei (fremden) Komponenten/Systemen
  - Bspw. Datenbanken, Netzwerke, Dateisysteme
- In Unit Tests ausgeschlossen → Isolation



# Integration Test



# Integration Tests – Wo?

Datenbanken

Web APIs

Externe  
Programme

Hardware

IoT /  
Sensoren

Und mehr...

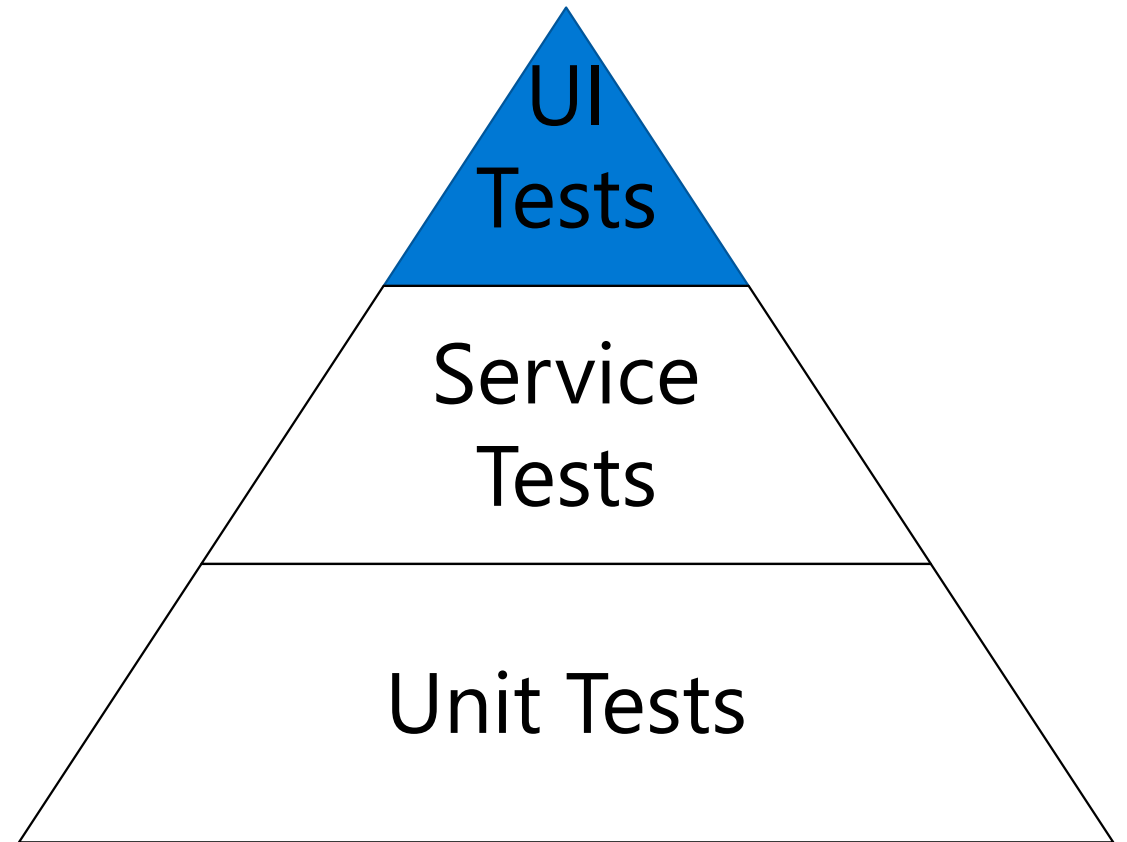


**2 UNIT TESTS,**

**0 INTEGRATION TESTS**

# UI Tests

- Auch bekannt als End-to-End (E2E) Tests
- Typisch bei manuellen Tests
- Verhalten, Layout, Usability, Accessibility, ...
- Hohe Parametrisierung notwendig
- ggf. Screenshots (Browser-Tests, etc.)

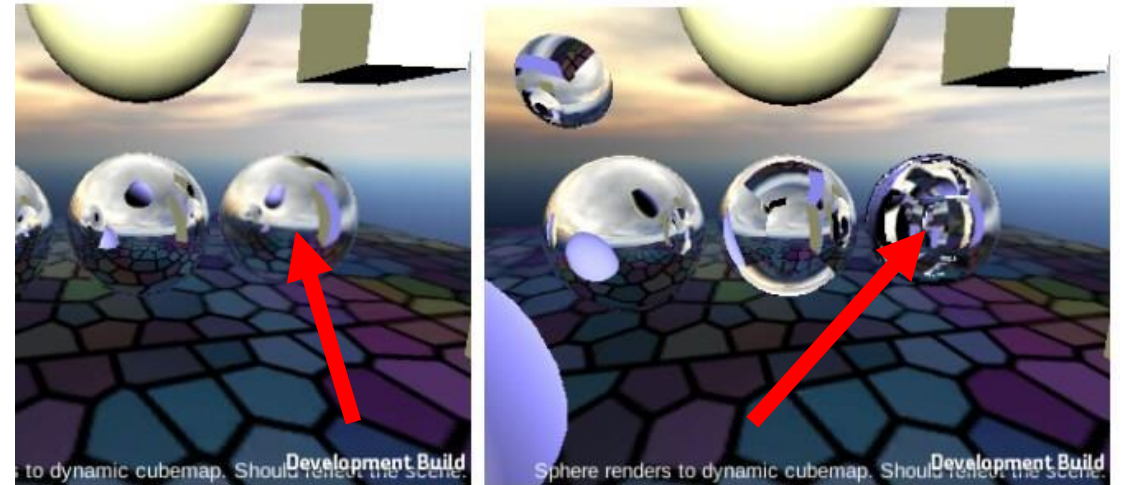


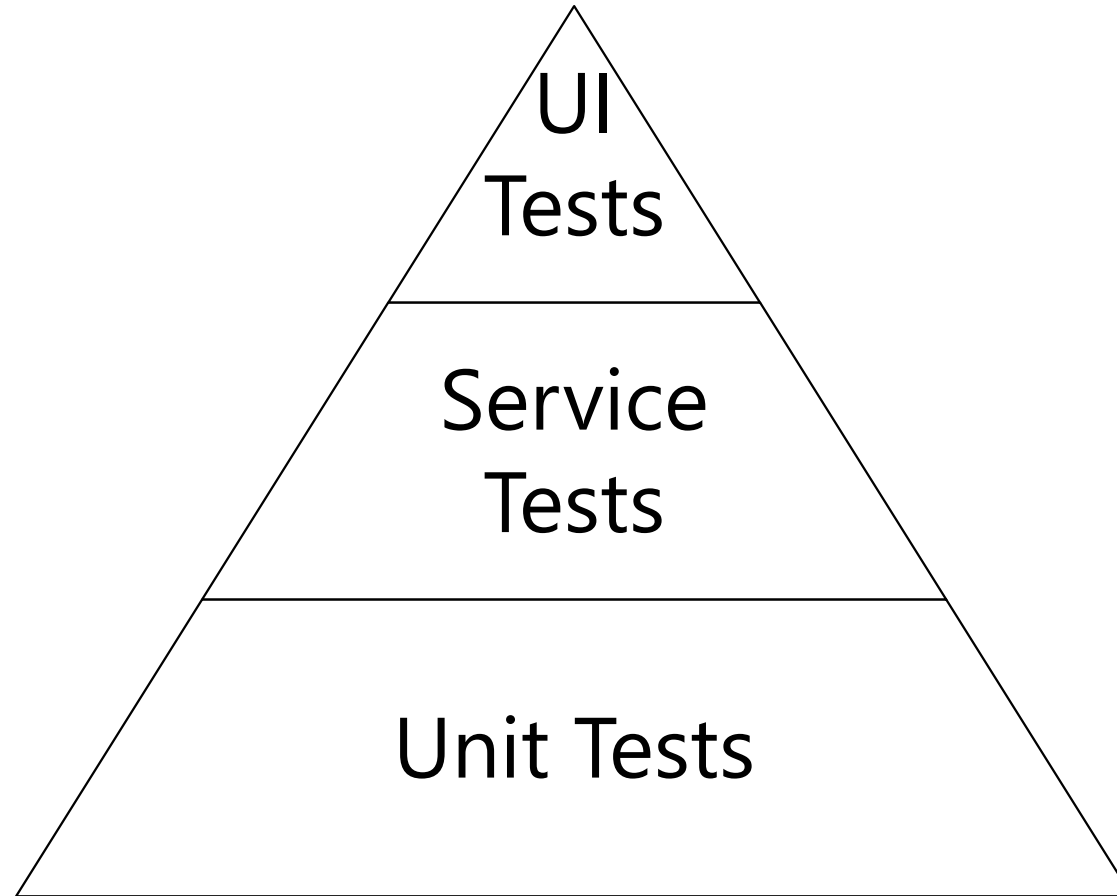


# UI Tests – Graphics Tests

The resulting image is then compared with a previously approved reference image for that scene/graphics settings and device. Should any of the resulting images differ from the reference images, we then flag the test as failed and someone needs to manually verify if the failure is the result of some intentional or unintentional change that needs to get fixed.

<https://blog.unity.com/technology/graphics-tests-the-last-line-of-automated-testing>





# Weitere Testarten

---

## Functional Tests

Acceptance Tests

---

Contract Tests

---

Sanity Tests

---

## Performance Tests

Stress Tests

---

Load Tests

---

Many more...

Usability Tests

---



# Fake it until you make it

Mocks, Stubs, Fakes



# Test Doubles - Mocks, Stubs, Fakes

## Mocks

- Vorprogrammierte Objekte mit erwarteten Rückgabewerten
- Agieren als Stellvertreter
- *Spezifisch* für den Test

## Stubs

- Liefern *explizite* Antworten für Methoden zurück
- Sehr leichtgewichtig
- Ahmen Interface oder abstrakte Klasse nach

## Fakes

- Funktionierende Implementierung
- ... die *keinen Nutzen in Production* hat
- Bspw. Fake Web Service



# Development Pattern

Inversion of  
Control

Separation  
of Concerns

SOLID

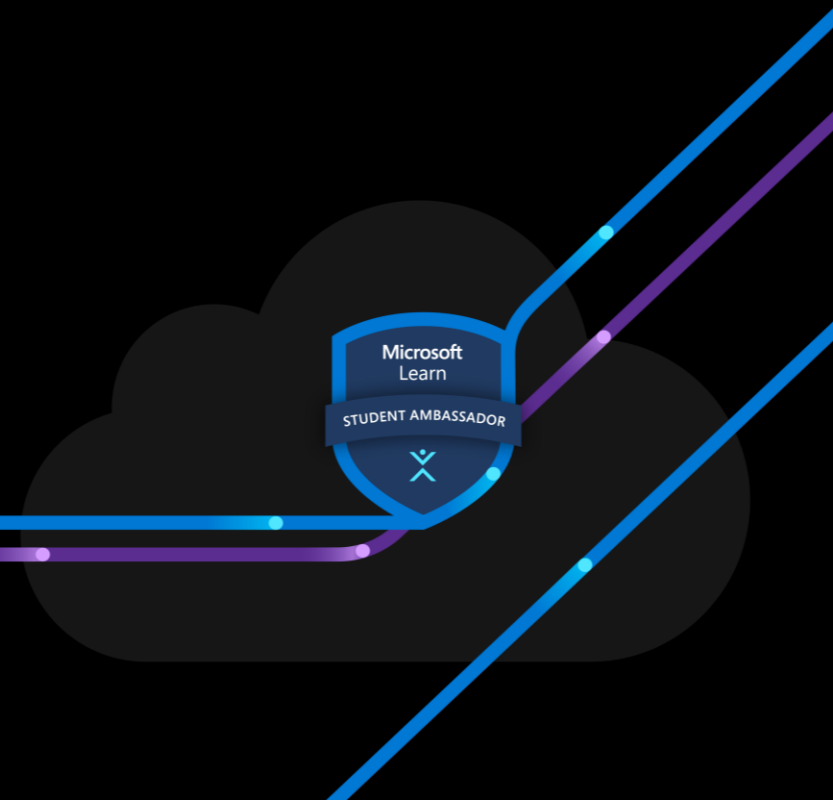
<https://refactoring.guru/>

DRY

KISS

...

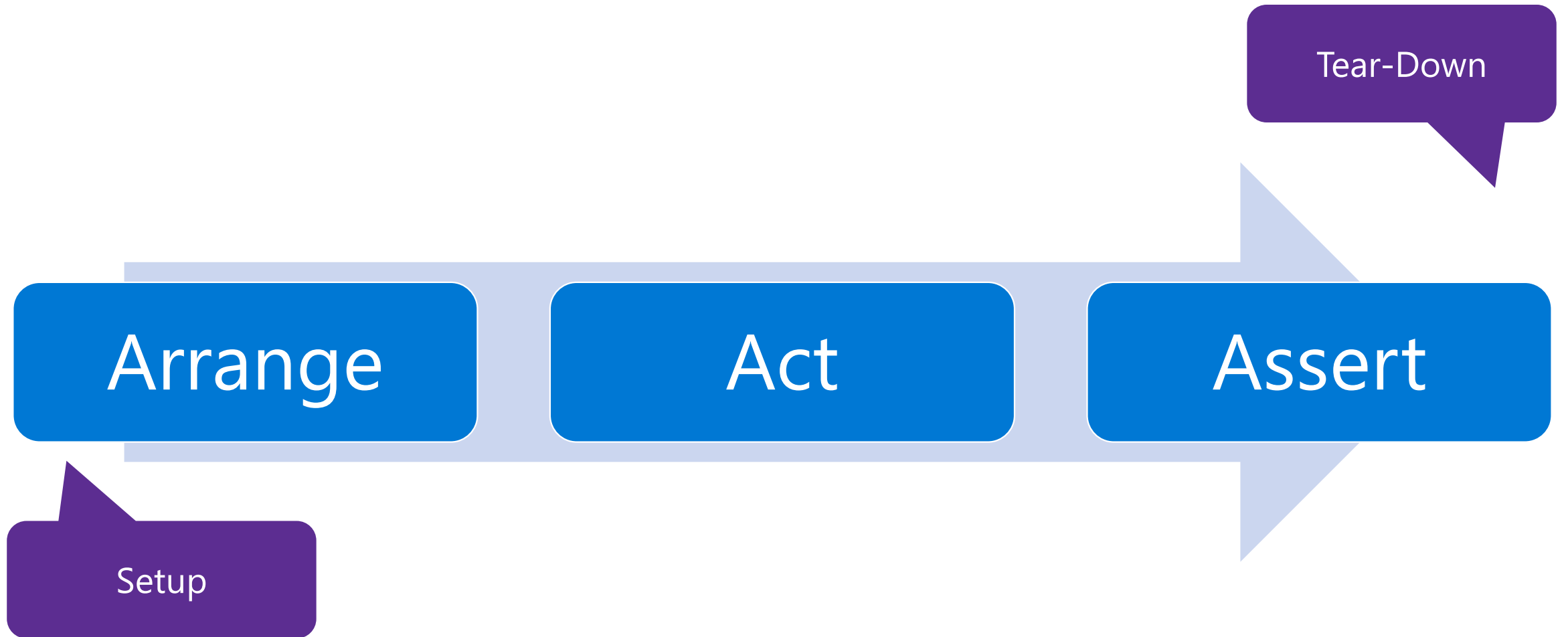
# Test-Driven Development



# Test-Aufbau



# Test-Aufbau



# Assert

- Equal
- Contains
- Same
- True/False
- Throws Exception
- Empty/Null/NotEmpty/NotNull

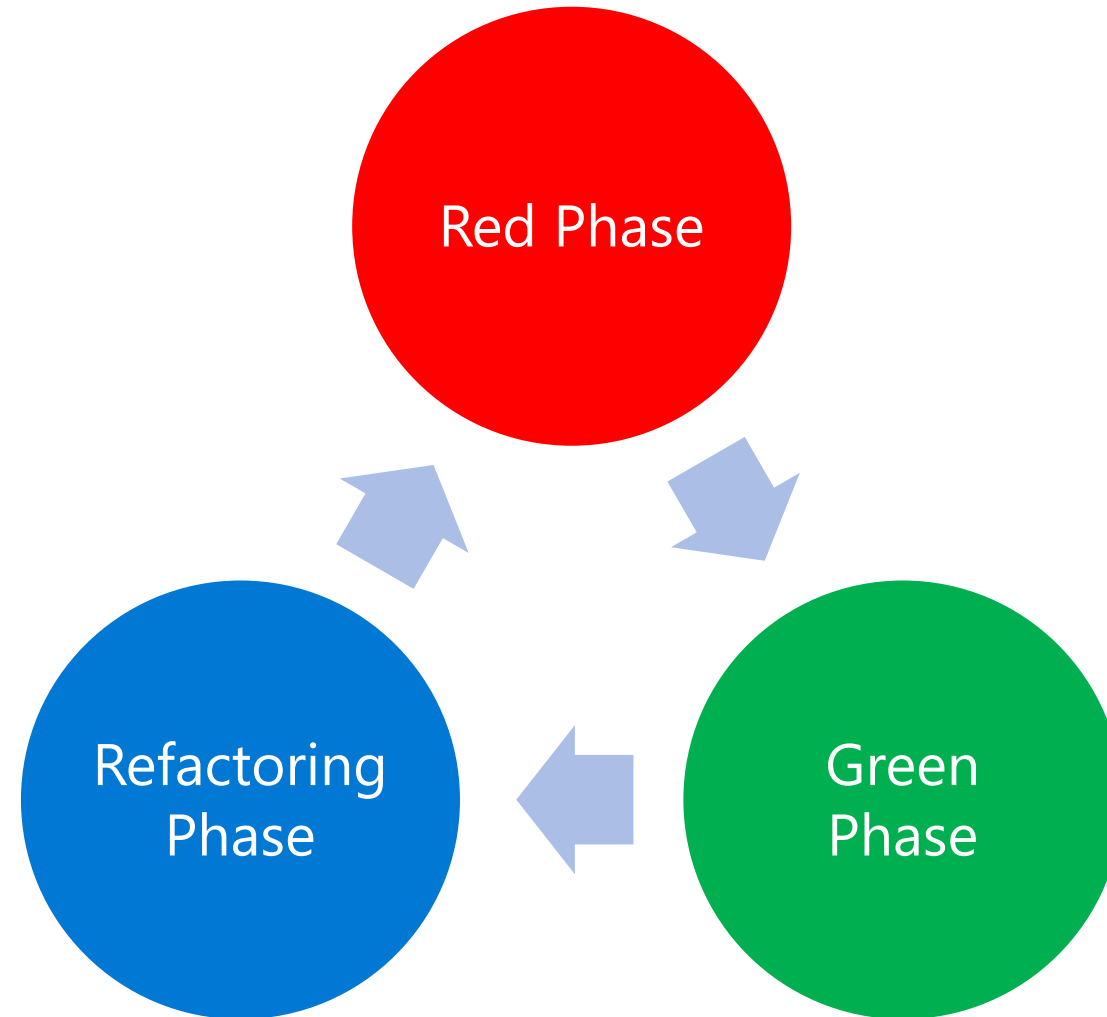
# Arrange, Act, Assert

```
[Fact]
public void Test()
{
    // Arrange
    var calculator = new Calculator();

    // Act
    var result = calculator.Add(1, 2);

    // Assert
    Assert.Equal(3, result);
}
```

# TDD Phasen



# Red Phase

- Mit Test beginnen
- Verhalten im Test definieren
- Ohne Implementierung
- Stattdessen Design-Phase
- Test schreiben um Production Code zu schreiben
- Nicht Test schreiben um Production Code zu testen



# Green Phase

- Hier kommt der Production Code
- Minimalste Implementierung um Test zu bestehen
- Einfache Lösung → Weniger Fehleranfällig
- Clean Code? Performance? Eins nach dem anderen...

# Refactoring Phase

- Production Code verbessern
- Einfachen Code in Clean Code umschreiben

# TDD destilliert

- Fokus auf Anforderungen bevor Code geschrieben wird
- Refactoring erleichtert
- Fear of Change eliminiert
- Wertvoll für die Qualität
- Testen wird zur Pflicht

TDD ist eine Disziplin



**JUST DO IT.**

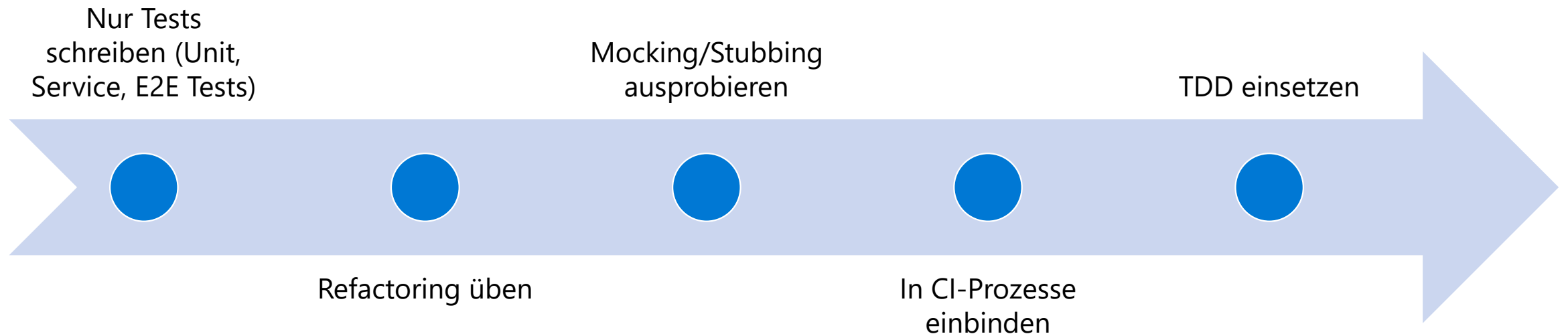
# Warum scheitern wir trotzdem?

- Ungewohnt
- Zweifel im Unternehmen (Kosten, Zeit)
- Für große Projekte eine komplexe Erscheinung
- Manche Tests zu einfach
- Schnelles Aufgeben
- Wartungsaufwand (ist der Test oder Code kaputt?)
- Fehlende Disziplin um Tests frühzeitig zu schreiben
- Sich ändernde Anforderungen

# TDD ist eine Disziplin

- Anforderungen festlegen
- Tools festlegen
- Tools verstehen und richtig anwenden
- Ggf. manifestieren
- Es ist wie Fahrradfahren...
- Ihr benötigt so oder so Tests. Warum nicht gleich TDD?
- Codequalität wird zu Beginn höher als sonst

# Roadmap to TDD



# Hands-On

<https://github.com/ginomessmer/voxium>







Euer Feedback  
(max. 5 Minuten)


# Vielen Dank – Fragen?

<https://linkedin.com/in/ginomessmer>

<https://linktr.ee/ginomessmer>



# Nützliche Ressourcen

- [GitHub Student Developer Pack](#)
    - JetBrains
    - Domain Names
    - Vieles mehr...
  - [Figma](#)
  - [Azure for Students – Free Account Credit](#)
  - [Jetbrain Free Educational Licenses](#)
  - [Software, Technology & Design Meetup Pforzheim | Meetup](#)
- 

# Referenzen

- [The Practical Test Pyramid \(martinfowler.com\)](http://martinfowler.com/practical/testpyramid/index.html)
- [UnitTest \(martinfowler.com\)](http://martinfowler.com/UnitTesting.html)
- [Test Driven Development: what it is, and what it is not. \(freecodecamp.org\)](http://freecodecamp.org/test-driven-development/what-it-is-and-what-it-is-not)
- [Unite Austin 2017 - Testing for Sanity: Using Unity's Integrated TestRunner – YouTube](https://www.youtube.com/watch?v=88331338380)
- [Graphics tests, the last line of automated testing | Unity Blog](https://unity3d.com/blog/unity/graphics-tests-the-last-line-of-automated-testing)
- [Mocks Aren't Stubs \(martinfowler.com\)](http://martinfowler.com/mock.html)
- [Softwaretests – Tim A. Majchrzak, Herbert Kuchen](#)