

DIGITAL CRAFTSMANSHIP NORDOBERPFALZ,  
15.6.2021

---

# BEHAVIOUR-DRIVEN DEVELOPMENT AM BEISPIEL


## DER FLO

- ▶ Florian Dütsch
- ▶ CTO @ [adigi.ai](https://adigi.ai)
- ▶ [mail@florian-duetsch.de](mailto:mail@florian-duetsch.de)
- ▶ [github.com/der-flo/bdd\\_dcn\\_website](https://github.com/der-flo/bdd_dcn_website)

**A MARKETING-FOLIE BRAUCH MA  
NIARD.**

der Flo

## AGENDA

- ▶ Mein Start mit BDD
- ▶ BDD?!
- ▶ **Live-Coding**
- ▶ Erfahrungen, Erkenntnisse
- ▶ Fragen, Diskussion
- ▶ FRA - GER 

# MEIN START MIT BDD

- ▶ Große Web-Projekte mit dynamisch typisierter Programmiersprache (Ruby)
- ▶ Geschwindigkeit, Qualität, *developer happiness* verbessern
- ▶ Mehr Sicherheit bekommen
  - ▶ für Refactorings
  - ▶ für zügiges Deployment
- ▶ ⇒ ohne automatisierte Tests geht es nicht.
- ▶ Wie kann ich möglichst viel Nutzen aus solchen Tests ziehen?
- ▶ Gibt es eine agile Entwicklungsmethode, die Tests natürlich entstehen lässt?

## TDD – TEST-DRIVEN DEVELOPMENT

- ▶ Aus *Extreme Programming* - *Acceptance Test-Driven Planning* (ATDP)
- ▶ *test first*
- ▶ *Top-Down*: Mit Akzeptanztest starten
- ▶ Nur so viel Code, bis Test grün ist  
kein neuer Code ohne fehlschlagenden Test
- ▶ *red - green - refactor*, permanentes Feedback auf allen Ebenen
- ▶ *emergent design*
- ▶ Naming-Problem: Objekt-Tests (*arrange - act - assert*)

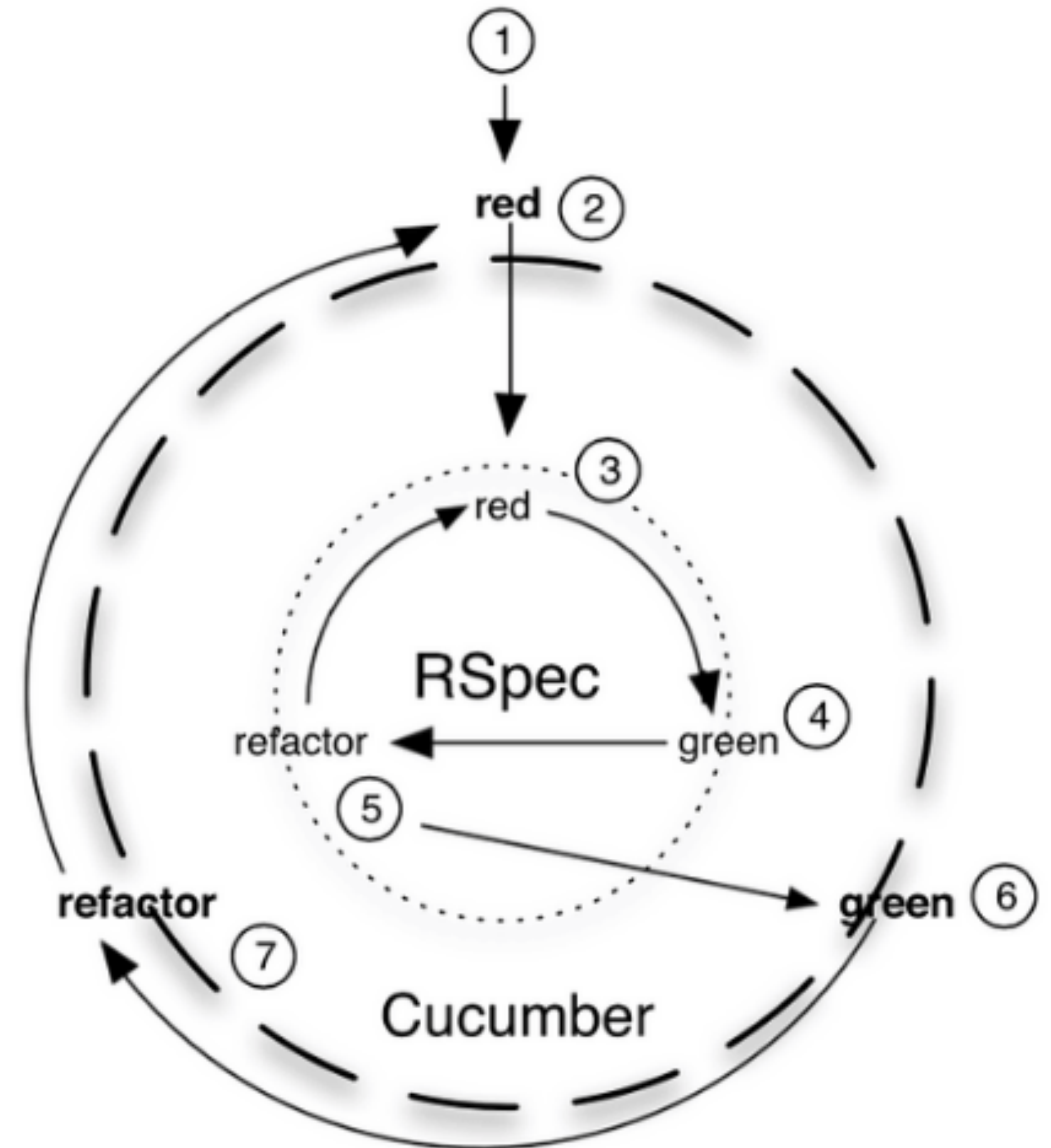
# BDD – BEHAVIOUR-DRIVEN DEVELOPMENT

- ▶ *Behaviour* spezifizieren und verifizieren
- ▶ Implementierung vs. Verhalten: Nicht testen, was ein Objekt *ist*, sondern was es *tut*
- ▶ Stakeholder interessieren sich nicht für Objekte und Implementierungen, sondern dafür, wie sich die Anwendung *verhält*
- ▶ BDD schafft eine gemeinsame Sprache: *given - when - then* (z. B. Gherkin)
- ▶ Ggf. mehr isoliertes Testen, Mocking

## BDD CYCLE

- ▶ Äußerer Kreis: Verhalten der Anwendung
  - ▶ *features, scenarios, steps*
  - ▶ Mit rotem *Step* starten, nach innen wechseln
- ▶ Innerer Kreis: Verhalten der Objekte
  - ▶ *descriptions, examples*
  - ▶ Wenn grün und nach Refactoring wieder nach außen wechseln

Quelle: The RSpec Book, David Chelimsky





**PRAKTISCH: DCN-  
WEBSITE**

## FEATURE:

Als ein an DCN Interessierter  
möchte ich die nächsten Events auf der Website sehen,  
damit ich mich informieren und ggf. anmelden kann.

fiktiver Stakeholder

## **AKZEPTANZKRITERIEN:**

- Zeige die zwei nächsten Events**
- Zeige Titel, Datum und einen Meetup-Link**

fiktiver Stakeholder

# MEIN ENTWICKLUNGS-STACK

- ▶ Ruby
- ▶ Ruby on Rails
- ▶ Cucumber
- ▶ RSpec
- ▶ Für euch: [https://de.wikipedia.org/wiki/Behavior\\_Driven\\_Development#Werkzeuge](https://de.wikipedia.org/wiki/Behavior_Driven_Development#Werkzeuge)

**CODEN. JETZT!**

**Teilnehmer**

**ERKENNTNISSE,  
ERFAHRUNGEN**

# AUCH FÜR PRODUKT-MANAGER, TEAMLEITER, ENTSCHEIDER

- ▶ Das passt in einen agilen Prozess!
  - ▶ Was passiert nach Sprint Planning Meeting, Daily und vor Retrospektive? BDD!
  - ▶ Wie setze ich ein Akzeptanzkriterium agil um, wie code ich? Mit BDD!
- ▶ Lesbare Akzeptanztests! (Features/Szenarien)
  - ▶ Das können auch Stakeholder verstehen
  - ▶ Dokumentation der Software?
- ▶ Selbstvertrauen (*confidence*) für Auslieferung, Refactorings, Anpassungen und Erweiterungen
- ▶ Eine der Vorbedingungen für Continuous Integration, v. a. für Continuous Deployment/Delivery

### TOP-DOWN!

- ▶ Von den Tests / Spezifikationen von oben nach unten leiten lassen
- ▶ Paradigmenwechsel?
- ▶ Nicht ganz unten starten und Code bauen, den keiner braucht
- ▶ (Modul-)Integration am Anfang, nicht am Ende des Zyklus
- ▶ UI-/UX-ler können eine ganz neue Rolle im Team bekommen, weil mit ihrem Themenfeld gestartet wird
- ▶ Toller Fokus: *Wo war ich nochmal? Wo geht es weiter?* Schnell beantwortet.



### VORBEDINGUNGEN

- ▶ Notwendigkeit von automatisierten Verhaltenstests sehen und sich in diese Richtung bewegen wollen → Bock drauf haben!
- ▶ Erfahrung mit Software-Stack
- ▶ Testing-Know-How
  - ▶ Wie teste ich integriert, isoliert, von außen, von innen, ...
  - ▶ Komme ich an schnelles Feedback?
- ▶ Zeitbudget für Einarbeitung

# ERFAHRUNGEN

- ▶ Vor allem in den ersten Tagen/Wochen keine Abkürzungen gehen! Refactoring nicht vergessen!
- ▶ Äußere Tests (Akzeptanztests) wichtig!
  - ▶ In der Regel *End-to-end*
  - ▶ Sie verifizieren die korrekte Integration.
  - ▶ Sie sagen uns, wann wir mit dem Programmieren aufhören können.
  - ▶ Ansonsten nicht zu viel über Test-Pyramide und isolierte Tests (Mocking) den Kopf zerbrechen
- ▶ BDD eignet sich oft
  - ▶ Auch für Bestandsprojekte
  - ▶ Etwas weniger für explorative Entwicklung, kurzlebige Projekte
- ▶ Externer State / externe Systeme nerven beim Testen ⇒ Hexagonal Architecture, Onion Architecture, Ports+Adapters

# ERKENNTNISSE FÜR ENTWICKLER

- ▶ Sehr hohe Testabdeckung entsteht automatisch
  - ▶ kein langweiliges „tests last“
  - ▶ sicher relevante Tests
  - ▶ Futter für den Build-Server
- ▶ Spezifikationen/Tests können Kommentare ersetzen.
- ▶ Debugging
  - ▶ Deutlich weniger
  - ▶ Viel leichter/fokussierter in Tests
- ▶ Schneller Workflow
  - ▶ Quasi nie der Browser nötig gewesen, trotz Webanwendung

**HER MIT EUREN FRAGEN!**

der Flo

**AUSPROBIEREN!**

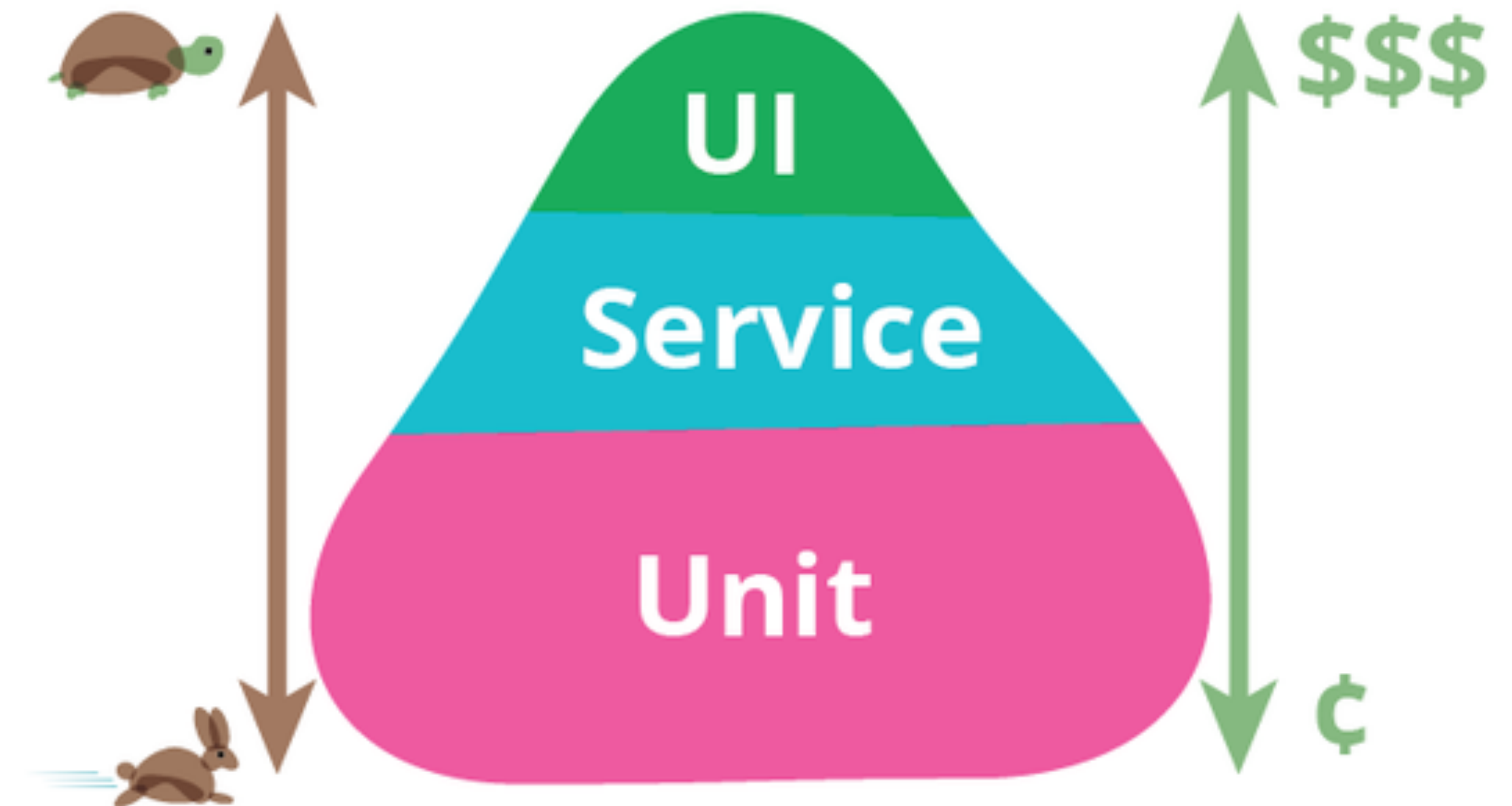
**DANKE!**

der Flo

## TESTING-PYRAMIDE 💣

Wie viele Tests auf welcher Ebene?

- ▶ 🤔 Nach oben integrierter
  - ▶ 😞 Nach oben teurer, langsamer und fragiler
  - ▶ 😊 Nach oben mehr *confidence*
  - ▶ 🙅 Erkenntnis: Nicht alles von außen testen
- 
- ▶ Quelle: <https://martinfowler.com/bliki/TestPyramid.html>
  - ▶ <https://martinfowler.com/articles/2021-test-shapes.html>



### WAS IST PASSIERT?

1. Feature entgegen genommen und für Stakeholder verständlich als *Behaviour* der Anwendung beschrieben
2. *Behaviour* der Objekte/Implementierungsartefakte spezifiziert und implementiert
3. Mit Refactoring-Schritten Code und Specs in eine saubere Form gebracht
4. Feature geliefert!



# The RSpec Book

Behaviour-Driven Development  
with RSpec, Cucumber,  
and Friends

David Chelimsky  
with Dave Astels,  
Zach Dennis,  
Aslak Hellesøy,  
Bryan Helmkamp,  
and Dan North

Foreword by Robert C. Martin  
(Uncle Bob)

Edited by Jacquelyn Carter



OLDIE, GOLDIE, RUBY, RAILS

THE RSPEC BOOK,  
CHELIMSKY, 2010



*The Addison-Wesley Signature Series*



# GROWING OBJECT-ORIENTED SOFTWARE, GUIDED BY TESTS

STEVE FREEMAN  
NAT PRYCE



TDD-KLASSIKER

---

„GOOS“, FREEMAN/  
PRYCE, 2010