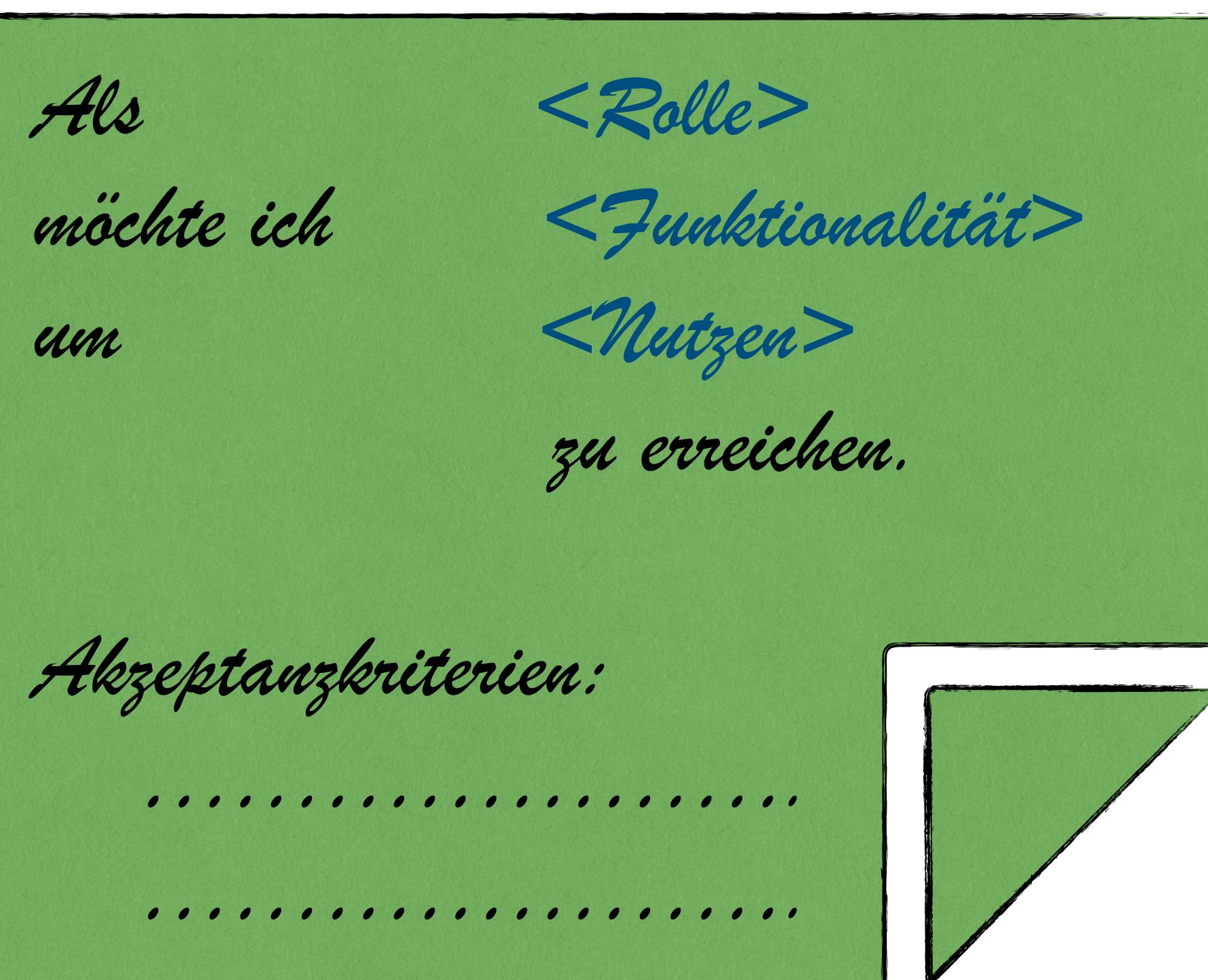


# [Workshop]

szut

## User Stories, Planning und Testing



[l.michaelis@neusta.de](mailto:l.michaelis@neusta.de)



[@larmicDE](https://twitter.com/larmicDE)



larmic



LarsMichaelis



# Tag 1

Willkommen

Theorie User Stories

Schneiden von User Stories

~ 12:00

Beispiele der SZUT

Planning I + II

16:00

# Mittag

Theorie zu Testing

Unit Tests und Mocking

Spring Boot Tests  
*Integrationstests*

Spring Boot Tests  
*Slicetests*

Abschluss und Feedback

15:00



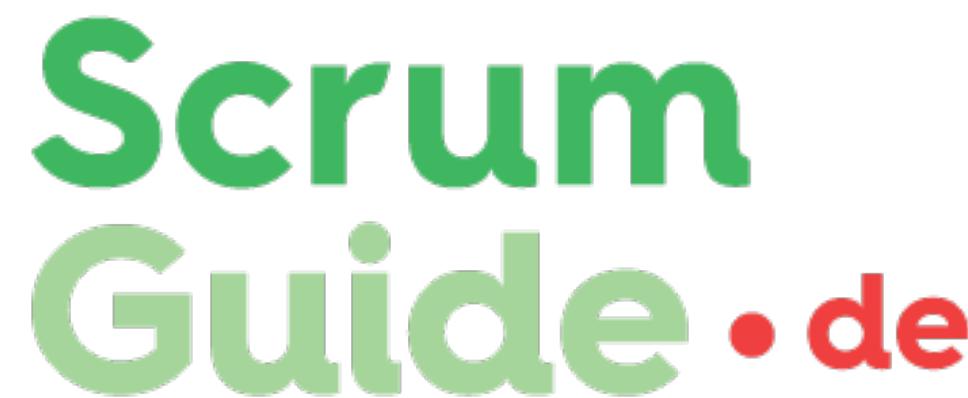
*Art I*

# THEORIE USER STORIES



# Definition

## [User Story]



... ist eine kurze Beschreibung (*Story*) dessen, was ein Benutzer (*User*) will.



# Definition

## [User Story]

Scrum  
Guide.de

powered by Agile Scrum Group

... ist eine kurze Beschreibung Geschichte (*Story*) dessen, was ein Benutzer (*User*) will.

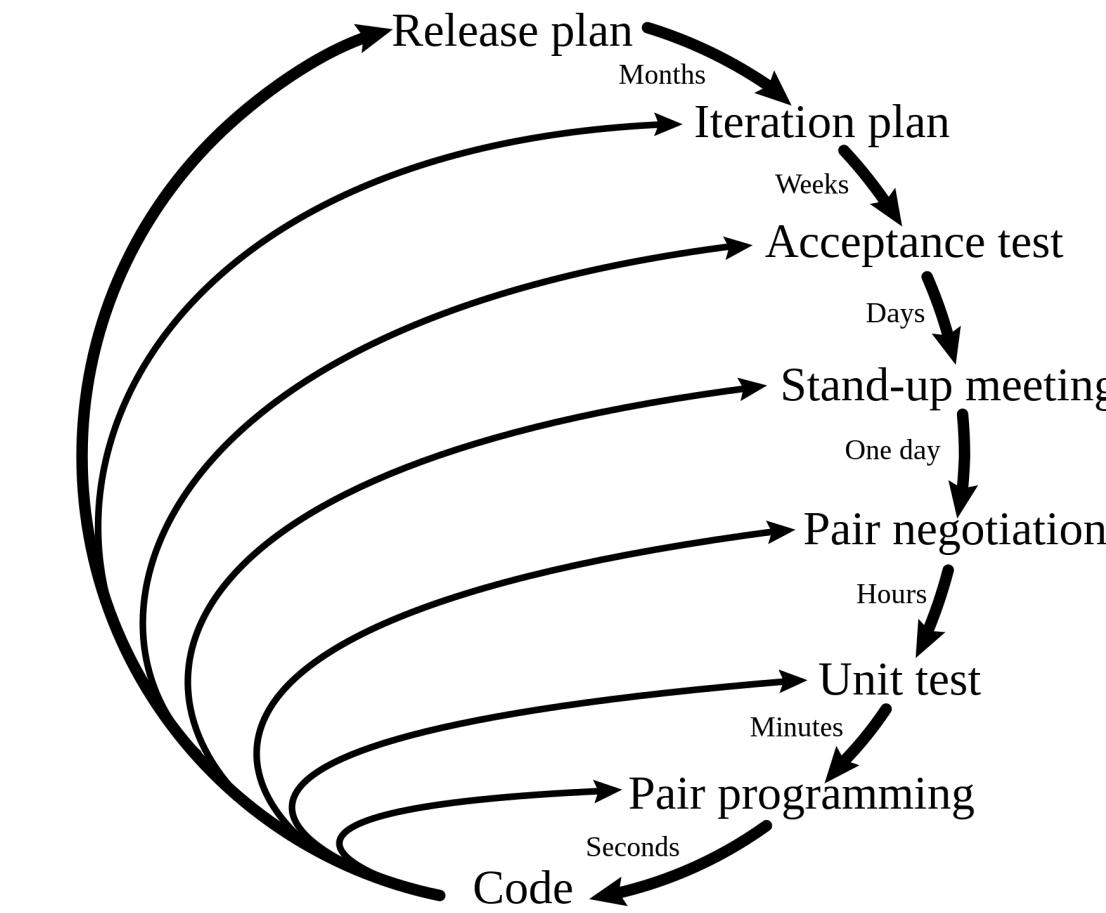


# Irrtum

## [User Story]



Planning/feedback loops



# Definition

## [User Story]



**... um die gewünschte Funktionalität aus Sicht eines Anwenders zu beschreiben.**



# Vorteile

## [User Story]

1. Leicht zu verstehen
2. Schnell erstellt
3. Unterstützt integrative Vorgehensmodelle



# Anforderungen

## [User Story]



Kurze Sätze / einfache Worte

① Ohne (großes) technisches Wissen verständlich

Aus Sicht des Anwenders



Wer will was und warum?



Akzeptanzkriterien

Ansprechpartner



# Schreibweisen

## [User Story]

Als  
möchte ich  
um  
zu erreichen.

<Rolle>  
<Funktionalität>  
<Nutzen>

Akzeptanzkriterien:

.....  
.....



# Schreibweisen

## [User Story]

Um (Nutzen) als (Rolle) zu erreichen, möchte ich (Funktionalität / Ziel / Wunsch)

*Chris Matts*

Als (wer) (wann) (wo) möchte ich (was) (warum)

*Wer, wann, was, warum?*

Als (Rolle) möchte ich (was) (warum)

*Rachel Davies*

Als (Persona) möchte ich (was) (warum)

*Roman Pichler*



# Akzeptanzkriterien

## [User Story]

Checkliste



Bindetglied zwischen Anforderungen und den Testfällen

Fragen Beantworten: Wer? Was? Wann? Wo?

Schreibweise im Präsens





## [Anforderung]

Ich als Lieferant von DSL-Anschlüssen möchte unserem Partner [x] unsere Produktverfügbarkeiten an allen bekannten Adressen in dem neuen SPR/I 1.6 Format als Replikat bereitstellen.



## [Akzeptanzkriterien]

- ✓ Das Replikat ist unter [http://meine.domain.de/\[datum\]/replikat\\_spri\\_1.6.zip](http://meine.domain.de/[datum]/replikat_spri_1.6.zip) abrufbar
- ✓ Es werden Zugangsdaten benötigt, um das Replikat abrufen zu können
- ✓ Das Replikat ist tagesaktuell
- ✓ Das Replikat steht spätestens morgens um 6:00 bereit
- ✓ Das Datum in der URL hat das Format: yyyy-MM-dd (Beispiel: 2022-02-14)
- ✓ Unserem Partner [x] wurden die Zugangsdaten mitgeteilt

## ✗ [Abgrenzung]

- ❖ Das Abräumen alter Replikate ist nicht Bestandteil dieser Story



## [Ansprechpartner]

- ★ Max Mustermann (Fachbereich), max@muster.mann, 555 3549
- ★ Monika Musterfrau (Kontakt beim Partner [x]), monika@muster.frau, 555 3548
- ★ Toni Betreiber (Betrieb), toni@betreiber.de, 555 3547



# Häufige Fehler

## [User Story]

Zu viel / zu großer Umfang

Komplexe (schwierige) Formulierungen

Verweise auf andere Dokumente

Dokumentation nicht-funktionaler Anforderungen

Andernde Schreibweisen



# Schätzen

[User Story]

*Storypoints*



*vS*

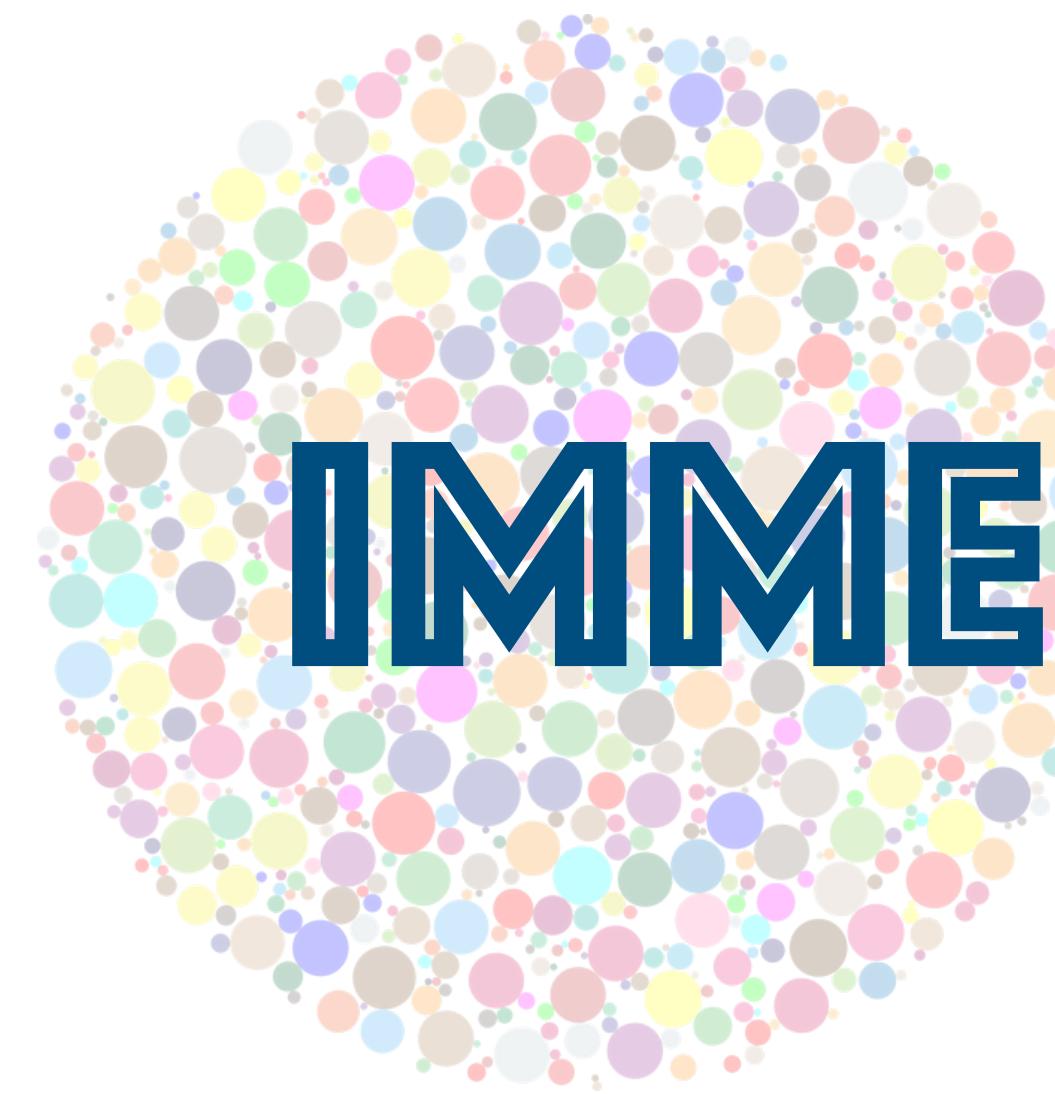
*Stunden*



# Schätzen

[User Story]

*Storypoints*



**IMMER UNGENAU!**

*Stunden*



# Schätzen

## [User Story]



*Der Mensch kann ...*

*...kann schlecht absolute Dinge schätzen*

- Ⓐ **der Zug ist 200m lang**
- Ⓐ **das Auto ist 125 km/h schnell**
- Ⓐ **das Haus ist 15m hoch**

*...kann gut relative Dinge schätzen*

- Ⓐ **das ist größer als das**
- Ⓐ **das dauert länger als das**
- Ⓐ **das ist ungefähr halb so lang wie das**



# Schätzen

## [User Story]



Referenz-Story



Fibonacci



# Schätzen

## [User Story]



*analytisch (Referenz-Story)*

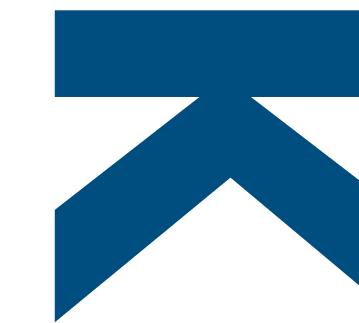


*Gefühl*



# Schätzen - KAR-Index

## [User Story]



### Komplexität

Wie komplex ist die Implementierung und der Test?  
Kann automatisiert getestet werden?  
Wieviele Software-Module sind beteiligt



### Aufwand

Wie hoch ist der Implementierungsaufwand?  
Wie hoch ist der manuelle Testaufwand?  
Gibt es zusätzliche Abstimmungen?



### Risiko

Gibt es externe Abhängigkeiten? (Schnittstelle, Betrieb, ...)  
Fehlt noch Wissen? (Technologie, Sachlichkeit, ...)  
Gibt es Deadlines?



# Weitere Begriffe

## [User Story]

Epic

Technical Story

Spike Story

Produkt Backlog

Sprint Backlog



*Akt II*

# SCHNEIDEN VON USER STORIES





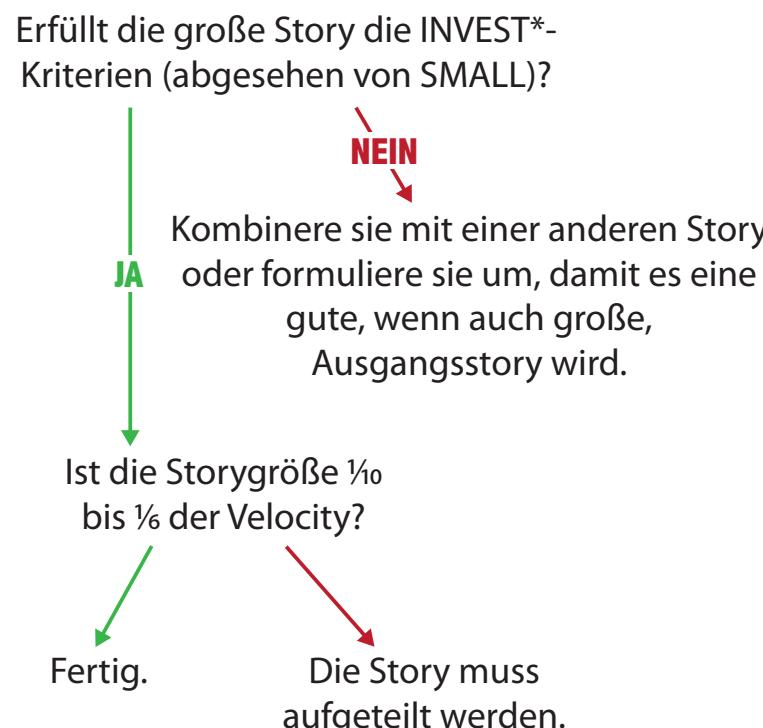
*William (Bill) Wake*



# USER STORIES AUFTTEILEN

1

## DIE EINGANGSSTORY VORBEREITEN



## WORKFLOW SCHRITTE

Kann man die Story so aufteilen, dass Workflowbeginn und -ende zuerst und Stories aus der Mitte des Workflows als Erweiterung umgesetzt werden?

## PERFORMANCE NACHLAGERN

Kann man die Story so aufteilen, dass erst eine nur funktionierende Version umgesetzt wird und später der nicht-funktionalen Anteil?

Wird die Story dadurch komplex, dass nicht-funktionale Anforderungen wie Performance umgesetzt werden sollen?

Kann man die Story so aufteilen, dass zuerst der einfache Kern und später Erweiterungen umgesetzt werden?

## SIMPEL / KOMPLEX

Kann man erstmal nur eine Story-Ausprägung mit dem Hauptaufwand auswählen und weitere Ausprägungen später hinzufügen?

## GRÖßTER AUFWAND

Kann man zuerst einen Minimalworkflow heraustrennen, welcher später mit anderen Stories erweitert wird?

## OPERATIONEN

Kannst Du die Operationen in separate Stories aufteilen?

Enthält die Story mehrere Operationen? (z.B. etwas "verwalten" oder "konfigurieren")?

Enthält die Story verschiedene Ausprägungen von Geschäftsregeln? (z.B. gibt es einen Domänenbegriff wie "flexible Datumswerte", der mehrere Variationen nahelegt?)

## MUSTER ANWENDEN

## VARIATION DER GESCHÄFTSREGELN

## VARIATION DER DATEN

Macht die Story die gleichen Dinge mit unterschiedlichen Daten?

Kann man die Story so aufteilen, dass erst ein Teil der Daten und später ein anderer Teil der Daten verarbeitet wird?

## EINEN SPIKE HERAUSBRECHEN

Immer noch keine Idee, wie die Story aufgeteilt werden kann?

Gibt es ein kleines Stück, das verständlich genug ist, um damit zu beginnen?

Schreibe diese Story zuerst, setze sie um und beginne diesen Prozess wieder von vorne.

3

## DIE AUFTHEILUNG ÜBERPRÜFEN

Sind die neuen Stories etwa gleich groß?

Ist jede Story in etwa  $\frac{1}{10}$  bis  $\frac{1}{6}$  der Velocity groß?

Erfüllt die Story die INVEST-Kriterien?

Könnten manche der Stories prinzipiell depriorisiert oder komplett gestrichen werden?

Gibt es eine Story mit der man offensichtlich anfangen kann, um frühen Wert, Lernen oder Risikovermeidung usw. zu erreichen?

DU BIST FERTIG ODER TESTEST NOCH EIN ANDERES MUSTER AUF BESSERE EIGNUNG.

Probieren ein anderes Muster mit der UrsprungssStory oder den neu entstandenen Stories.

Probieren ein anderes Muster.

Probieren ein anderes Muster. Wahrscheinlich ist noch etwas Überflüssiges in jeder der Stories.

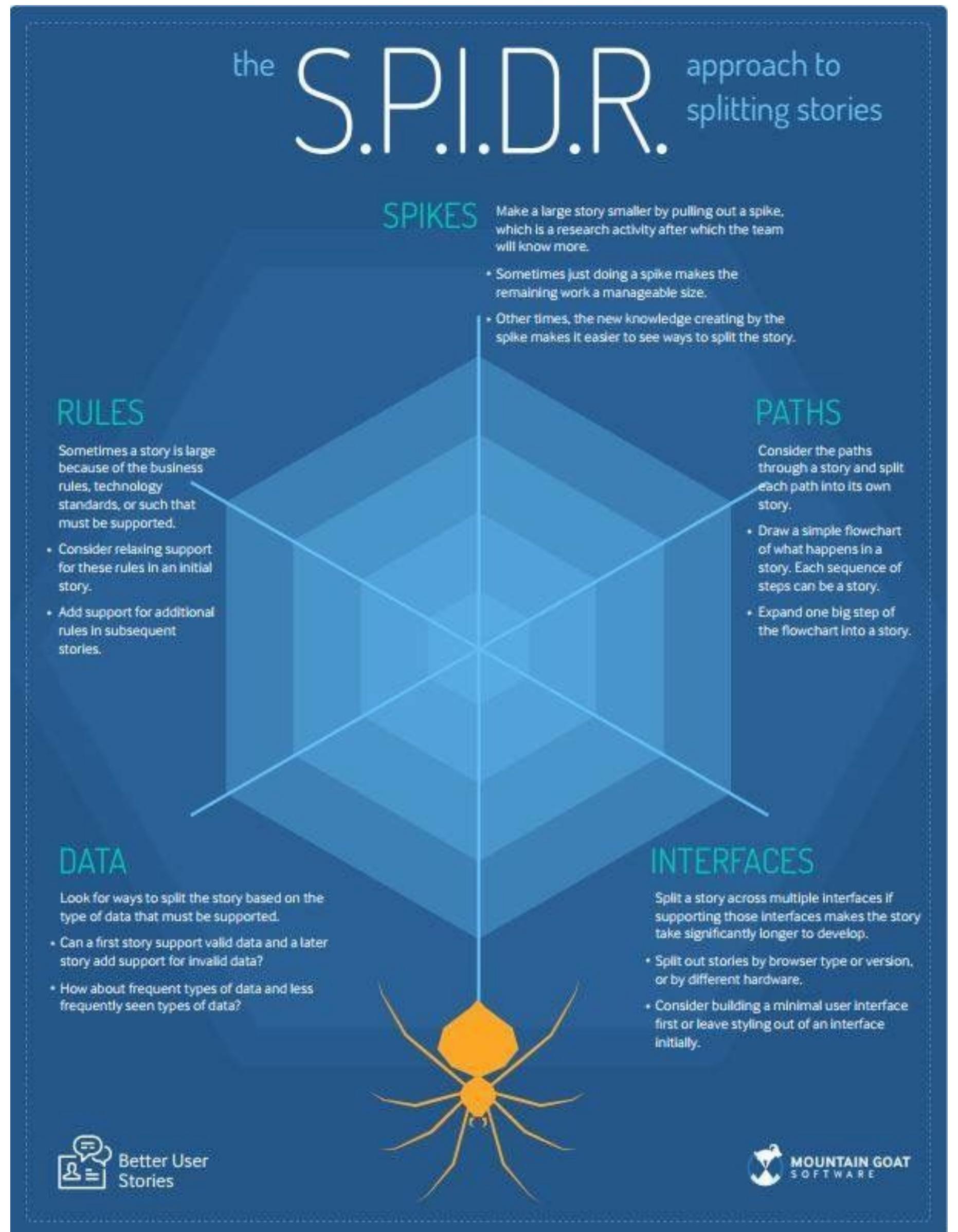
Probieren ein anderes Muster, um die Story zu zerlegen.

Welche 1-3 Fragen halten Dich am meisten zurück?

Mach eine Pause und versuch es erneut.

Schreibe eine explorative Story, welche mit minimalem Aufwand zur Klärung dieser Fragen führt und beginne diesen Prozess von vorne.





*Mike Cohn*



# Weitere Tipps

## [User Stories schneiden]

Kein Frontend vs Backend

Kein technischer Schnitt

Stories nicht alleine schreiben



# Weitere Tipps

## [User Stories schneiden]

Kein Frontend vs Backend

**ERFAHRUNG!**

Stories nicht alleine schreiben



*Akt III*

**SZUT STORIES**



*Akt IV*

**PLANNING I + II**



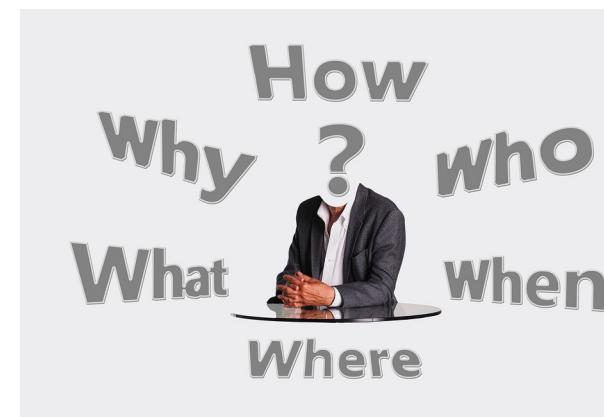
# Allgemeines

## [Planning I + II]



Warum ist dieser Sprint wertvoll?  
*Sprint-Ziel?*

Was kann in diesem Sprint abgeschlossen (Done) werden?  
*DoR?*



Wie wird die ausgewählte Arbeit erledigt?  
*DoD?*



# Allgemeines

## [Planning I + II]



Warum ist dieser Sprint wertvoll?  
Sprint-Ziel?

# 2 STUNDEN PRO SPRINT-WOCHE



Wie wird die ausgewählte Arbeit erledigt?  
DoD?



# Sprint-Ziel

## [Planning I]

Ein Produkt ist ein Instrument, um Wert zu liefern. Es hat klare Grenzen, bekannte Stakeholder:innen, eindeutig definierte Benutzer:innen oder Kund:innen. Ein Produkt kann eine Dienstleistung, ein physisches Produkt oder etwas Abstrakteres sein.

1-2 Sätze

fachlich

messbar

externe Abhängigkeiten sind gelöst

[ Am Ende des Sprint steht dem Carrier [x] ein Download-Link für das SPR/I 1.6 Format zur Verfügung. Die Zugangsdaten wurden ihm mitgeteilt. ]



# Definition of Ready

## [Planning I]



*Jeff Sutherland*

**ready is when the team says: ,Ah, we got it**

### Beispiele für DoR

- ✓ klein genug
- ✓ geschätzt
- ✓ hat Akzeptanzkriterien
- ✓ hat Anforderer / Stakeholder



# Definition of Done

## [Planning II]

Die Definition of Done ist eine formale Beschreibung des Zustands des Increments, wenn es die für das Produkt erforderlichen Qualitätsmaßnahmen erfüllt

Transparenz

Team-Standard

Klarheit

Mindestmaß der Qualität

### Beispiele für DoD

- ✓ Akzeptanzkriterien wurden vollständig erfüllt
- ✓ Ein Code Review wurde durchgeführt
- ✓ Dokumentation wurde erstellt oder erweitert
- ✓ Testabdeckung hat mindestens 90%



# Vorbedingung

## [Planning I]



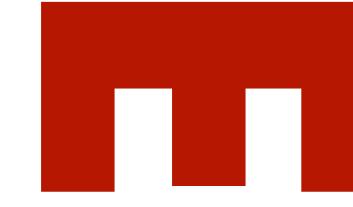
### Detailed Appropriately

Product Backlog Einträge sind angemessen detailliert



### Estimated

Product Backlog Einträge sind geschätzt



### Emergent

Product Backlog Einträge verändern sich (Hinzufügen, Ändern, Löschen)



### Priorised

Product Backlog Einträge sind priorisiert geordnet



# Planning I



*Teilnehmer*

**Product Owner**

**Scrum Master**

**Development Team**

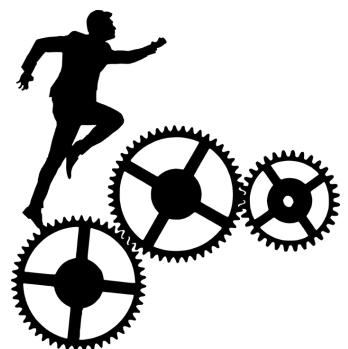


*Zweck*

**Was ist das Sprint-Ziel?**

**Welche Stories kommen in den Sprint?**

**Was wird in einem Sprint geschafft?**



*Ablauf*

**Der PO stellt die jeweilige Story vor**

**Das Team zieht die Story in den Sprint DoR?**

**Story wird gemeinsam im Sprint sortiert**

*Was ist mit nicht-abgeschlossenen Stories??*



*20% für*  
**Bugs**  
**Refactorings**  
**Termine**



# Planning II



*Teilnehmer*  
**Development Team**  
ggf. Experten



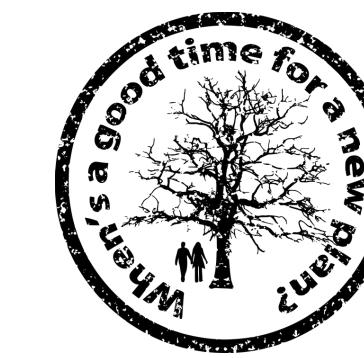
*Zweck*

**Wie erreichen wir das Sprint-Ziel?**  
**Welche konkreten Aufgaben (Tasks) gibt es?**  
**Welche technischen Hindernisse gibt es?**



*Ablauf*

**Product Owner und Scrum Master dürfen\*** den Raum verlassen  
**Tasks werden für jede Story erstellt**  
**Ein Task dauert in der Regel nicht länger als einen Tag**



*Commitment*

**Schaffen wir alles?**  
**Müssen wir zurück in Planning I?**  
**Sind noch Fragen offen?**



\* Kann ich nicht empfehlen. Es gibt immer Nachfragen und Diskussionen!

# Häufige Fehler

## [Planning I + II]

Zu spät kommen

Neue Stories vor „nicht fertigen“ Stories

Zu Detailliert

Tasks schon Personen zuweisen

Nur „Alphas“ machen mit und sprechen

Zuviele Abhängigkeiten zwischen Stories

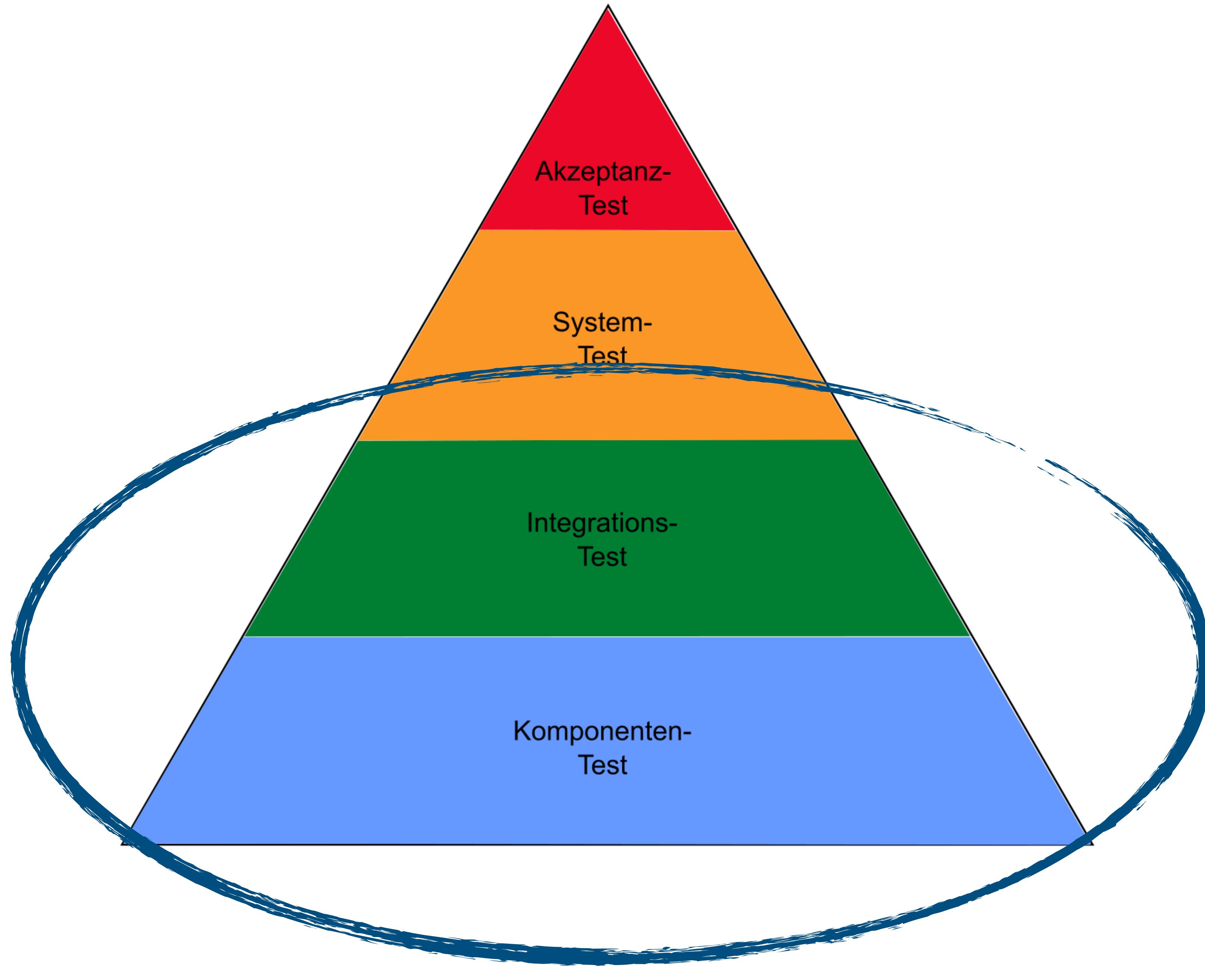
Tasks sind nicht genau genug beschrieben



*Akt V*

# THEORIE ZU TESTING

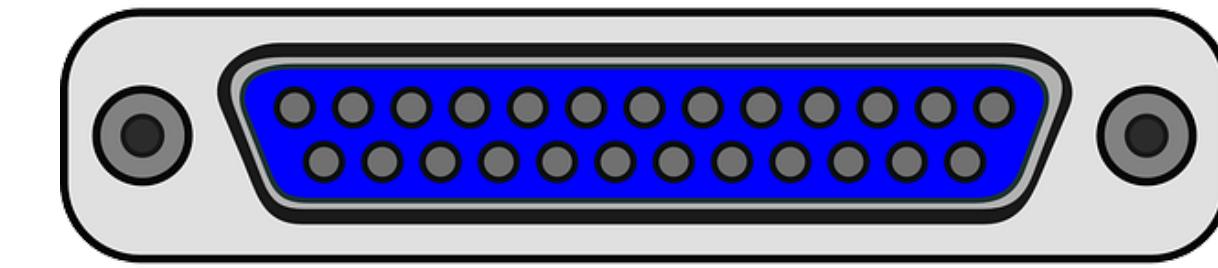
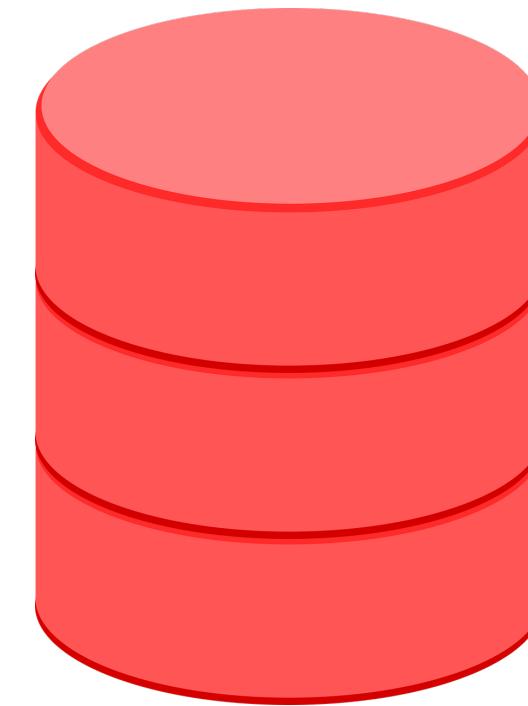




# **Was ist ein Unitest?**



# Unit test



# Unit test

## Unit Testing

Unit tests mock interactions with external dependencies (database, IO, etc.)

Developers write unit tests.

## Integration Testing

Integration tests use the real dependencies

Testers or other QA professionals usually write integration tests





**Unit Testing** bedeutet, einzelne Module einer Anwendung isoliert zu testen (ohne Interaktion mit Abhängigkeiten), um zu Bestätigen, dass der Code die Dinge richtig macht.

**Integrationstests** bedeutet zu überprüfen, ob verschiedene Module gut funktionieren, wenn sie als Gruppe kombiniert werden.





**Unit Testing** bedeutet, einzelne Module einer Anwendung isoliert zu testen (ohne Interaktion mit Abhängigkeiten), um zu Bestätigen, dass der Code die Dinge richtig macht.

**Integrationstests** bedeutet zu überprüfen, ob verschiedene Module gut funktionieren, wenn sie als Gruppe kombiniert werden.

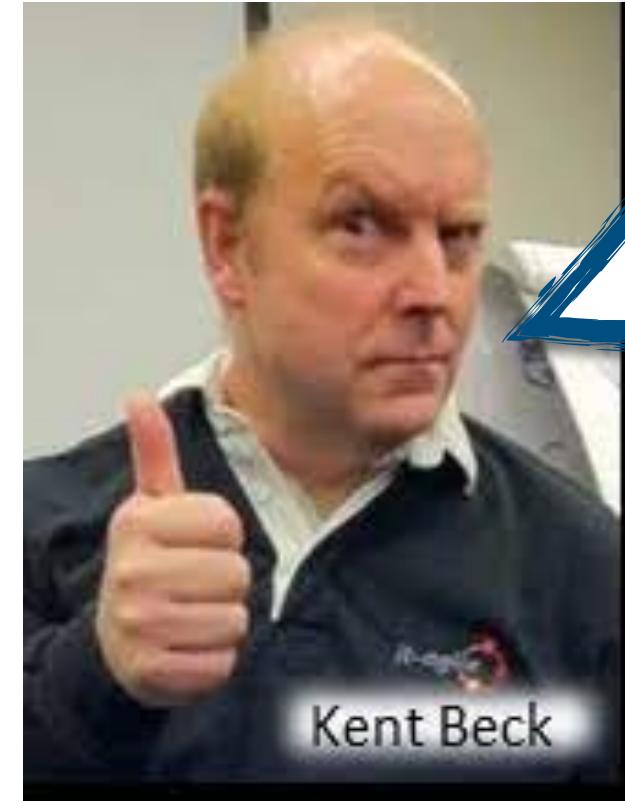






Martin Fowler

# Suite < 10 Minuten



Kent Beck

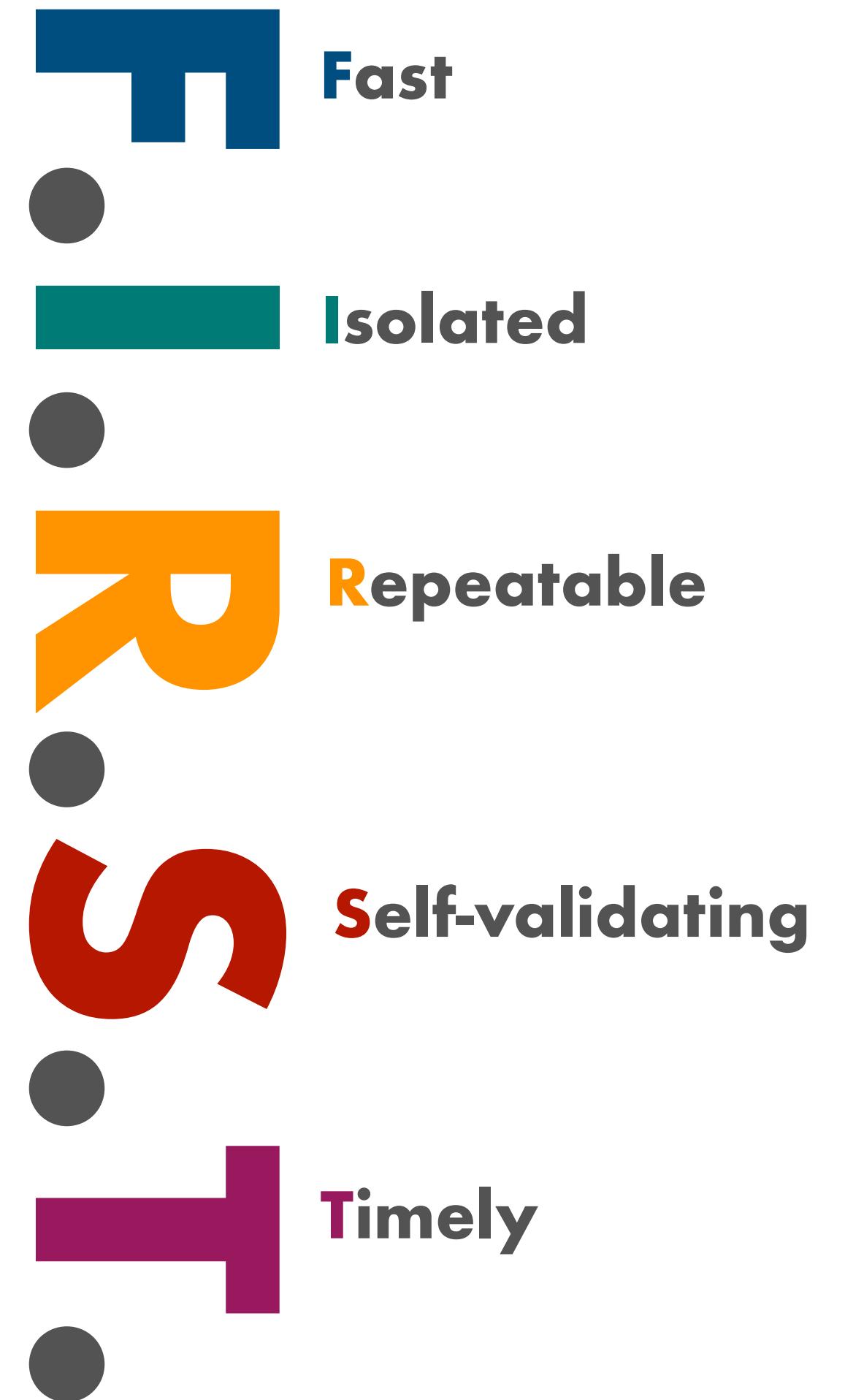


# Unitests



-  **sind isoliert.**
-  **sichern jeweils genau eine Eigenschaft ab.**
-  **sind leicht verständlich und kurz.**
-  **testen relevanten Code (und z.B. keine Getter/Setter).**
-  **werden vor dem zu testenden Code geschrieben.**



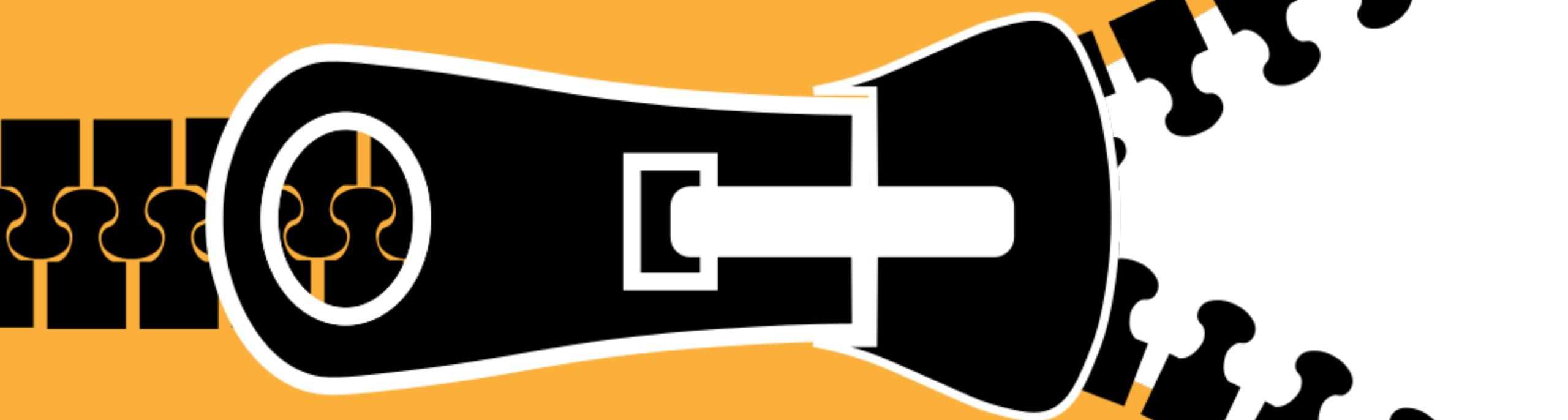




und noch viel mehr...



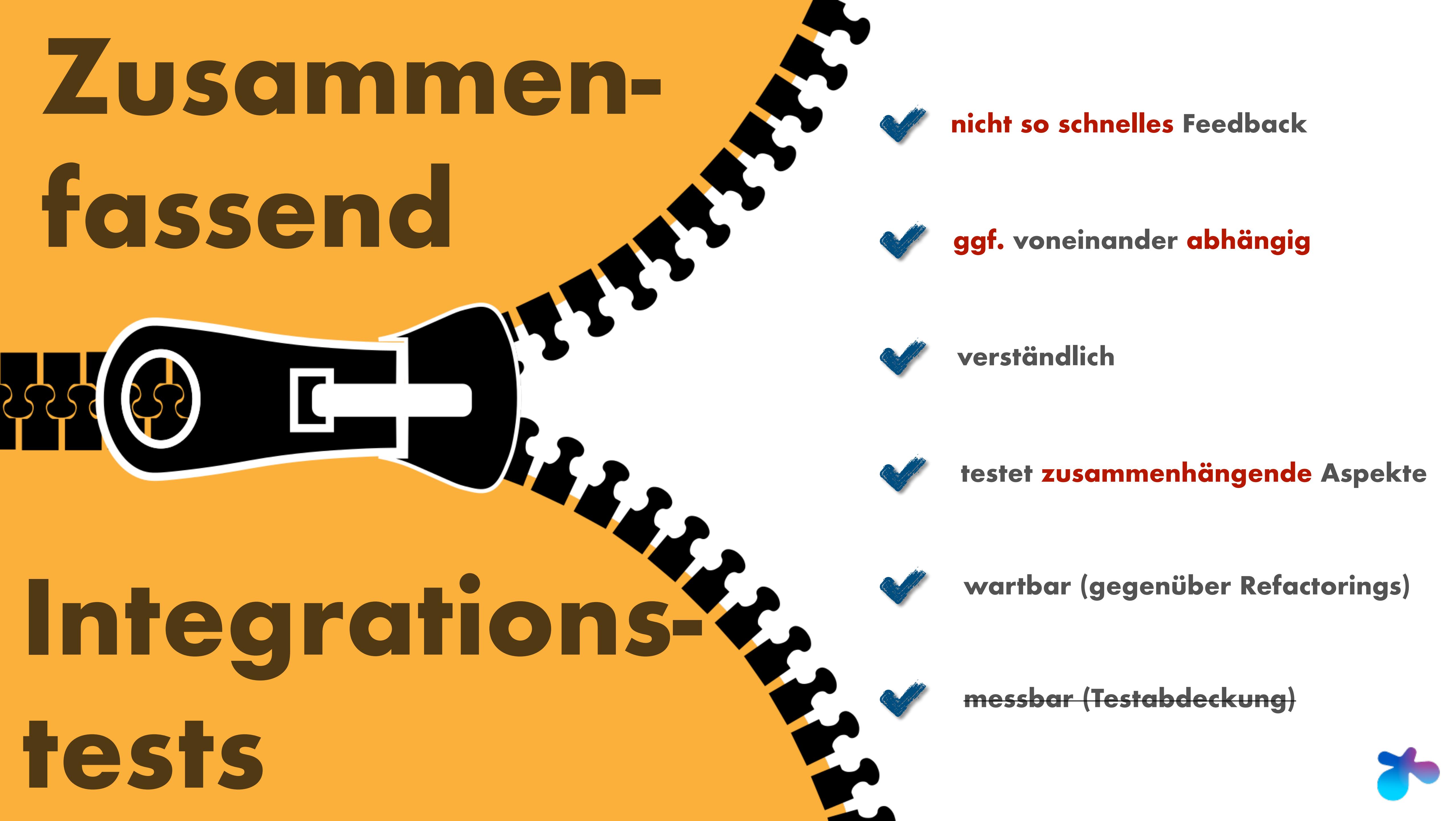
# Zusammenfassend



## Unitests

- ✓ schnelles Feedback
- ✓ voneinander unabhängig
- ✓ verständlich
- ✓ testet nur wenige Aspekte (asserts)
- ✓ wartbar (gegenüber Refactorings)
- ✓ messbar (Testabdeckung)





# Zusammen- fassend

# Integrations- tests

✓ nicht so schnelles Feedback

✓ ggf. voneinander abhängig

✓ verständlich

✓ testet zusammenhängende Aspekte

✓ wartbar (gegenüber Refactorings)

✓ messbar (Testabdeckung)



# hacking time



**Code:** [https://github.com/FA-Team-SZUT/workshop-neusta-2022-05-18\\_und\\_19](https://github.com/FA-Team-SZUT/workshop-neusta-2022-05-18_und_19)





