

Risikobewusste und modulare Testdurchführung?

Ach, was soll schon schiefgehen!

Das sind wir...



Ralf Sträßner
ralf.strassner@atruvia.de



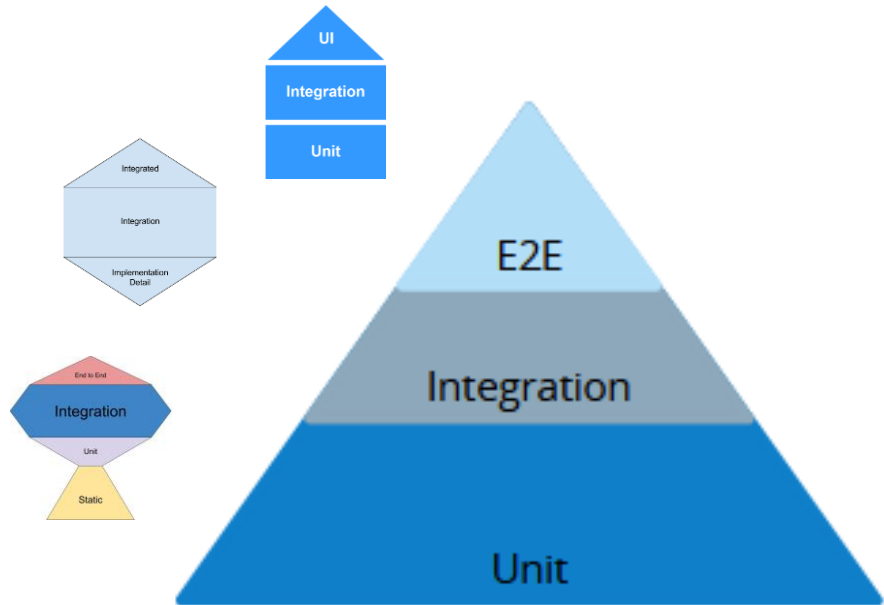
Peter Fichtner
peter.fichtner@atruvia.de



<https://sammancoaching.org/>



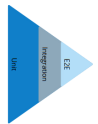
Formvollendeter Testgenuss: Ist die Testpyramide out?



"Don't trust the pyramid!"

The Test Pyramid is a model. This means that it's useful only in some contexts."

<https://www.simpleorientedarchitecture.com/test-strategy-for-continuous-delivery/>



Form: mögliches Ergebnis unseres Testansatzes

→ keine Abkürzung nehmen

→ besser betrachten, was hinter dem Modell steckt

Ist die Testpyramide out?

Nein 😊

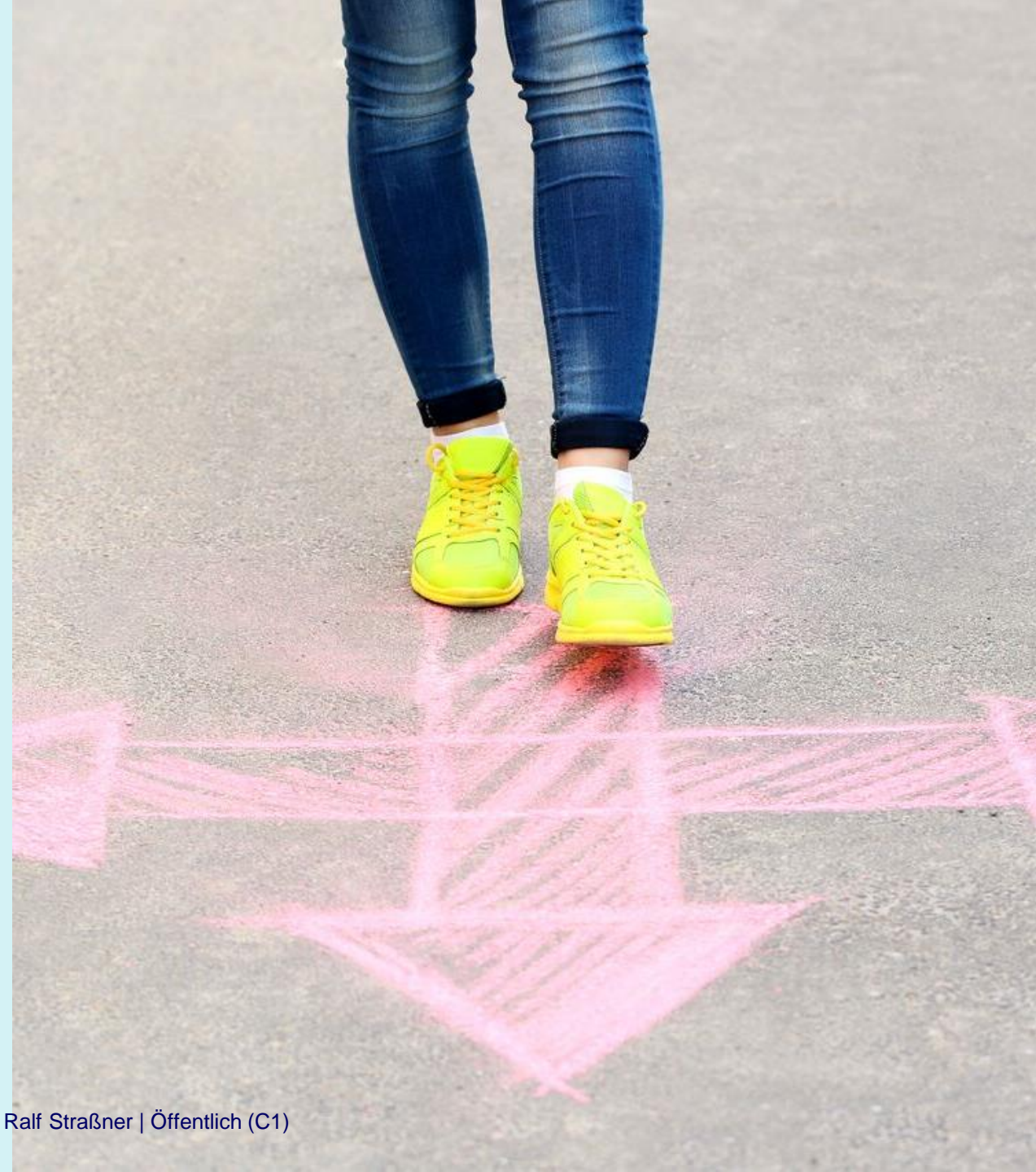
- weiterhin **wichtige Prinzipien** für **sinnvolle Abdeckung** des gesamten Produkts mit automatisierten Tests*
 - Welche **Eigenschaften** müssen bestimmte Tests/Testarten haben?
 - Was stellen sie jeweils sicher, was bewusst nicht?
- Gutes Verhältnis aus Aufwand und Nutzen
- schnelle Reaktionsfähigkeit

* gilt auch für andere Testmodelle

Und nun?

- Eine bestimmte Anzahl Tests pro Testart anstreben??
- Annahme hinter der **Darstellung** der Testpyramide:
 - Wir können viele Unit-Tests schreiben und brauchen nur wenige integrative Tests
 - Gilt für viele, aber nicht für alle Produkte
 - Richtige Prinzipien und Testeigenschaften → **automatisch richtige Verteilung der Tests?**
- Ja, und... 😊
 - ... aufgrund **Produkt-/Projektkontext**:
 - Bestimmte Testarten höher oder niedriger priorisieren
 - Evtl. Testarten sogar weglassen

Wie komme ich zu den passenden Tests?



Herangehensweise im Überblick

allgemeingültig

Testprinzipien ✓

→ geringerer Aufwand, höhere
Präzision, schnelleres Feedback

Komponentengrenzen und Testschnitt

→ einfache Testbarkeit

individuell

Kontext

Fachlichkeit, Technik, Markt, Team...

Risiken und Testziele

zielgerichtete Testarten und
passende Testmethodik

Passende Tests

→ richtige Verteilung auf
passende Testarten

Herangehensweise im Detail

allgemeingültig

Testprinzipien

→ geringer Aufwand, höhere
Präzision, schnelleres Feedback

Komponentengrenzen und Testschnitt

→ einfache Testbarkeit

individuell

Kontext

Fachlichkeit, Technik, Markt, Team...

Risiken und Testziele

zielgerichtete Testarten und passende Testmethodik

Passende Tests

→ richtige Verteilung auf
passende Testarten

Herangehensweise im Detail

allgemeingültig

Testprinzipien ✓

→ geringer Aufwand, höhere
Präzision, schnelleres Feedback

Komponentengrenzen und Testschnitt

→ einfache Testbarkeit

individuell

Kontext

Fachlichkeit, Technik, Markt, Team...

Risiken und Testziele

zielgerichtete Testarten und
passende Testmethodik

Passende Tests

→ richtige Verteilung auf
passende Testarten

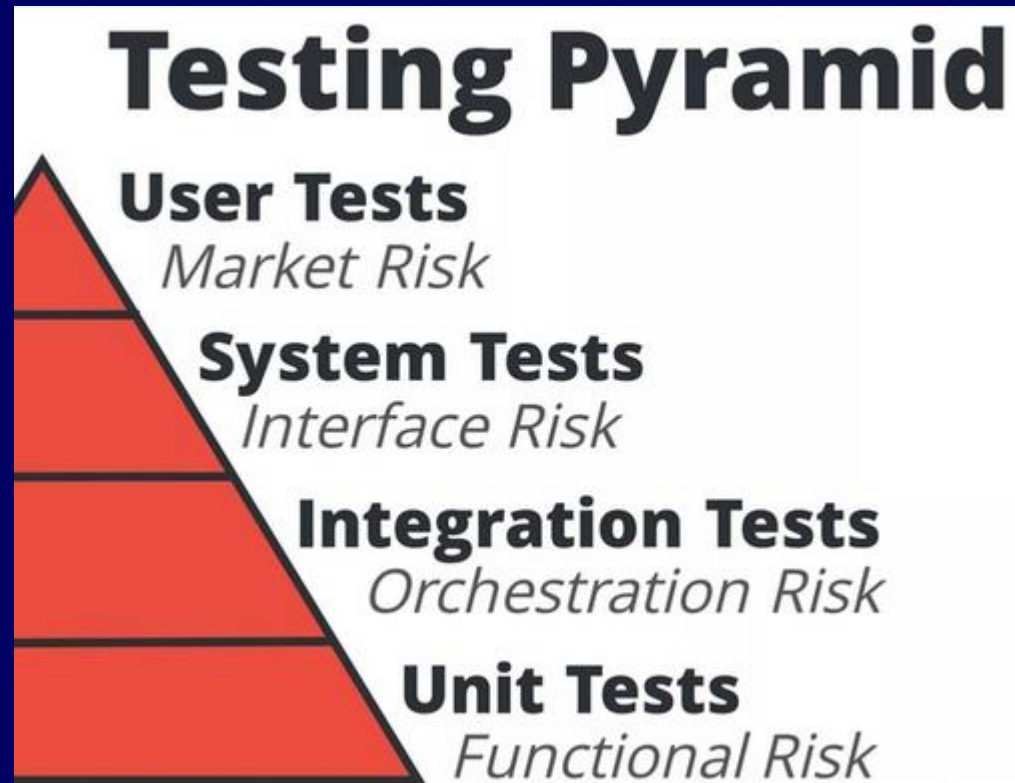
Warum ist es entscheidend, die Risiken rund um das Produkt zu kennen?

- Was ist Testen? **Risikomanagement!**
- Spezifische Risiken außer Acht lassen?
 - evtl. **Fokus** auf Testarten, die wenig zum Projekterfolg beitragen
- 100%ige Absicherung* gegen alle Risiken: in der Praxis oft **nicht wirtschaftlich**
 - Dennoch: Falls Lücken, dann nur ganz bewusst
- **Qualität** der (automatisierten) Tests nur bedingt messbar
 - Tests priorisieren, die am besten gegen Hauptrisiken schützen
 - Erhöhung der Qualität und **Wirksamkeit** dieser Tests

*damit ist nicht zwingend eine 100%ige Testabdeckung gemeint

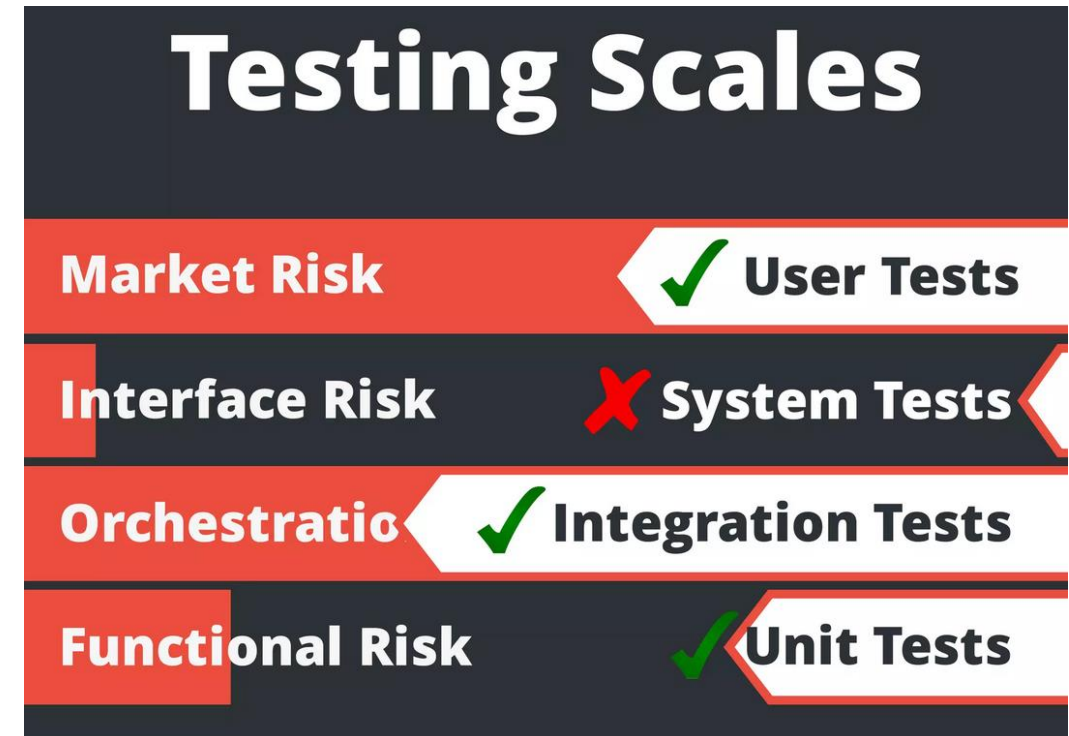
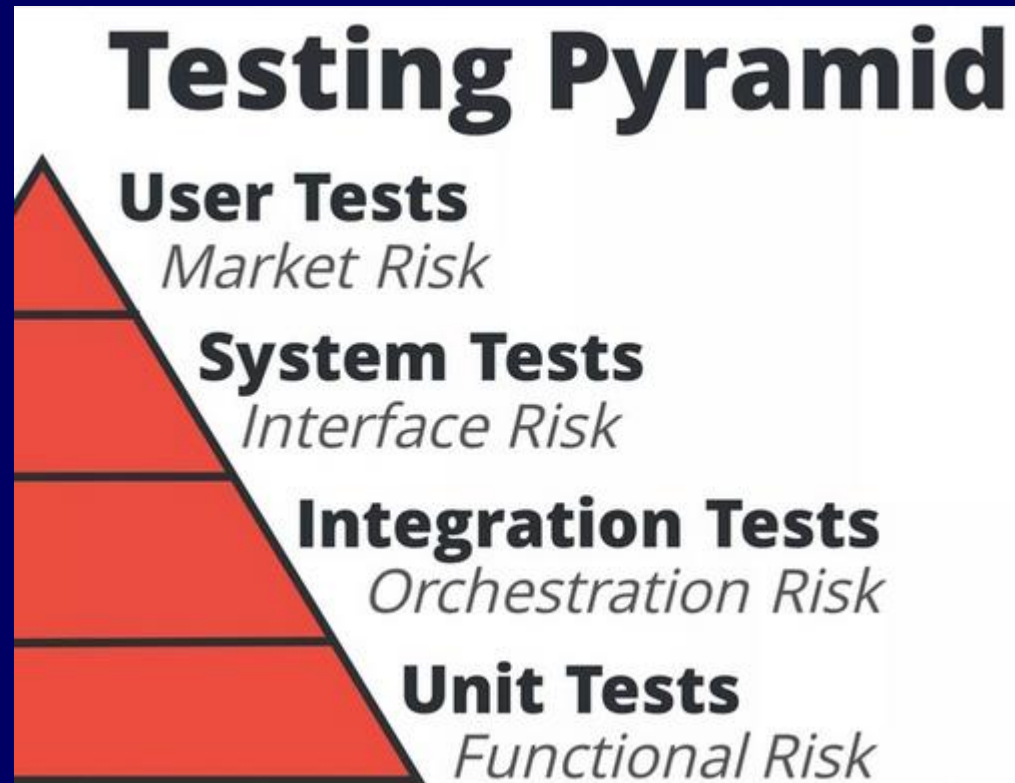
Risiken setzen die Prioritäten

<https://de.slideshare.net/todd3091/case-studies-in-terrible-testing>
<https://testguild.com/testing-pyramid/>



Risiken setzen die Prioritäten

<https://de.slideshare.net/todd3091/case-studies-in-terrible-testing>
<https://testguild.com/testing-pyramid/>



➤ Risiken und Tests im Gleichgewicht

Welche Risiken möchte ich managen?

<https://www.simpleorientedarchitecture.com/test-strategy-for-continuous-delivery/>

Kontext	Risiko	Mögliche Testziele
Komplexe Domänenlogik, komplizierter Algorithmus	Funktionales Risiko	Frühes Feedback zur Entwicklungszeit, ob die Logik korrekt ist
Viele Fremdsysteme	Integrationsrisiko	Annahmen zu den Fremdsystemen verifizieren, Unabhängigkeit beim Testen und zur Entwicklungszeit
Die Hauptaufgabe ist es, andere (unternehmensinterne) Komponenten/Services einzubinden und zu kombinieren	Orchestrierungsrisiko	Die orchestrierende Komponente isoliert testen, Vereinbarungen zwischen den Serviceanbietern sicherstellen
Bereitstellen einer Open-Source-Bibliothek	API-Design-Risiko	API aus Sicht der Verwender vorantreiben
Extrem hohe Benutzeranzahl (Last), performancekritische Prozesse, ...	Schwer erreichbare Qualitätsmerkmale	Frühzeitig mangelhafte Qualität erkennen
Unklar, ob es das richtige Produkt ist	Marktrisiko	Frühes End-User-Feedback einholen
Schwierige rollenübergreifende Zusammenarbeit, kein gemeinsames fachliches Verständnis	Risiko von Fehlkommunikation	Tests werden rollenübergreifend zusammen erstellt
Bedarf für technische Umbauten, hohe technische Schulden	Risiko beim Refactoring	Möglichst robuste Tests
...

Welche Risiken möchte ich managen?

<https://www.simpleorientedarchitecture.com/test-strategy-for-continuous-delivery/>

Kontext	Risiko	Mögliche Testziele
Komplexe Domänenlogik, komplizierter Algorithmus	Funktionales Risiko	Frühes Feedback zur Entwicklungszeit, ob die Logik korrekt ist
Viele Fremdsysteme	Integrationsrisiko	Annahmen zu den Fremdsystemen verifizieren, Unabhängigkeit beim Testen und zur Entwicklungszeit
Die Hauptaufgabe ist es, andere (unternehmensinterne) Komponenten/Services einzubinden und zu kombinieren	Orchestrierungsrisiko	Die orchestrierende Komponente isoliert testen, Vereinbarungen zwischen den Serviceanbietern sicherstellen
Bereitstellen einer Open-Source-Bibliothek	API-Design-Risiko	API aus Sicht der Verwender vorantreiben
Extrem hohe Benutzeranzahl (Last), performancekritische Prozesse, ...	Schwer erreichbare Qualitätsmerkmale	Frühzeitig mangelhafte Qualität erkennen
Unklar, ob es das richtige Produkt ist	Marktrisiko	Frühes End-User-Feedback einholen
Schwierige rollenübergreifende Zusammenarbeit, kein gemeinsames fachliches Verständnis	Risiko von Fehlkommunikation	Tests werden rollenübergreifend zusammen erstellt
Bedarf für technische Umbauten, hohe technische Schulden	Risiko beim Refactoring	Möglichst robuste Tests
...

Herangehensweise im Detail

allgemeingültig

Testprinzipien

→ geringer Aufwand, höhere
Präzision, schnelleres Feedback



Komponentengrenzen und Testschnitt

→ einfache Testbarkeit

individuell

Kontext

Fachlichkeit, Technik, Markt, Team...



Risiken und Testziele



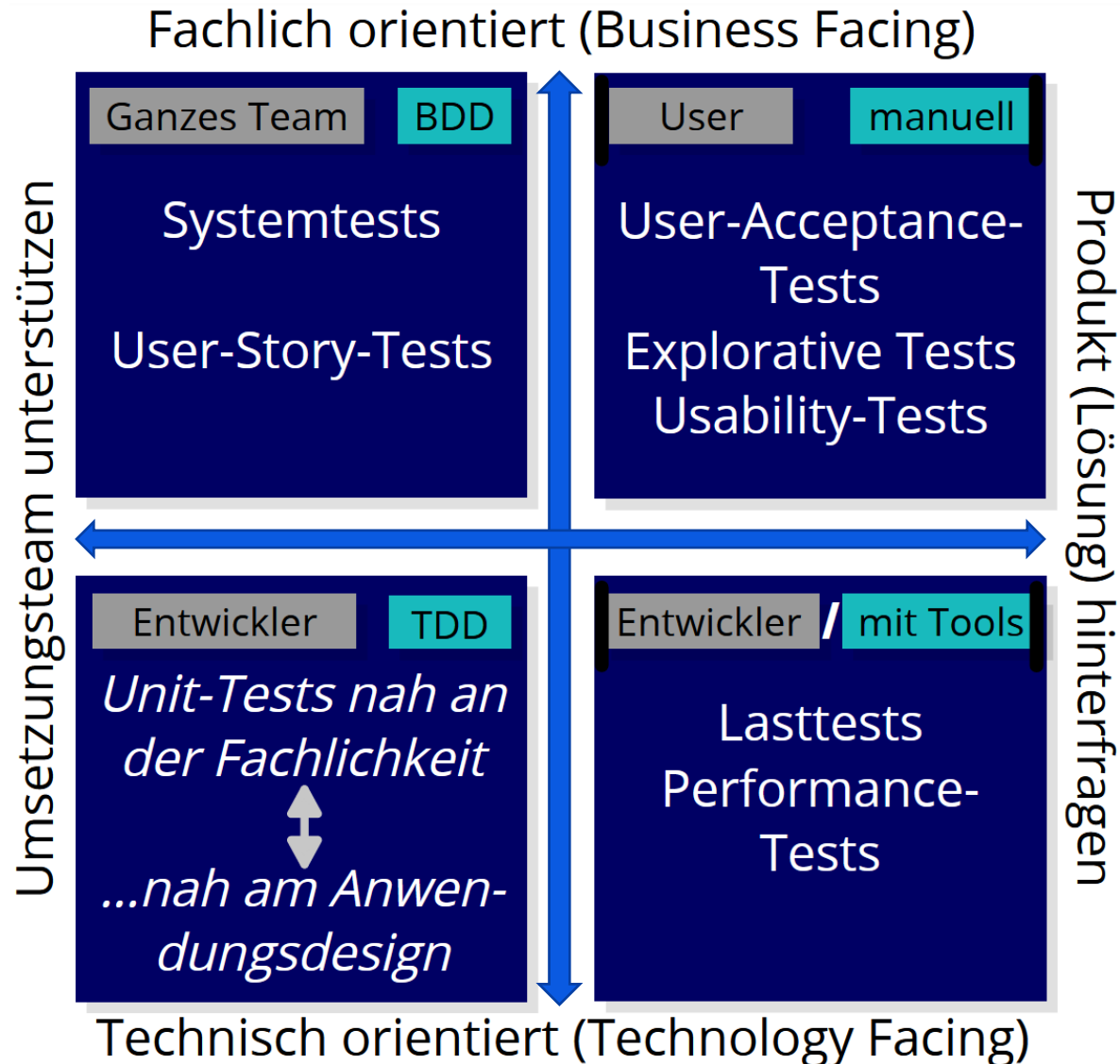
zielgerichtete Testarten und passende Testmethodik



Passende Tests

→ richtige Verteilung auf
passende Testarten

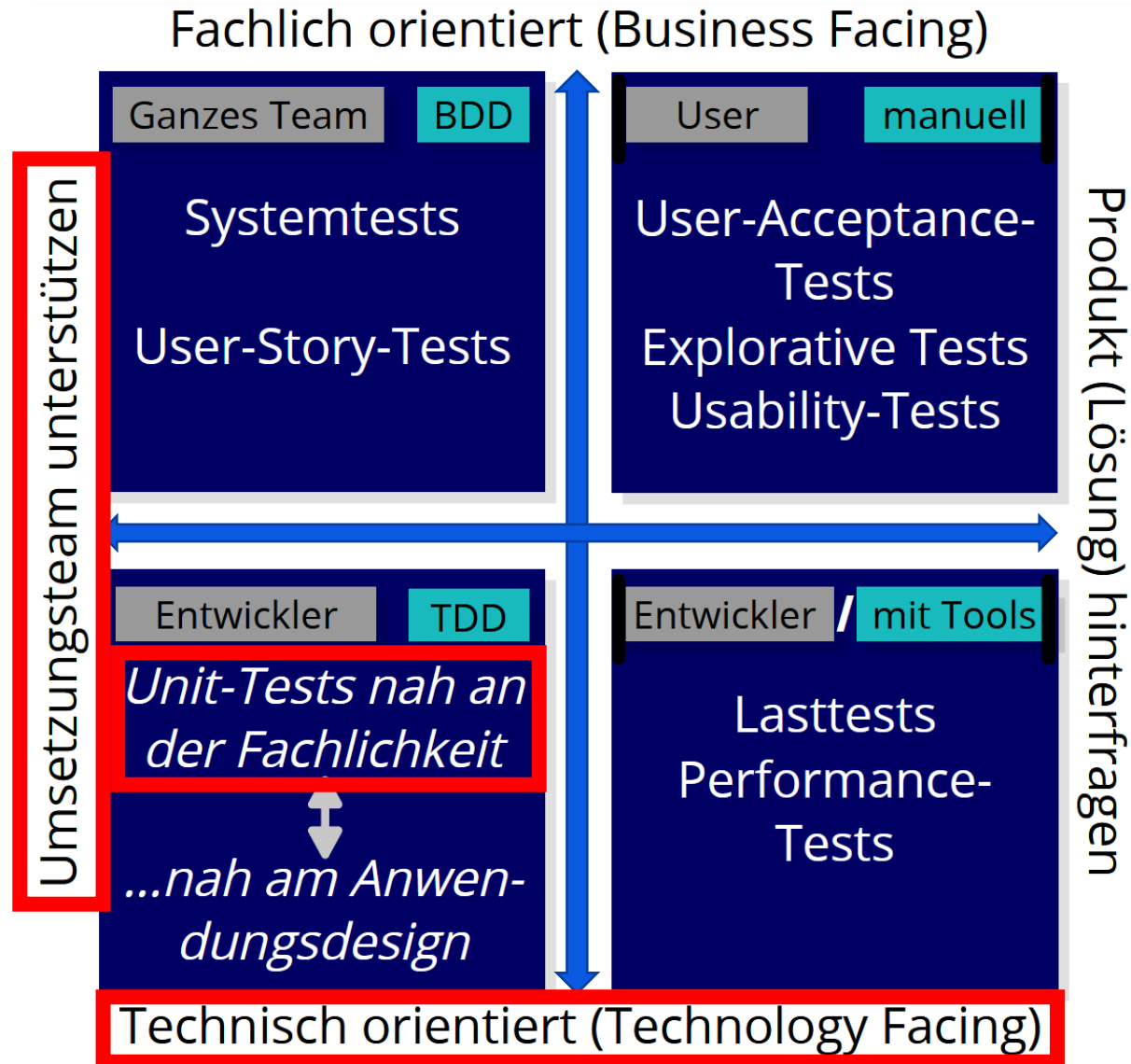
Wie kommen wir von Testzielen zu Testarten?



Agile Testquadranten

<https://lisacrispin.com/2011/11/08/using-the-agile-testing-quadrants/>

Beispiel: Passende Testart zum Testziel finden



Agile Testquadranten

<https://lisacrispin.com/2011/11/08/using-the-agile-testing-quadrants/>

Kontext

Bedarf für technische Umbauten, hohe technische Schulden

Risiko

Risiko beim Refactoring

Testziel

Möglichst robuste Tests

**Und dann nur noch
umsetzen 😊**

Herangehensweise im Detail

allgemeingültig

Testprinzipien

→ geringer Aufwand, höhere
Präzision, schnelleres Feedback



Komponentengrenzen und Testschnitt

→ einfache Testbarkeit

individuell

Kontext

Fachlichkeit, Technik, Markt, Team...



Risiken und Testziele



zielgerichtete Testarten und
passende Testmethodik

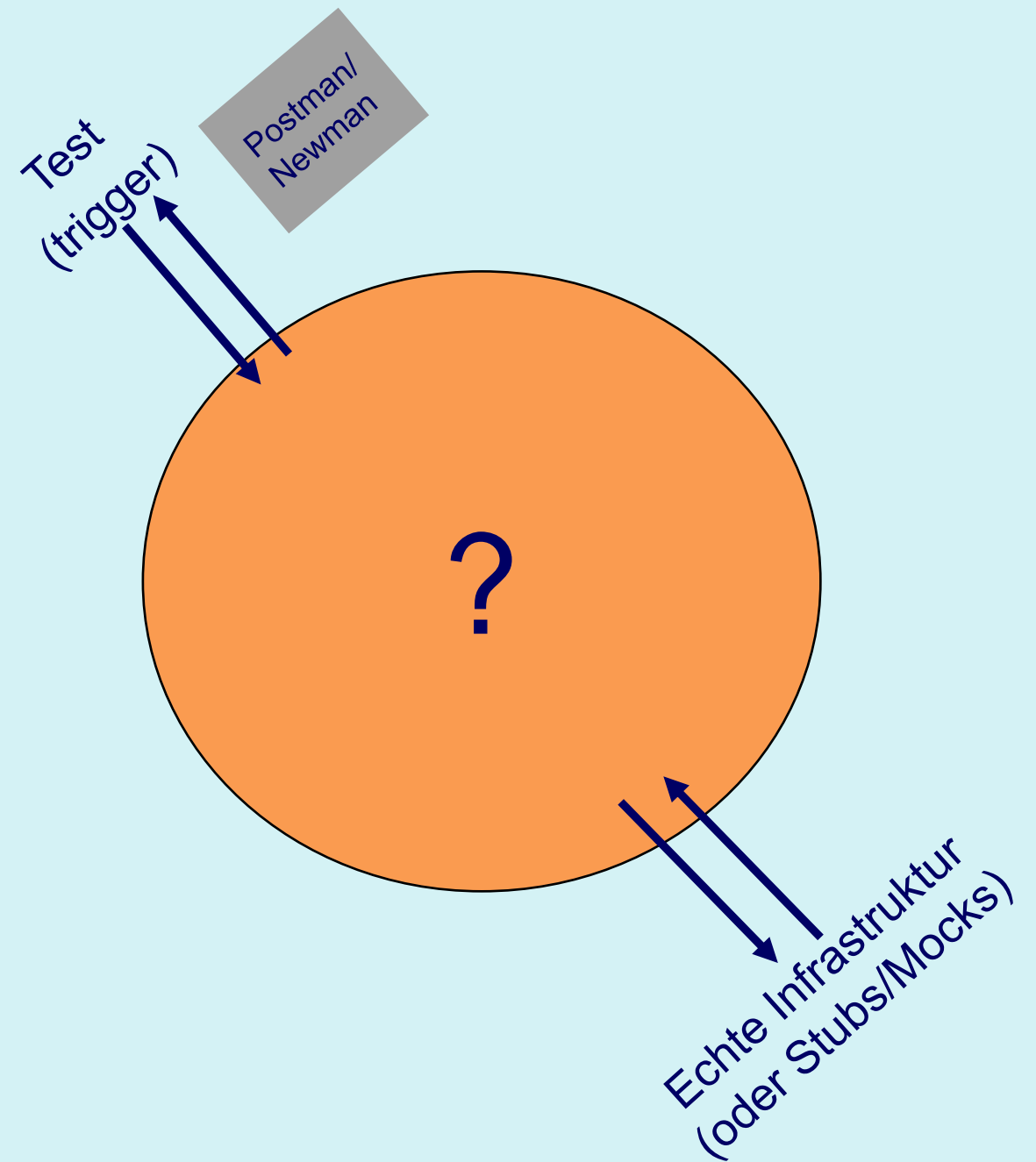


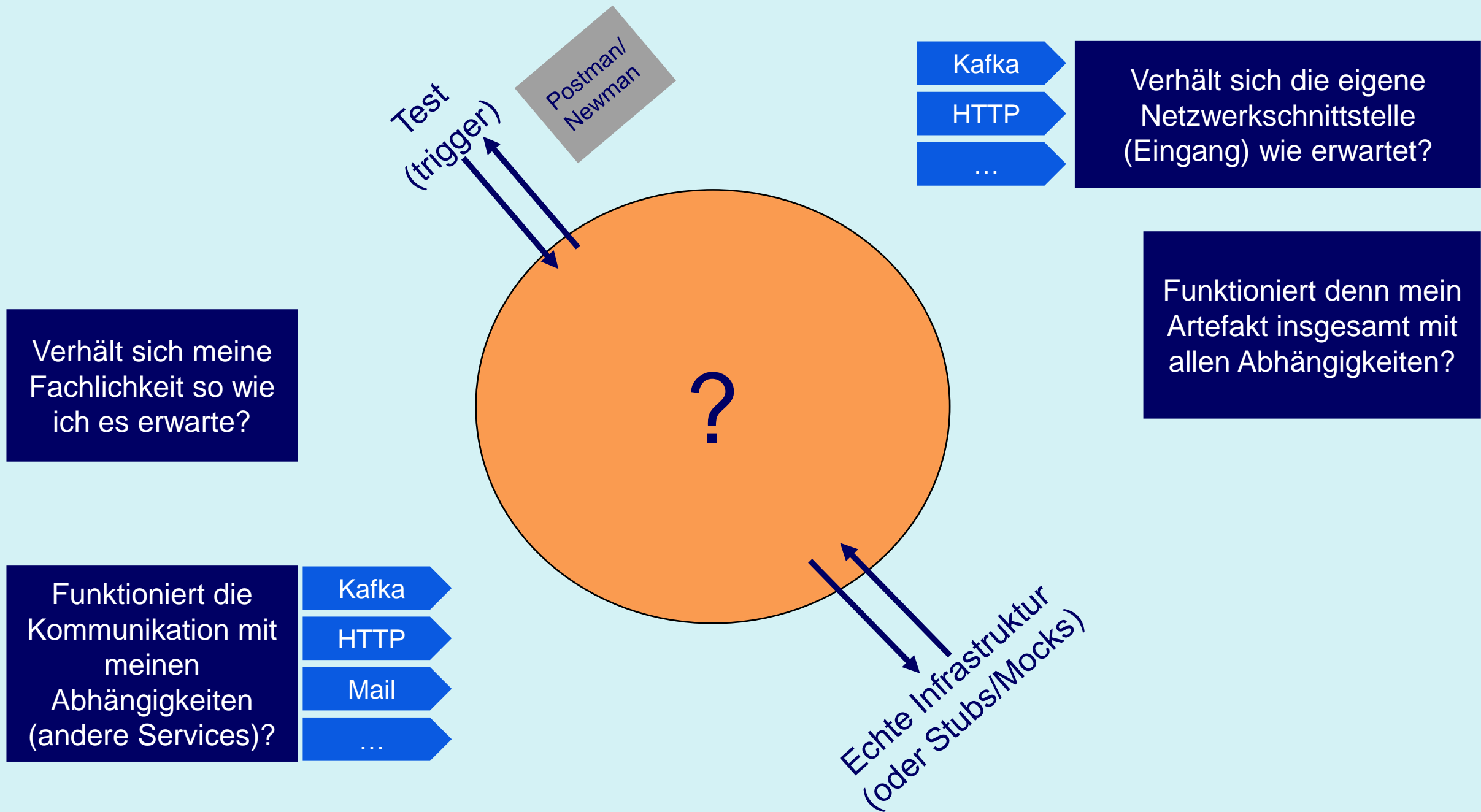
Passende Tests

→ richtige Verteilung auf
passende Testarten











































































































Wie adressieren wir hier unterschiedliche Testziele?

...irgendwo hier drin liegt unser HTTP-Controller
...und irgendwo unsere Domänenlogik
...und irgendwo unsere Persistenzlogik
...und irgendwo unsere Zugriffe auf Umsysteme

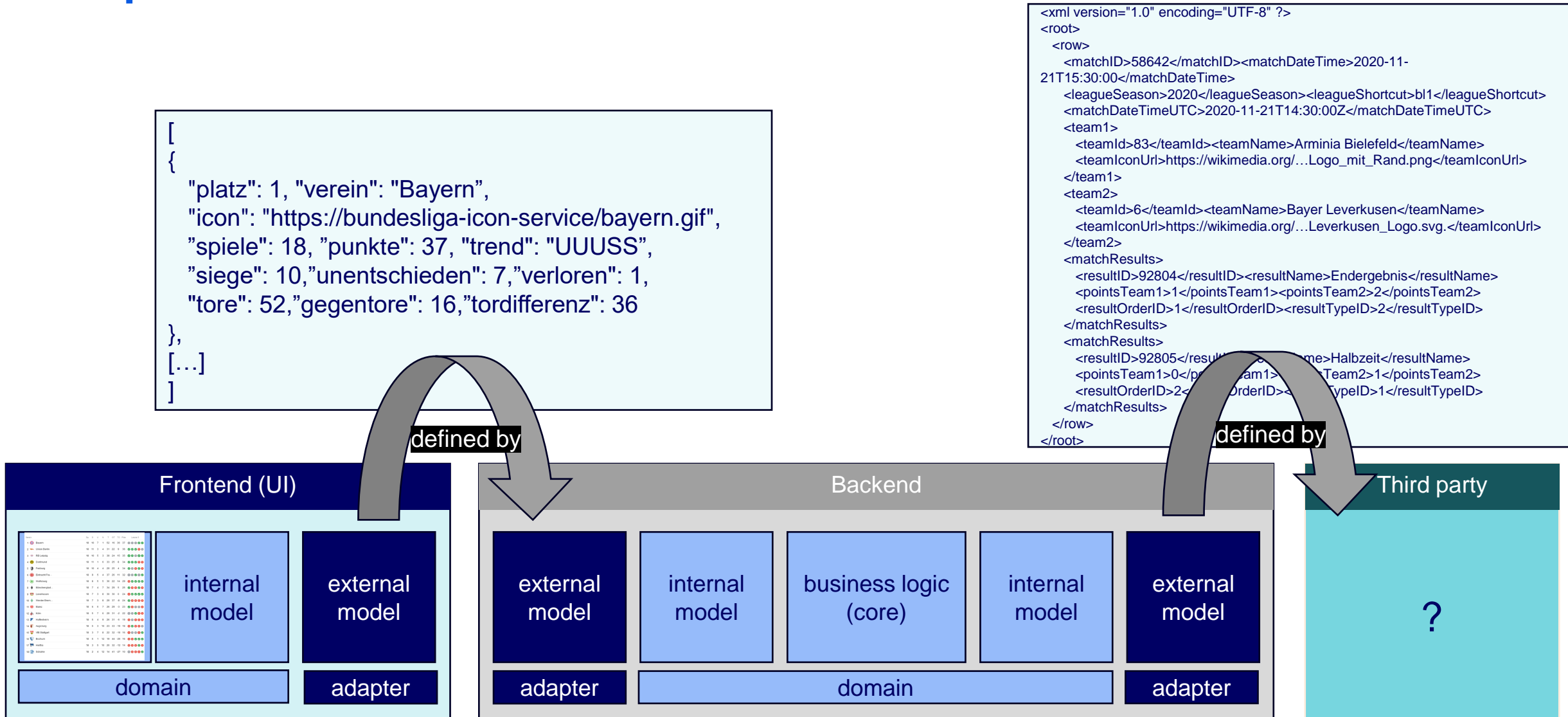




Beispiel: Wir basteln uns eine Bundesliga-Tabelle

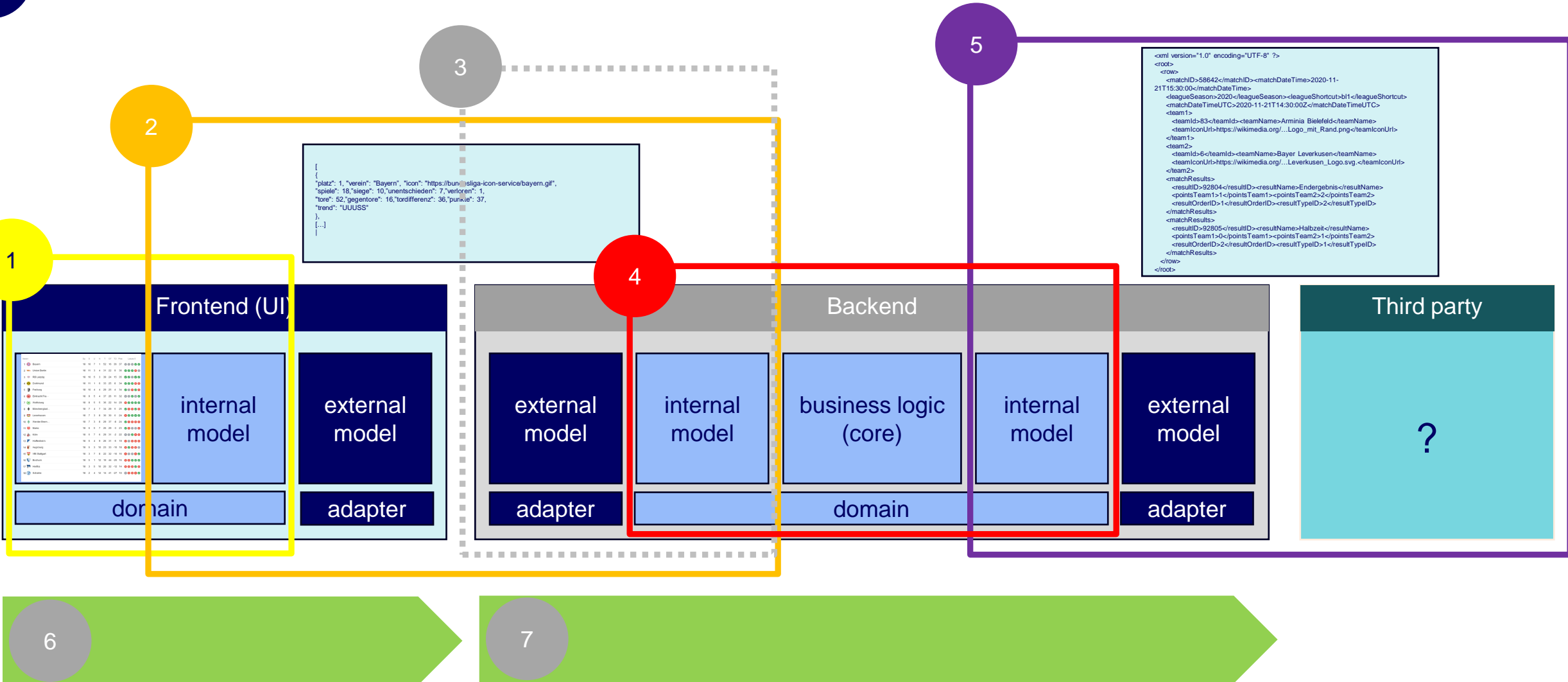
Verein	Sp	S	U	N	T	GT	TD	Pkte	Letzte 5
1  Bayern	18	10	7	1	52	16	36	37	    
2  Union Berlin	18	11	3	4	31	22	9	36	    
3  RB Leipzig	18	10	5	3	39	24	15	35	    
4  Dortmund	18	11	1	6	33	25	8	34	    
5  Freiburg	18	10	4	4	29	25	4	34	    
6  Eintracht Fra...	18	9	5	4	37	26	11	32	    
7  Wolfsburg	18	8	5	5	36	22	14	29	    
8  Mönchenglad...	18	7	4	7	34	29	5	25	    
9  Leverkusen	18	7	3	8	30	30	0	24	    
10  Werder Brem...	18	7	3	8	29	37	-8	24	    
11  Mainz	18	6	5	7	26	29	-3	23	    
12  Köln	18	5	7	6	29	31	-2	22	    
13  Hoffenheim	18	5	4	9	26	31	-5	19	    
14  Augsburg	18	5	3	10	23	33	-10	18	    
15  VfB Stuttgart	18	3	7	8	22	32	-10	16	    
16  Bochum	18	5	1	12	19	44	-25	16	    
17  Hertha	18	3	5	10	20	32	-12	14	    
18  Schalke	18	2	4	12	14	41	-27	10	    

Komponentenschnitt für eine einfache Testbarkeit



Testschnitt für Tests mit klarer Verantwortung

8



Und wie ist das jetzt mit dem Kontext?

Kontext und Risiken für das Projekt Bundesligatabelle

In welchem Kontext befindet sich das Produkt/Projekt?

- **Fachlichkeit:** Berechnung von Tabellen aufgrund von Spielergebnissen
- **Technik:**
 - BE: Java/Spring-Boot
 - FE: JS/Angular
- **Markt:** Freier Markt, kostenfreies Angebot
- **Team:** neu zusammengesetzt, zwei Java/Spring-Boot-Experten mit überschaubarem JS-Know-How, ein Fachexperte ohne Programmier-Know-How

Kontext und Risiken für das Projekt Bundesligatabelle

In welchem Kontext befindet sich das Produkt/Projekt?

- **Fachlichkeit:** Bei der Entwicklung von Spielereigenschaften und von Spielergebnissen
- **Technik:**
 - BE: Java/Spring-Boot
 - FE: JS/Angular
- **Markt:** Freier Markt, kostenfreies Angebot
- **Team:** neu zusammengesetzt, zwei Java/Spring-Boot-Experten mit überschaubarem JS-Know-How, ein Fachexperte ohne Programmier-Know-How

Risiko 1:

Fehlendes Know-How

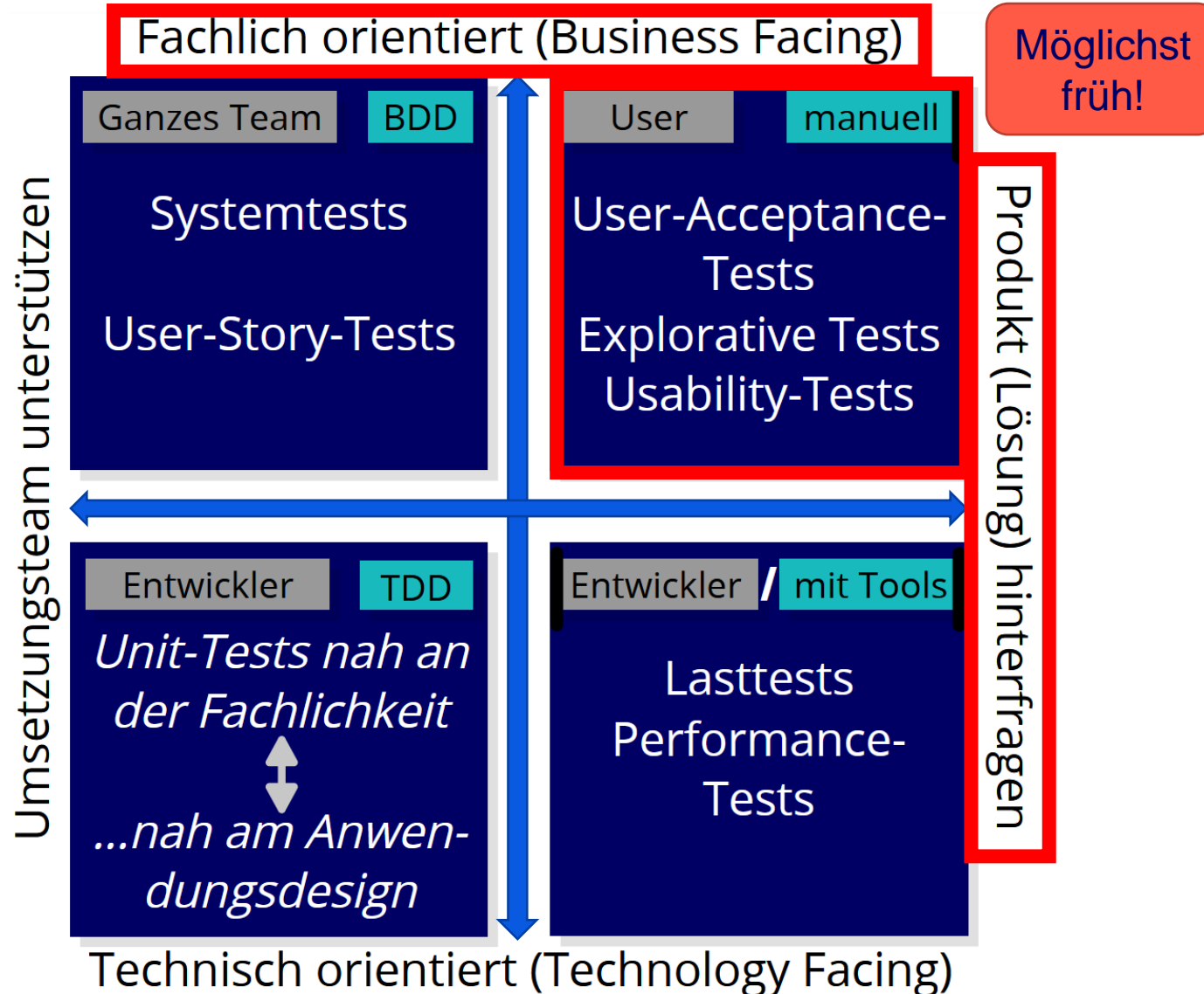
→ Frontendtests priorisieren

Risiko 2: Bedarf für Produkt?

→ Frühes End-User-Feedback

Risiko 3: mögliche Fehlkommunikation
→ rollenübergreifendes Testen

Passende Testarten für frühes End-User-Feedback



Agile Testquadranten

<https://lisacrispin.com/2011/11/08/sing-the-agile-testing-quadrants/>

Kontext

Unklar, ob es das richtige Produkt ist

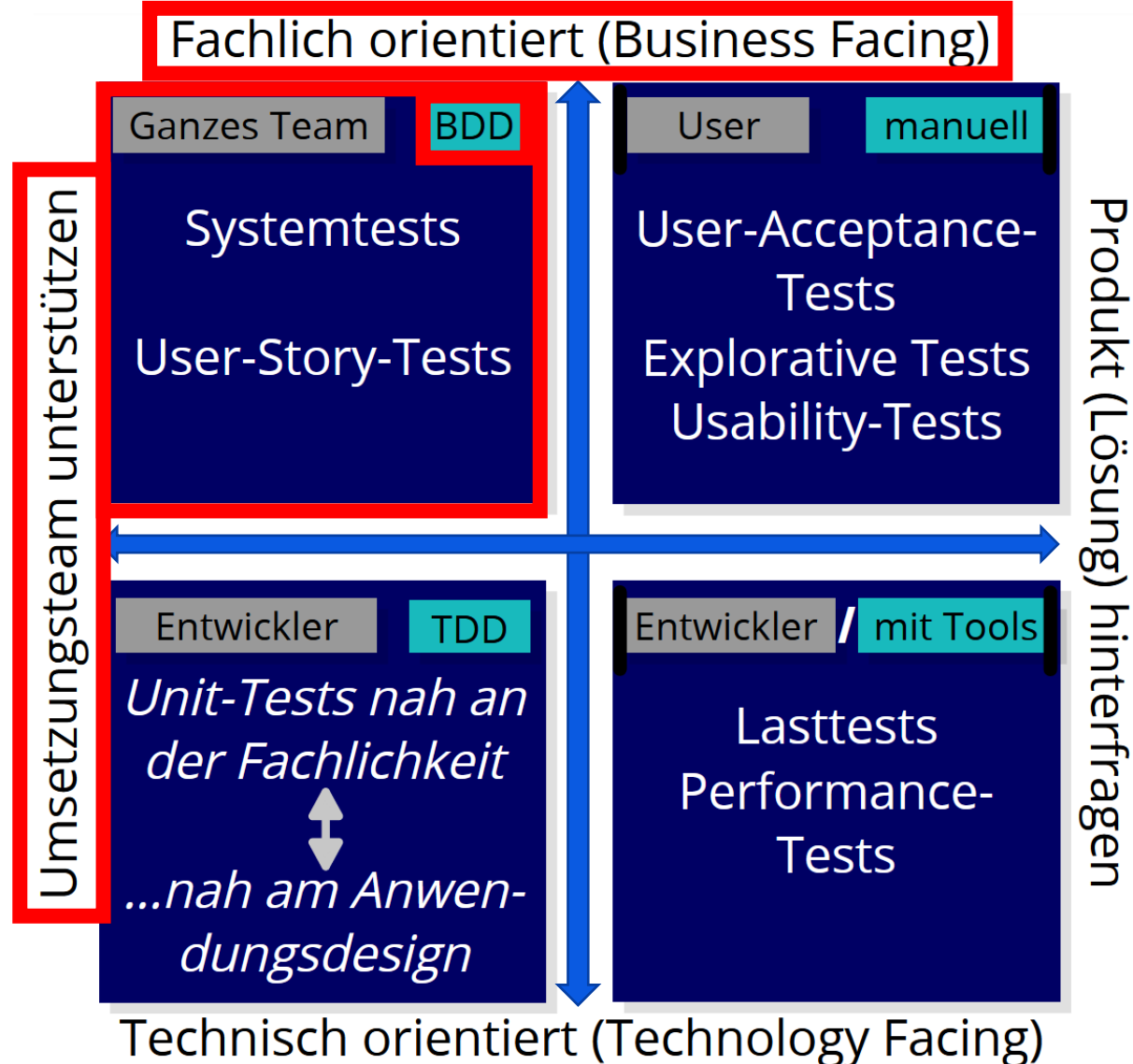
Risiko

Marktrisiko

Testziel

Frühes End-User-Feedback einholen

Passende Testarten für rollenübergreifendes Testen



Agile Testquadranten

<https://lisacrispin.com/2011/11/08/using-the-agile-testing-quadrants/>

Kontext

Schwierige rollenübergreifende Zusammenarbeit, kein gemeinsames fachliches Verständnis

Risiko

Risiko von Fehlkommunikation

Testziel

Tests werden rollenübergreifend zusammen erstellt

Sollten wir das dokumentieren?



Herangehensweise im Detail

allgemeingültig

Testprinzipien

→ geringer Aufwand, höhere
Präzision, schnelleres Feedback

Komponentengrenzen und Testschnitt

→ einfache Testbarkeit

individuell

Kontext

Fachlichkeit, Technik, Markt, Team...

Risiken und Testziele

zielgerichtete Testarten und passende Testmethodik

Passende Tests

→ richtige Verteilung auf
passende Testarten

Hilfreiche Dokumentation des tatsächlichen Testansatzes

- Ein Testansatz „lebt“ und **wandelt sich mit der Zeit**
 - Risiken ändern sich im Produktlebenszyklus
 - Technologien veralten und werden ausgetauscht
 - Neue Bibliotheken und Frameworks bieten neue Möglichkeiten (auch für den Test)
- Wie behalten wir den **Überblick**?
 - **Idee: Eine kompakte, visuelle Darstellung unserer Testarten**

Teststeckbrief

- Jede Testart erhält einen **Namen**
- Die wesentlichen Punkte* aus unserem Testansatz auf einen Blick:
 - Welche **Risiken und Testziele** adressiert der Test?
 - Was ist mein **System Under Test** (SUT) und was davon ist für den Test relevant?
 - Was deckt der Test ab, was lässt er bewusst außen vor?
 - Mit welcher **Testmethodik** und mit Hilfe welcher **Technologie(n)** wird der Test erstellt?
- Die exakte Darstellung kann von Team zu Team variieren
 - Wichtig ist: Jedes Teammitglied hat gleiches Verständnis der Begrifflichkeiten
 - Oft hilft bereits die **Diskussion**
 - Was haben wir bereits?
 - Was benötigen wir?

* die sich auf einzelne Tests beziehen

Teststeckbrief (Schablone)

Name der Testart	
WARUM (Testziele, adressierte Risiken)	
WAS (SUT, relevante Details)	WAS NICHT (irrelevant oder kein Teil des SUTs)
WIE (Technologien, Methodik)	

Teststeckbrief für Frontend-Tests im Projekt Bundesligatabelle

Name Testart: (1) Frontend-Tests

WARUM (Testziele, adressierte Risiken)

Technology Facing + Team unterstützen -> absichern (Regression);
Darstellungsprobleme und Browserinkompatibilitäten vermeiden

WAS

(SUT, relevante Details)

UI, Darstellung, Interaktion

WAS NICHT

(irrelevant / kein Teil des SUTs)

Fachliche Korrektheit der
angezeigten Daten;
Bezug der Daten vom Backend

WIE (Technologien, Methodik)

cypress;
Test Last

Teststeckbrief für User-Story-Tests im Projekt Bundesliga-Tabelle

Name Testart: (8) User-Story-Tests

WARUM (Testziele, adressierte Risiken)

Business Facing + Team unterstützen -> leiten und absichern;
Leiten: Fachlichkeit über Tests beschreiben und klären;
Sind alle Komponenten zusammen lauffähig (techn. Durchstich)?
Konfigurationsrisiko: Sind die Komponenten korrekt konfiguriert?

WAS

(SUT, relevante Details)

alle Komponenten im
Zusammenspiel sowie
dazugehörige Infrastruktur
(FE/BE/3d party)

WAS NICHT

(irrelevant / kein Teil des SUTs)

grundsätzlich nicht das "Was"
von (1) - (7), nur stichproben-
haft 1x pro User Story

WIE (Technologien, Methodik)

Karate
Behaviour Driven Development

Herangehensweise im Detail

allgemeingültig

Testprinzipien

→ geringer Aufwand, höhere
Präzision, schnelleres Feedback



Komponentengrenzen und Testschnitt

→ einfache Testbarkeit

individuell

Kontext

Fachlichkeit, Technik, Markt, Team...



Risiken und Testziele



zielgerichtete Testarten und passende Testmethodik



Passende Tests

→ richtige Verteilung auf
passende Testarten

Test Canvas

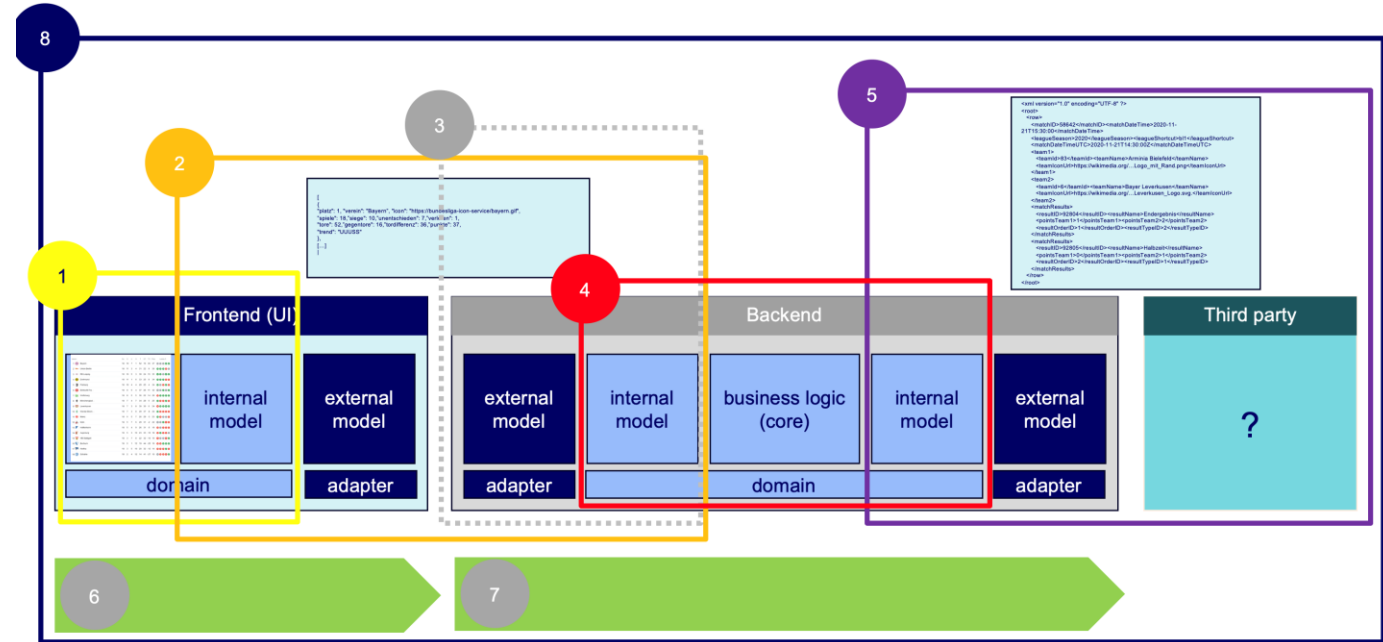
In welchem Kontext befindet sich das Produkt/Projekt?

- **Fachlichkeit:** Berechnung von Tabellen aufgrund von Spielergebnissen
- **Technik:**
 - BE: Java/Spring-Boot
 - FE: JS/Angular
- **Markt:** Freier Markt, kostenfreies Angebot
- **Team:** neu zusammengesetzt, zwei Java/Spring-Boot-Experten mit überschaubarem JS-Know-How, ein Fachexperte ohne Programmier-Know-How

Risiko 1

Risiko 2

Risiko 3



Test Canvas

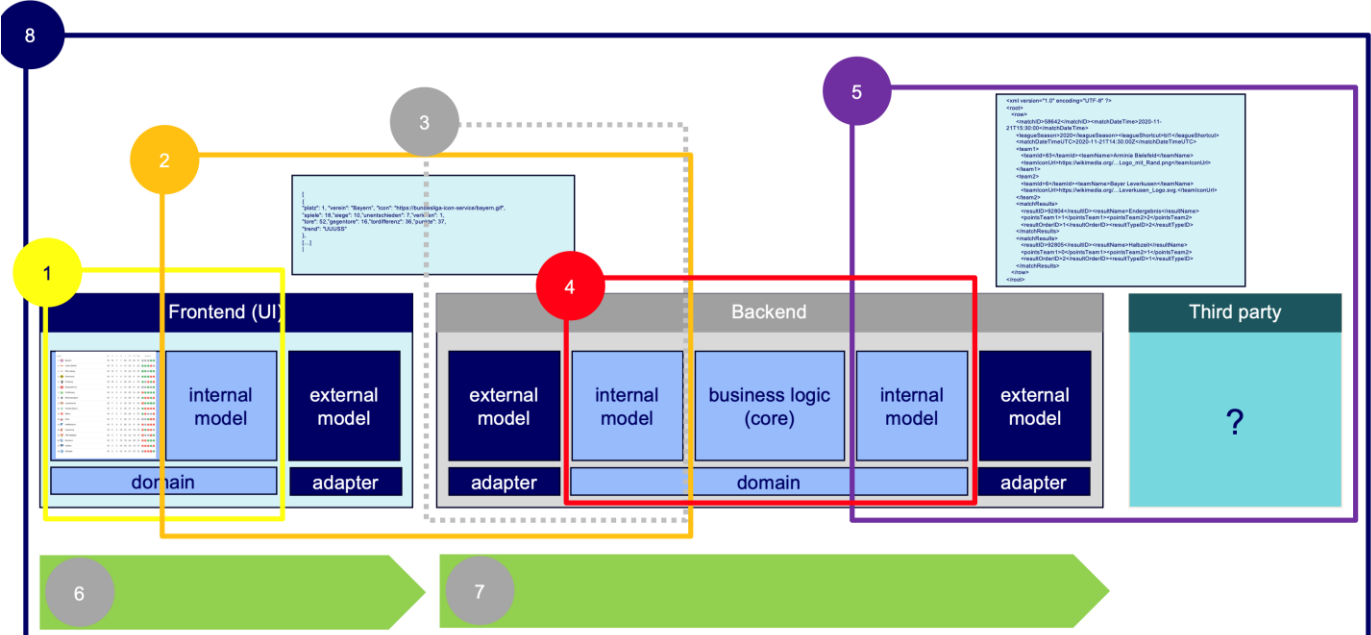
In welchem Kontext befindet sich das Produkt/Projekt?

- **Fachlichkeit:** Berechnung von Tabellen aufgrund von Spielergebnissen
- **Technik:**
 - BE: Java/Spring-Boot
 - FE: JS/Angular
- **Markt:** Freier Markt, kostenfreies Angebot
- **Team:** neu zusammengesetzt, zwei Java/Spring-Boot-Experten mit überschaubarem JS-Know-How, ein Fachexperte ohne Programmier-Know-How

Risiko 1

Risiko 2

Risiko 3



depre-
cated

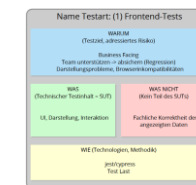
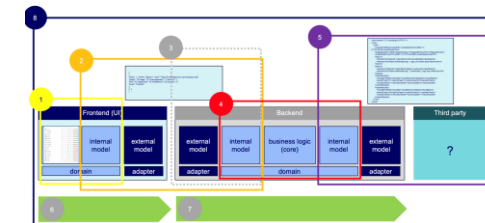
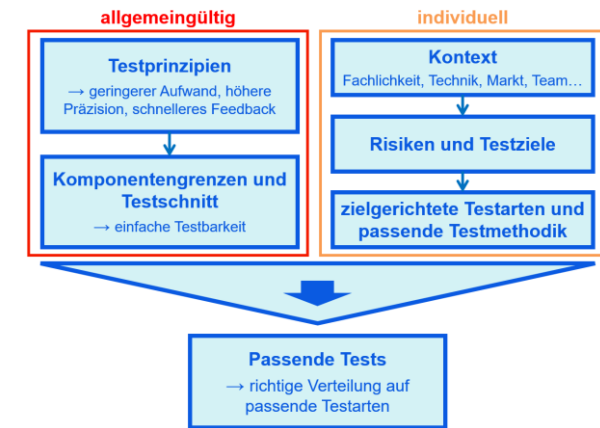


Überlappung mit 2

Testart für XY fehlt aus Zeitgründen

Fazit

- Modelle wie Testpyramide und Agile Testquadranten
 - hilfreich beim Einsatz im richtigen Kontext
 - in kleinen Schritten vorgehen, Prinzipien kombinieren
- Beachtung des individuellen Produkt-/Projektkontexts
 - wichtig, um entsprechend der Risiken den Fokus zu setzen
- Modularer Aufbau unserer Komponenten und Tests
 - zielgerichtet testen
- Test Canvas (u. a. mit Teststeckbriefen)
 - hilft, den Überblick zu bewahren
 - hilft für Diskussion und Weiterentwicklung des Testansatzes



Vielen Dank!