

# INTEGRATION TEST HELL

## ODER WIE INTEGRATIV SOLL ICH TESTEN?



IT-Consultant

Schwerpunkte

- Test-Driven Development
- Softwaredesign & Clean Code
- Software Craftsman  
@softwerkskammer

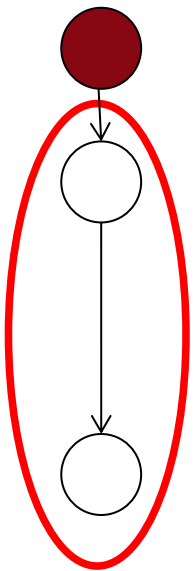


Twitter: @davidvoelkel

# WIE INTEGRATIV TESTEN?

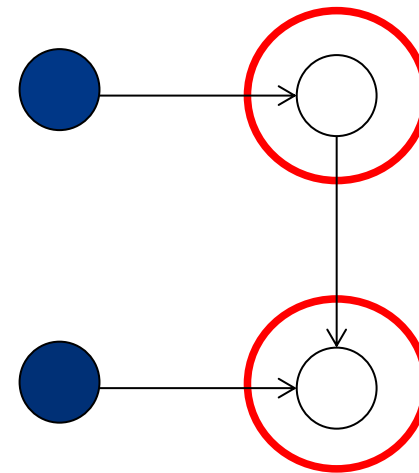
---

## Integrative Tests



TEUER

## Isolierte Tests



REALITÄTSFERN

# AGENDA

Isoliert vs. integriert

Testpyramide

Test Dekomposition

Fazit

Diskussion

# QUALITÄTSLEVEL?

---

$$\$(\text{Tests}) = \text{Nutzen} - \text{Kosten}$$

Senken von **Risiken**

Testauswahl

- 20% der Tests 80% Nutzen
- Zahl, Art, **Integrationsgrad**
- Risiko?
  - Domäne: Prototyp bis sicherheitskritisch
  - Verteilung in App

Tests an Risiken ausrichten!

# ISOLIERT ODER INTEGRIERT?

	Isolierende Tests	Integrierende Tests
Feedback	schnell	langsam
Fehlerfindung	einfach	schwer
Fragilität	gering	hoch
Kosten (inkl. Wartung)	billig	teuer
Aussagekraft Nutzer	gering	hoch

# TESTOBJEKT?

	Isolierende Tests	Integrierende Tests
Stärken	Logik, Komplexität, Breite	Integration, Datenflüsse, Tiefe
Beispiel Domäne	REPL	CRUD

Give me some Clojure:

```
> (+ 1 2)  
3  
>
```

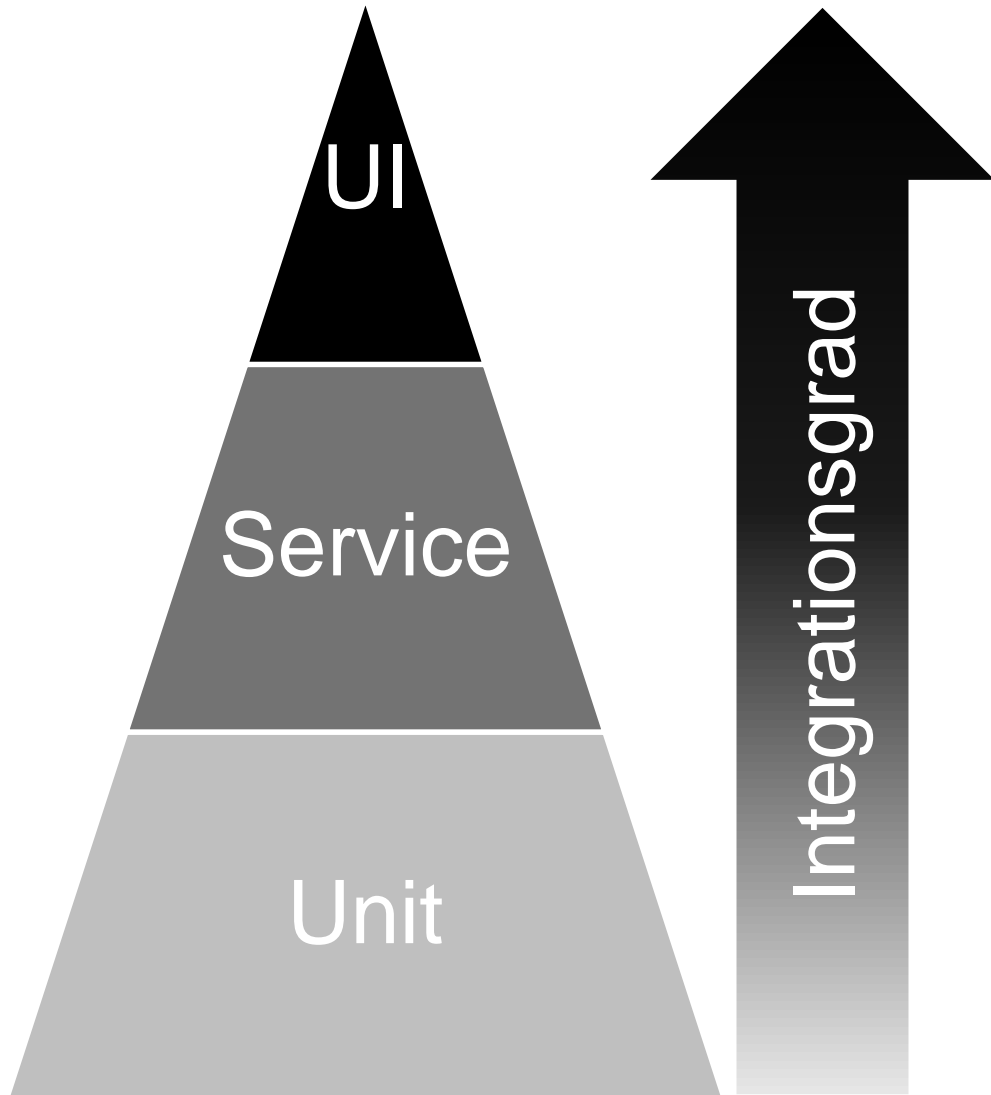
<http://tryclj.com/>

My Users				
	First Name	Last Name	Phone	Email
1	fname1	lname1	(000)000-0000	name1@gmail.com
2	fname2	lname2	(000)000-0000	name2@gmail.com
3	fname3	lname3	(000)000-0000	name3@gmail.com
4	fname4	lname4	(000)000-0000	name4@gmail.com
5	fname5	lname5	(000)000-0000	name5@gmail.com
6	fname6	lname6	(000)000-0000	name6@gmail.com
7	fname7	lname7	(000)000-0000	name7@gmail.com
8	fname8	lname8	(000)000-0000	name8@gmail.com

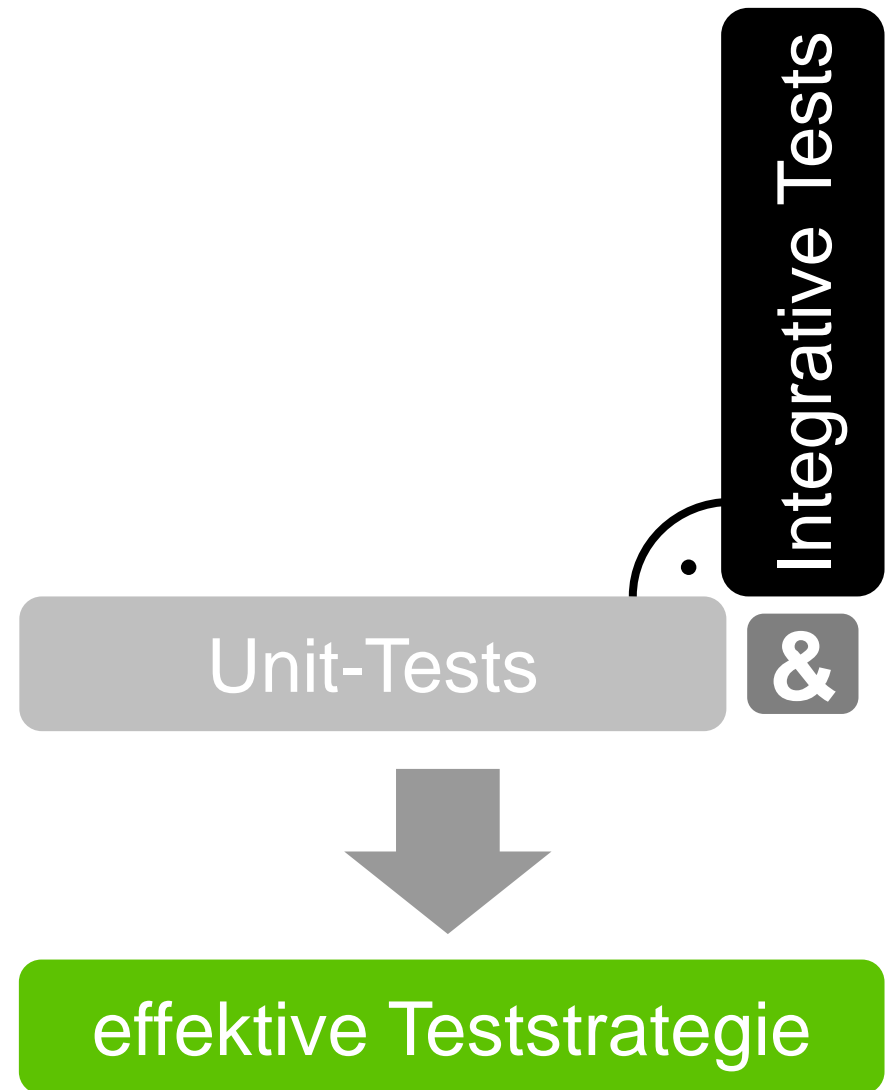
<http://www.jeasyui.com/tutorial/app/crud.php>



# LÖSUNG: TESTPYRAMIDE



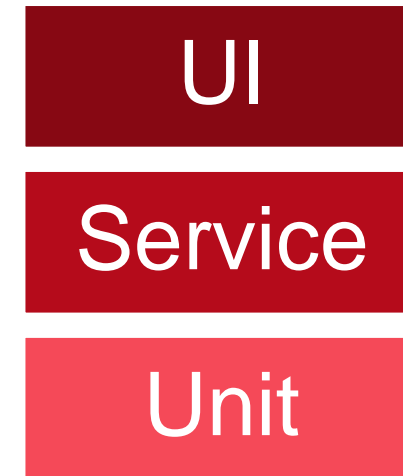
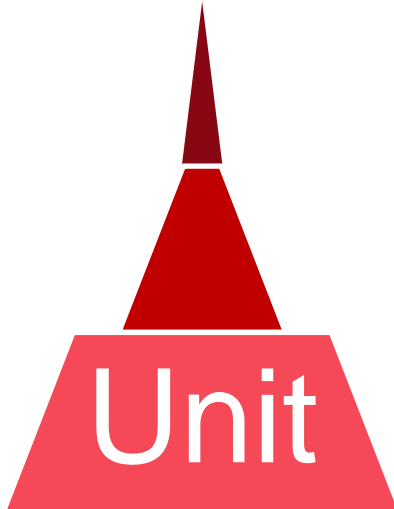
Pyramide nach Martin Fowler





# INTERPRETATIONEN

---



"Integrated Tests Are a Scam"

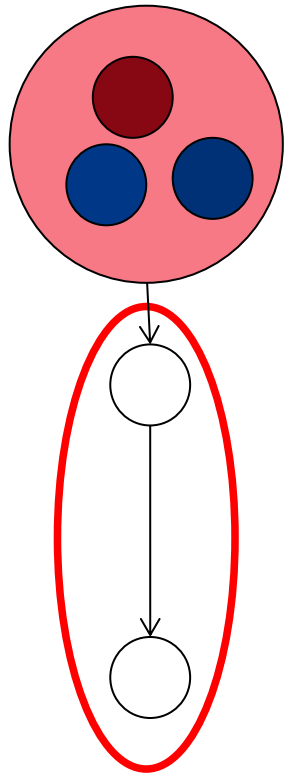
- J.B. Rainsberger

ATDD E2E

- z.B. GOOS, Continuous Delivery

# TEST DEKOMPOSITION

---



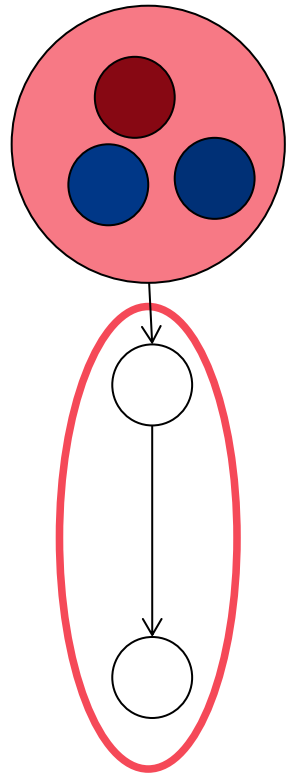
Problem überladene Integrationstests

– Vermischung Verantwortlichkeiten

– **Integration**

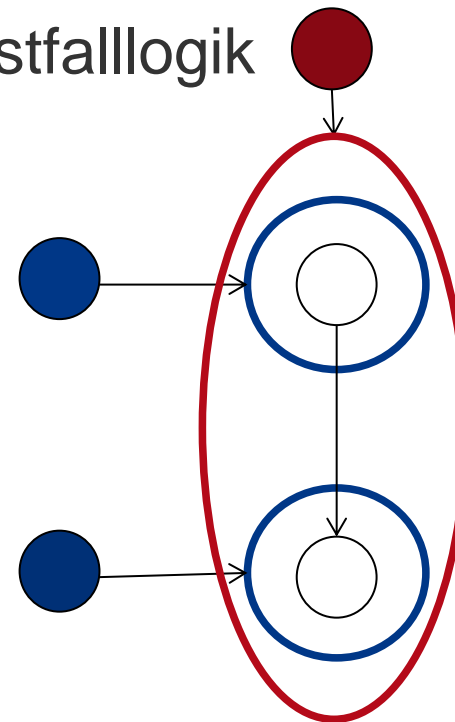
– **Logik** (ggf. auch verschiedene)

# TEST DEKOMPOSITION



– „**SRP für Tests**“

– Deduplizierung von Testfalllogik

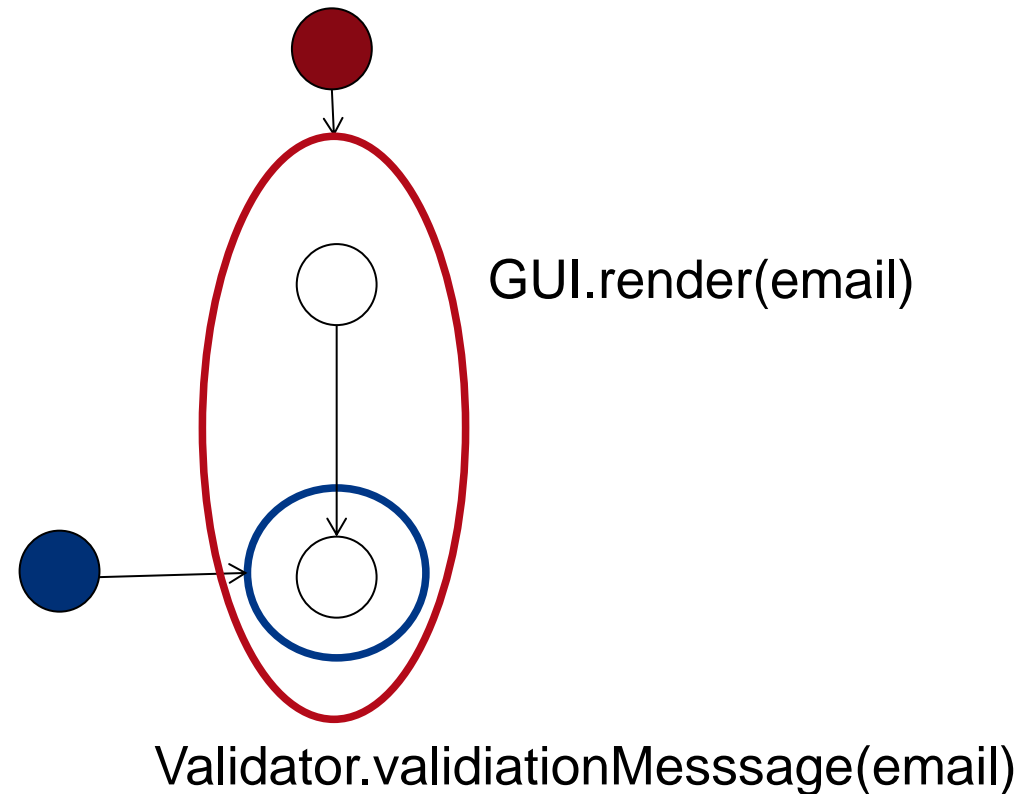


# BEISPIEL EMAIL-VALIDIERUNG

- integrativer Webtest
- Unittest

Eingabe	Ausgabe
„gueltige@email.de“	„Email OK“
„mail.ohne@tld“	„Email ungültig“
„mail.ohne.at.de“	„Email ungültig“
„@ohne-user.de“	„Email ungültig“
...	

E-Mail:  Email OK



# FACHLICHE DIMENSION

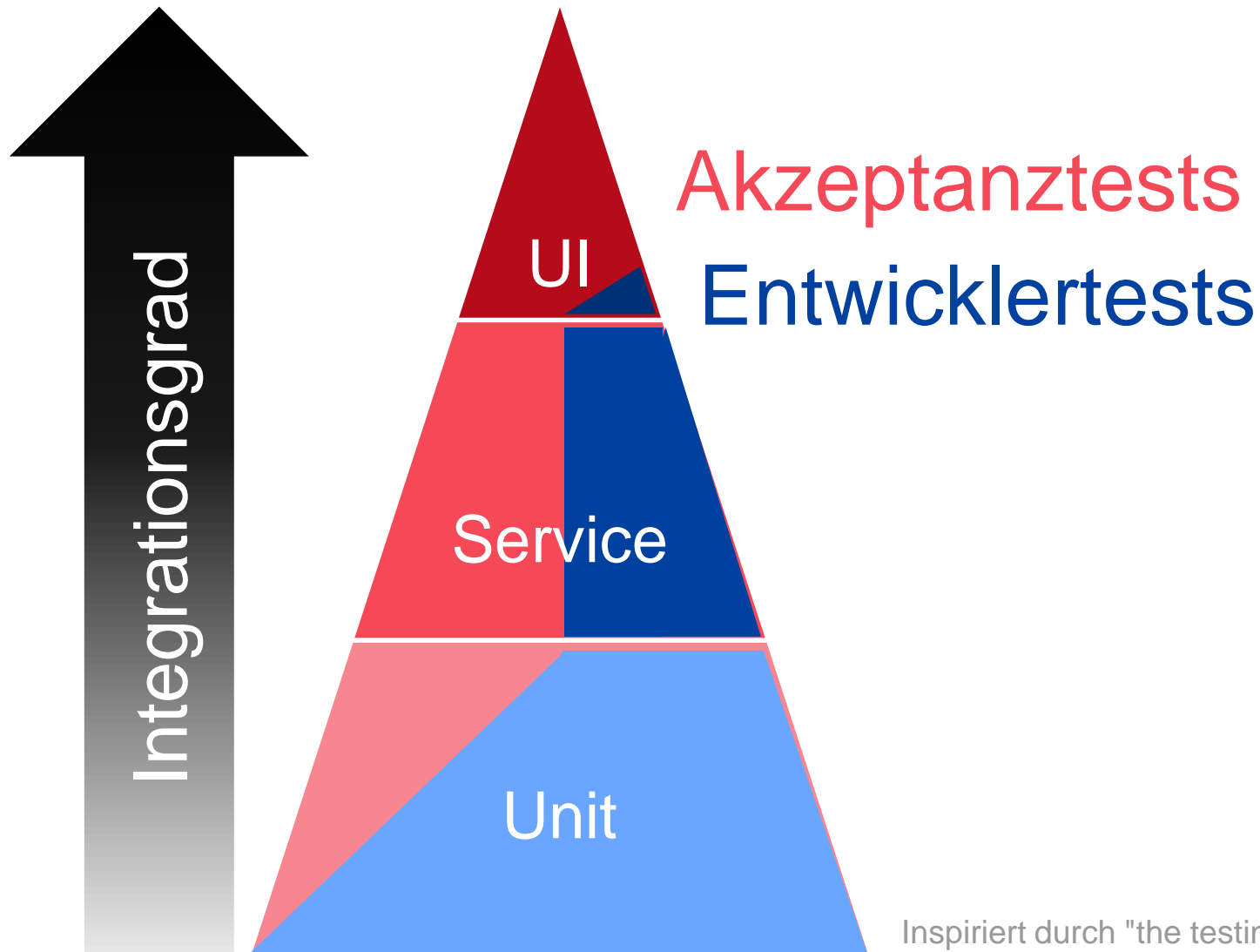
---

## Akzeptanztests vs. Entwicklertests

– fachlich

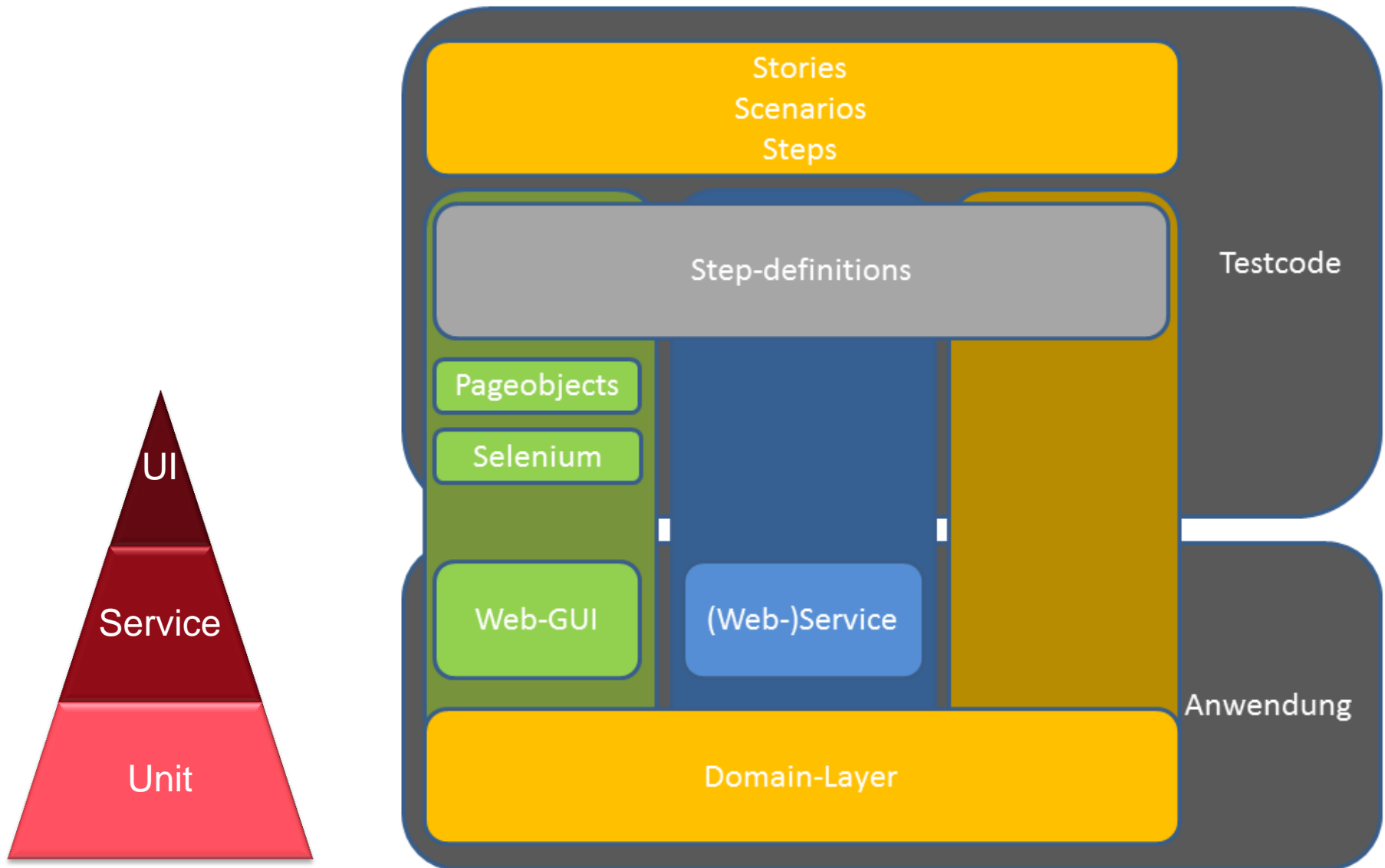
– technisch

# THE TESTING ICEBERG



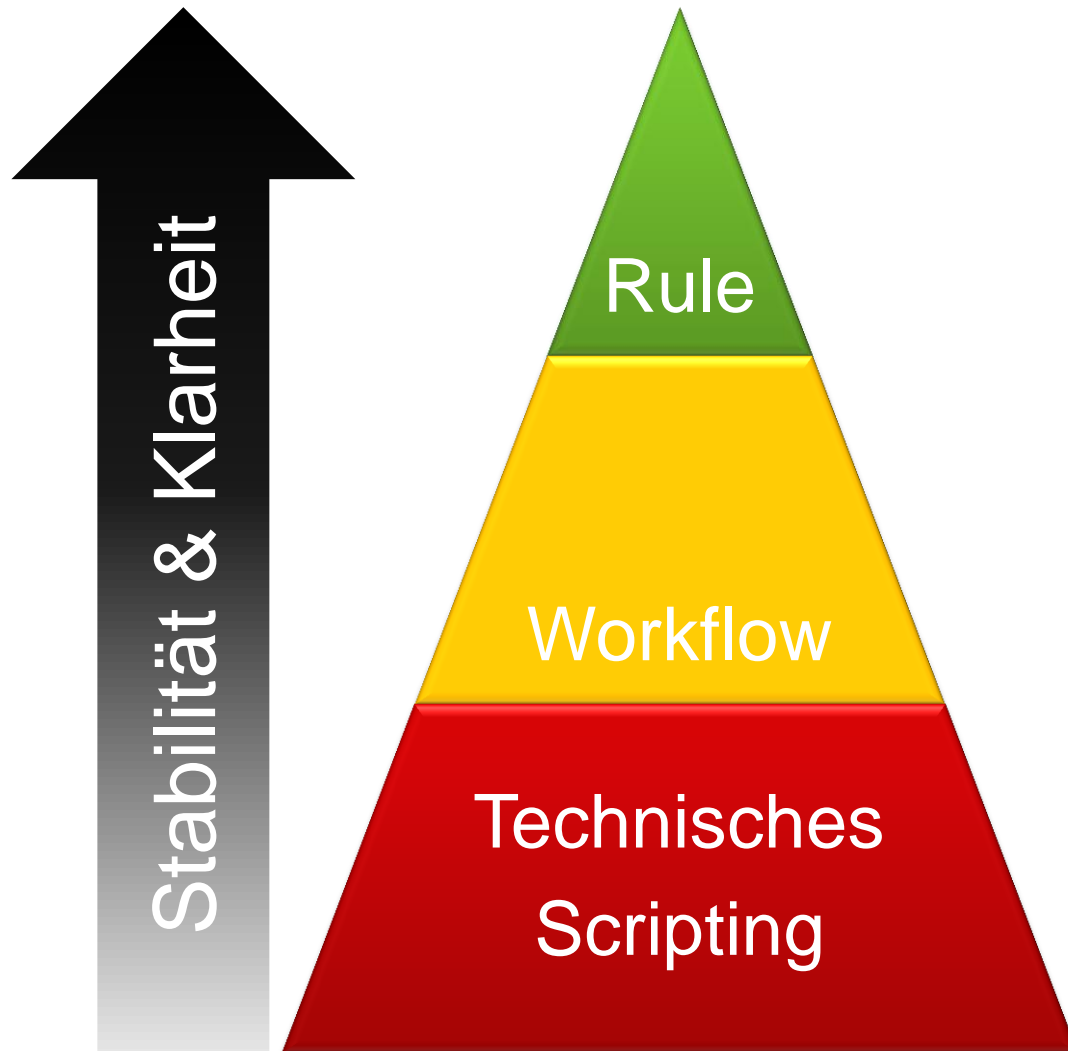
Inspiziert durch "the testing iceberg" von Seb Rose / Matt Wynne

# TEST-PYRAMIDE FÜR AKZEPTANZTEST



# GUI-TEST-LAYERING

---

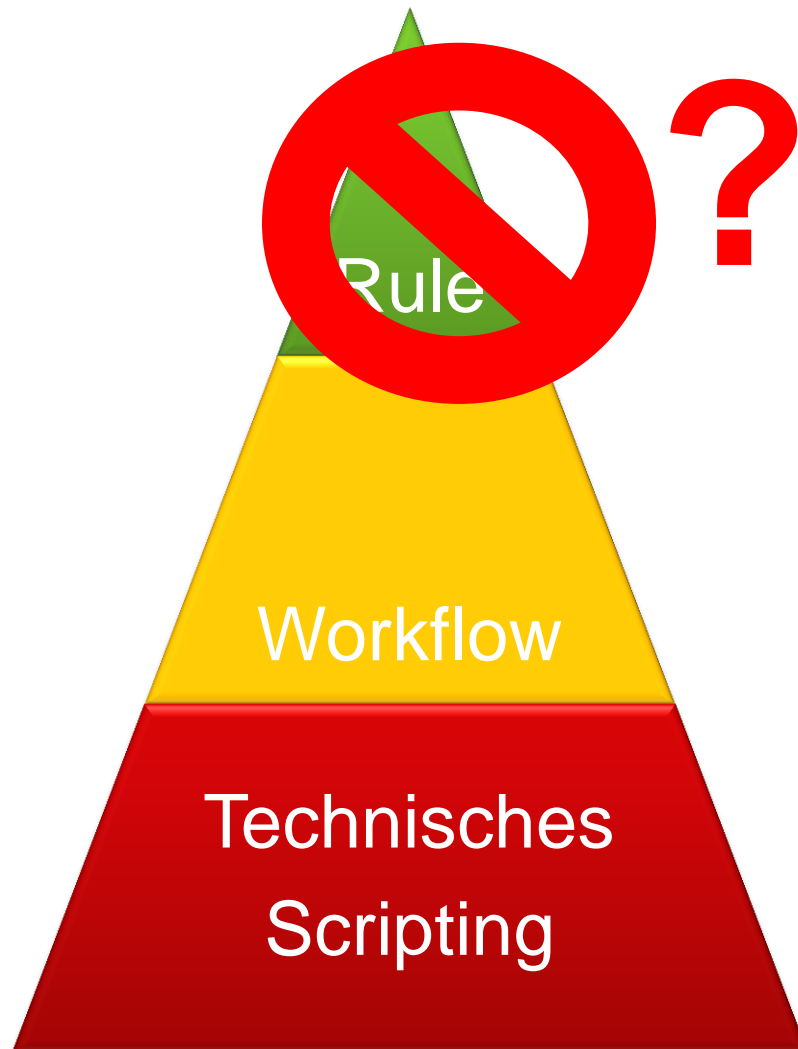


Gojko Adzics "UI testing without shooting yourself in the foot"



# RULE-TESTS GEGEN UI?

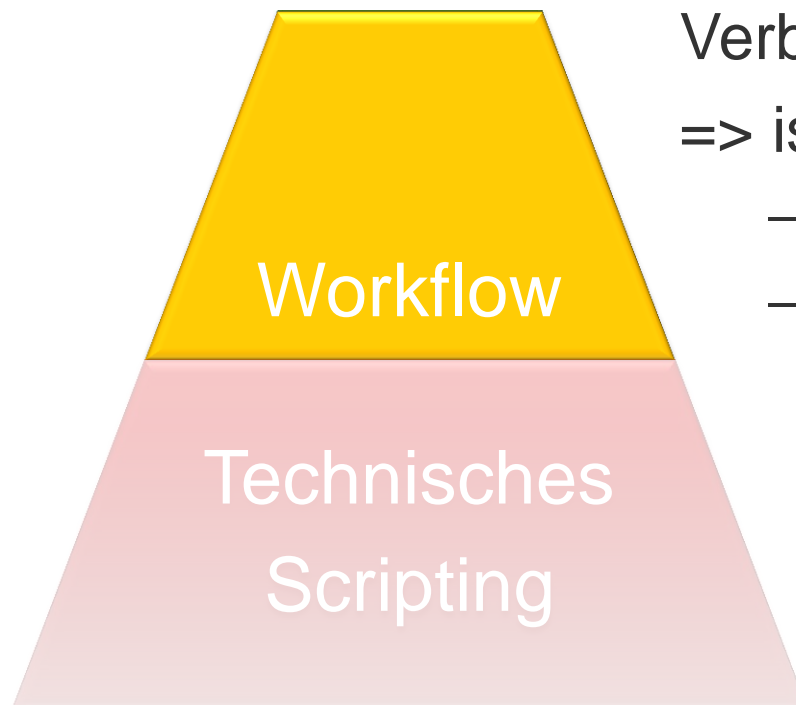
---



Gojko Adzics "UI testing without shooting yourself in the foot"

# ISOLIERTE GUI TESTS?

---



Verbleibende UI-Workflowtests

=> isolierte UI-Tests

- Subcutaneous Tests
- Backend-Mocking
  - Fat Client
  - Client-Side Javascript + REST

## Test Pyramide

- Orthogonal & spitz
  - Integrative Tests: Integration, Tiefe, einen je Integrationspunkt
  - Isolierte Tests: Logik, Komplexität, Breite
- Beschaffenheit d. Applikation
  - Art & Risiken

## Test Dekomposition

- Integrative Tests dekomponieren
- Rules-Tests raus aus UI
- Isolierte UI-Tests möglich?

# QUELLEN

---

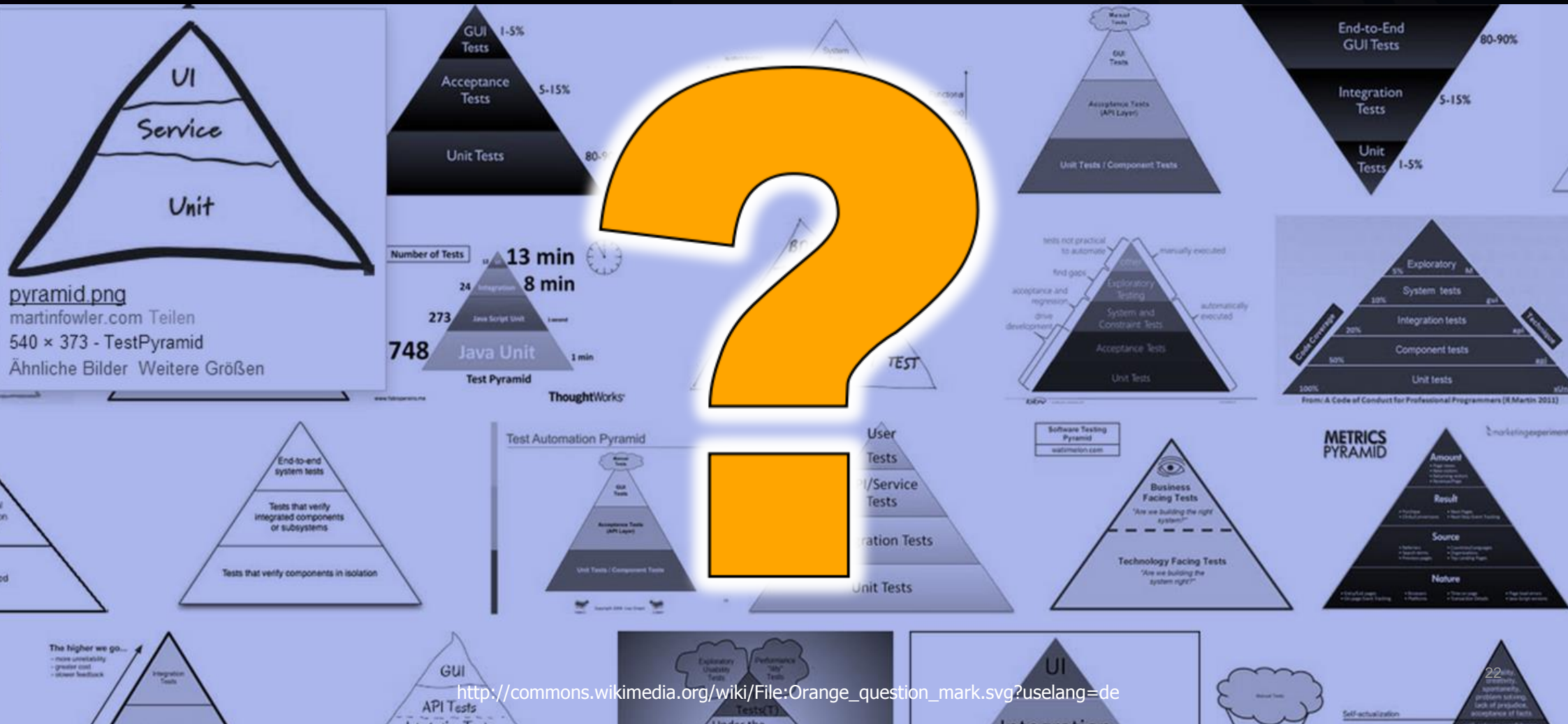
- Bücher
  - "Continuous Delivery" - Jez Humble, Dave Farley
  - "Growing Object-Oriented Software, Guided by Tests" - Steve Freeman, Nat Pryce
  - "Specification by Example" - Gojko Adzic
- GOOS Mailingliste
  - <https://groups.google.com/forum/#!forum/growing-object-oriented-software>
- "Integrated Tests are a scam" - J.B. Rainsberger
  - <http://www.jbrains.ca/permalink/using-integration-tests-mindfully-a-case-study>
  - <http://www.jbrains.ca/permalink/integrated-tests-are-a-scam-part-1>
- "The Testing Iceberg" - Matt Whyne und Seb Rose
  - <http://claysnow.co.uk/the-testing-iceberg/>

# QUELLEN

---

- End-to-end Tests?
  - <http://www.higherorderlogic.com/2010/01/responding-to-brian-marick>
- Testing-Pyramide im Kontext Continuous Delivery, James Crisp
  - <http://jamescrisp.org/2011/05/30/automated-testing-and-the-test-pyramid/>
- Wartbare Akzeptanztests, Jez Humble
  - <http://de.slideshare.net/jezhumble/creating-maintainable-automated-acceptance-tests>
  - <http://skillsmatter.com/podcast/agile-testing/the-long-term-value-of-acceptance-tests>
- „The Three Amigos“, George Dinwiddie
  - [http://www.nxtbook.com/nxtbooks/sqe/bettersoftware\\_1111/#/19](http://www.nxtbook.com/nxtbooks/sqe/bettersoftware_1111/#/19)

# FRAGEN UND DISKUSSION



The image is a collage of various test pyramid diagrams and a central orange question mark. The diagrams represent different testing strategies and their components:

- pyramid.png** (top left): A pyramid divided into three horizontal sections labeled "UI", "Service", and "Unit".
- ThoughtWorks Test Pyramid** (top middle): A pyramid with three levels: "Unit Tests" (80-90%), "Acceptance Tests" (5-15%), and "GUI Tests" (1-5%).
- Test Automation Pyramid** (middle left): A pyramid with four levels: "Unit Tests / Component Tests" (1 min), "Java Unit" (748 tests), "Integration" (24 tests, 8 min), and "System" (13 min).
- Software Testing Pyramid** (middle right): A pyramid with four levels: "Unit Tests", "Acceptance Tests", "System and Constraint Tests", and "Exploratory Testing".
- METRICS PYRAMID** (bottom right): A pyramid with four levels: "Amount", "Result", "Source", and "Nature".
- Test Automation Pyramid** (bottom middle): A pyramid with four levels: "Unit Tests / Component Tests", "Acceptance Tests (API Layer)", "API/Service Tests", and "User Tests".
- Business Facing Tests** (bottom right): A pyramid with two levels: "Business Facing Tests" and "Technology Facing Tests".
- Integration Tests** (bottom left): A pyramid with two levels: "Integration Tests" and "Unit Tests".
- End-to-End GUI Tests** (top right): A pyramid with three levels: "Unit Tests" (1-5%), "Integration Tests" (5-15%), and "End-to-End GUI Tests" (80-90%).
- Exploratory Testing** (middle right): A pyramid with four levels: "Unit tests" (100%), "Component tests" (30%), "Integration tests" (20%), "System tests" (10%), and "Exploratory" (5%).
- Self-actualization** (bottom right): A pyramid with two levels: "Self-actualization" and "Acceptance of tests".

A large orange question mark is centered in the image, with the word "TEST" written below it.

[http://commons.wikimedia.org/wiki/File:Orange\\_question\\_mark.svg?uselang=de](http://commons.wikimedia.org/wiki/File:Orange_question_mark.svg?uselang=de)

# LIZENZ

---

- Attribution-ShareAlike 3.0 Germany
- <http://creativecommons.org/licenses/by-sa/3.0/de/>

