Ziele beim Testen

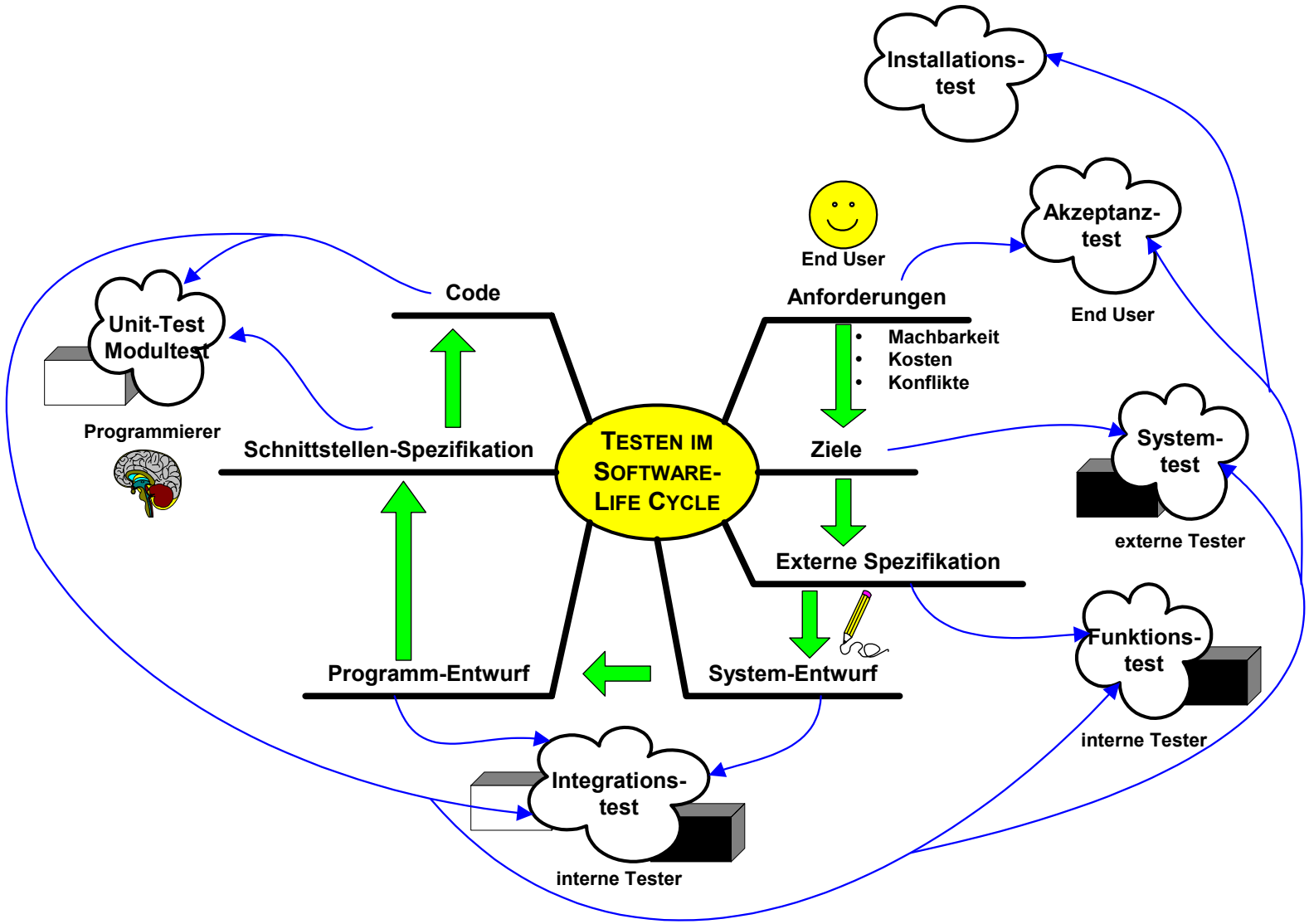
- Programme ohne „Bugs“
- Testbare Programme
  - validierbar oder falsifizierbar
  - wartbar
- Anforderungen (Requirements)
  - explizit oder implizit,
  - implementierbar, vollständig, konsistent
- Explizite Anforderungen validieren
  - durch „clean test“, konstruktive/positive Tätigkeit
- Implizite Anforderungen falsifizieren
  - durch „dirty tests“, subversive/negative Tätigkeit
  - z.B. das Programm zum Absturz bringen
  - Benötigt wesentlich mehr Testfälle

SMART Criteria:
- Specific
- Measurable
- Achievable
- Realistic
- Timely

C^3 Criteria:
- Correct
- Complete
- Consistent

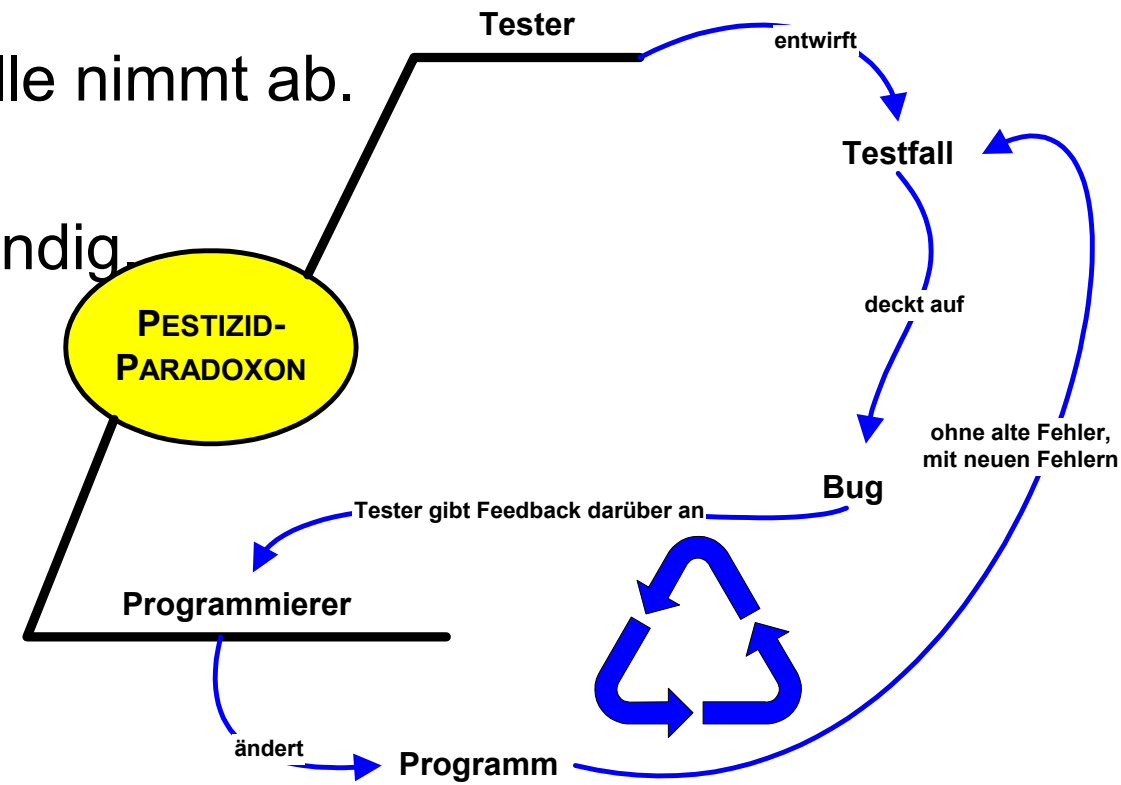
Möglichst früh im Software-Lebenszyklus

Testen im Software-Lebenszyklus



Pestizid-Paradoxon

- Jede Methode, die man anwendet, um Fehler zu vermeiden oder zu finden, hinterlässt einen gewissen Rest von raffinierteren Fehlern, gegen welche die eingesetzte Methode nichts ausrichten kann.
- Die Effektivität der Testfälle nimmt ab.
- Neue Testfälle für
  neue Bugs werden notwendig.



Orakel-Problem

- Es ist möglich, automatisch Testdaten zu entwerfen. Es ist aber nicht möglich, automatisch Testfälle zu entwerfen.
- Ein Testfall besteht aus:
  - Eingabedaten
  - erwarteten Ausgabedaten
- Für die automatische Generierung der erwarteten Ausgabedaten braucht man ein **Orakel**:
  - es muss die erwartete Ausgabe vorhersagen
  - ansonsten bräuchte man ein (automatisch generiertes) Programm mit derselben Funktionalität wie das zu testende Programm
- Häufig eingesetzte Orakel: menschlicher Experte
  - validiert die Ausgabeergebnisse oder
  - sagt Ausgabewerte voraus

Properties of good tests

- A-TRIP
  - **Automatic** (in at least two ways: invoking the tests and checking the results)
  - **Thorough** (test everything that's likely to break; bugs are not evenly distributed but clump together in problematic areas)
  - **Repeatable** (each test should produce the same results every time)
  - **Independent** (keep the tests tightly focused, and independent from the environment and each other)
  - **Professional** (tests must be written and maintained to the same professional standards as your production code, e.g. Encapsulation, low coupling, no duplication)

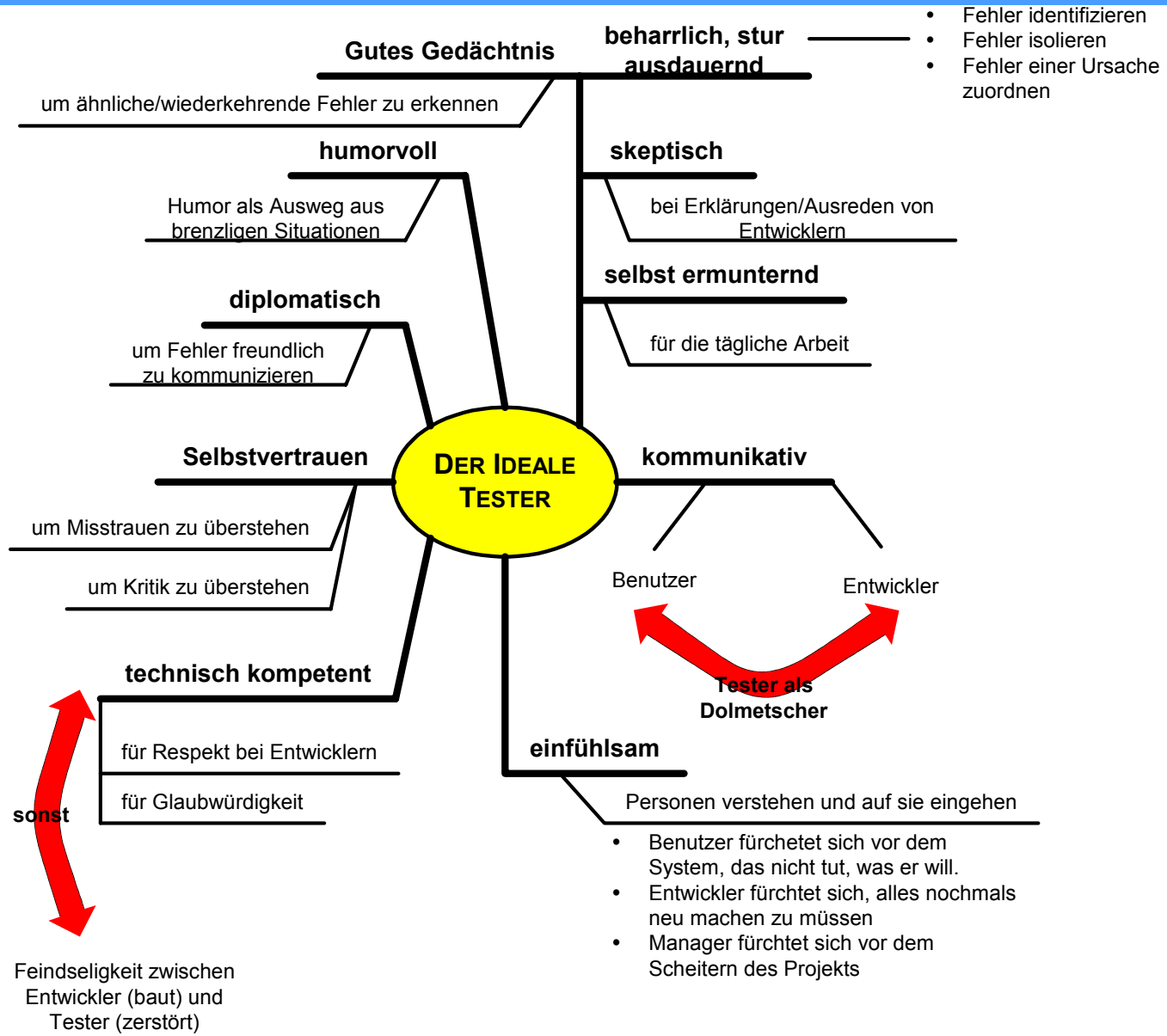
Testprinzipien (1)

- Ein Testfall muss das erwartete Ergebnis spezifizieren.
  - Sonst akzeptiert man plausible aber falsche Resultate.
  - Sonst hat man nicht verstanden, was passieren soll oder wie etwas erledigt werden soll.
    - Ergebnisse alter Testläufe als erwartete Ergebnisse heranziehen
    - Kommerziell erhältliche „Test Suites“ verwenden
    - Ergebnisse mit denen anderer (verlässlicher) Programme vergleichen
- Schau dir die Ergebnisse des Tests genau an.
  - Das Auge sieht, was es sehen möchte.
- Teste neben gültigen und erwarteten auch ungültige und unerwartete Eingaben
  - Fehler werden sonst erst entdeckt, wenn das Programm auf neue, unerwartete Weise eingesetzt wird.

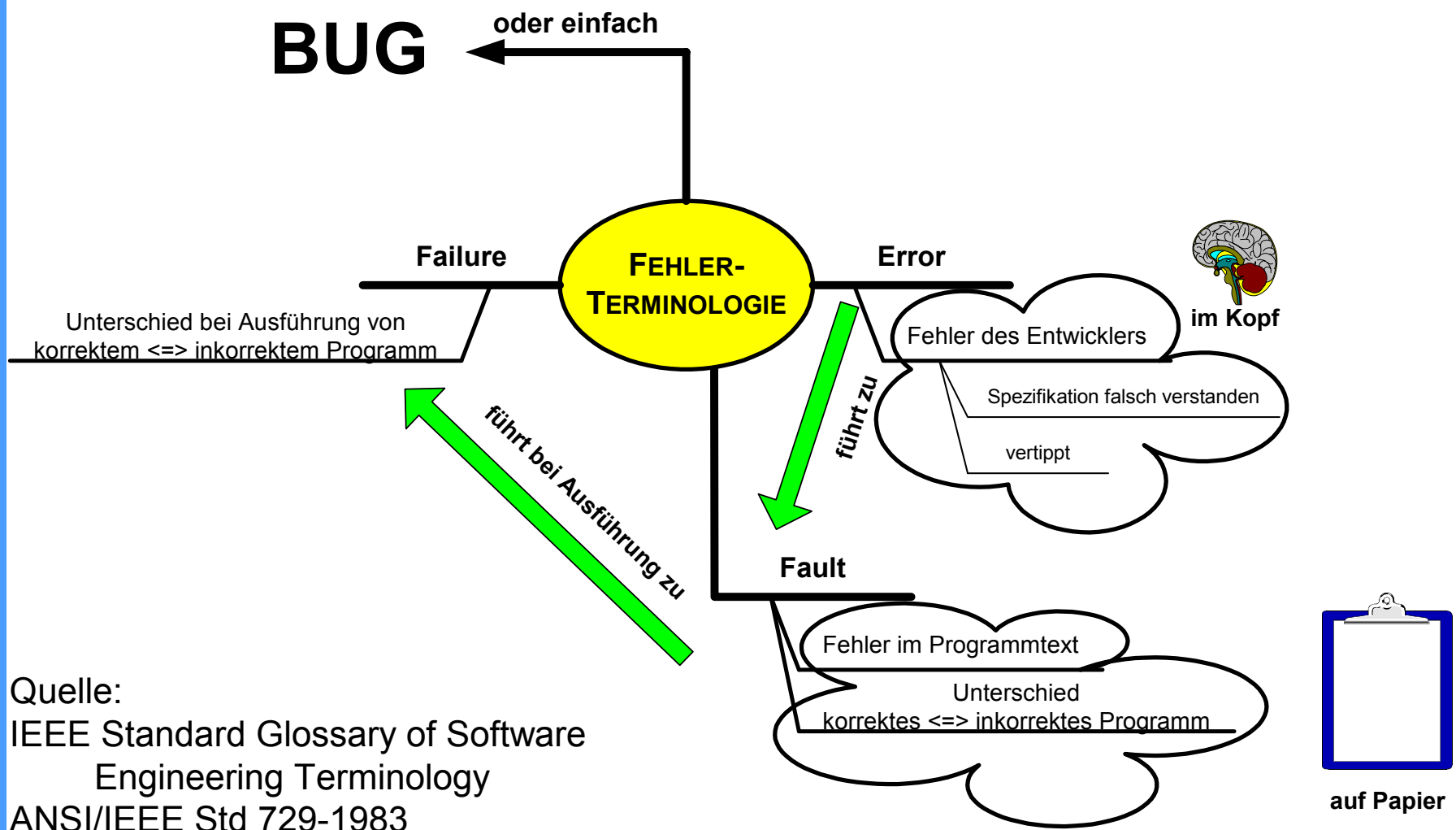
Testprinzipien (2)

- Schaue auch, ob das Programm tut, was es nicht tun soll!
  - Suche ungewollte Nebeneffekte.
- Wirf Testfälle nicht weg!
  - Sie sind in der Zukunft goldeswert.
  - Sie sind wiederholbar.
- Teste mit der Einstellung, Fehler zu finden!
  - Auch wenn die Korrektur Zeit brauchen wird.
- Je mehr Fehler man in einem Programmstück findet, desto mehr verstecken sich dort noch!
  - Das erlaubt fokussiertes Testen, indem man gezielt weitertestet.
- Testen ist eine kreative und anspruchsvolle Tätigkeit!
- Ein Testfall ist gut, wenn er mit hoher Wahrscheinlichkeit Fehler findet.

Der ideale Tester



Fehler-Terminologie



Quelle:
IEEE Standard Glossary of Software
Engineering Terminology
ANSI/IEEE Std 729-1983