SOFTWARE ENGINEERING 2

TESTEN – GRUNDLAGEN UND BEGRIFFE

Fehlerbegriff

- **Fehler(wirkung)**
  Nichterfüllung einer festgelegten Anforderung
  (Abweichung zwischen Ist- und Sollverhalten)
  Feststellbar, ohne den inneren Aufbau des Systems kennen zu müssen

- **Mangel**
  Anforderung (oder berechtigte Erwartung) wird nicht angemessen erfüllt
  Beispiele:
  Lange Reaktionszeit bei der Eingabe;
  „unschöne“ Graphik; sehr großer Speicherbedarf

➢ Ein Fehler oder Mangel kann nur festgestellt werden, wenn vorher festgelegt wurde, was das erwartete/korrekte Verhalten ist

- **Fehlerwirkung (failure)**
  Fehlfunktion; äußerer Fehler; Ausfall
  Tritt nur bei Ausführung der Software auf
- **Fehlerzustand (fault)**
  Defekt; innerer Fehler; Bug
  Zustand der Software (z.B. fehlerhafte Anweisung)
- **Fehlerursache (error)**
  Fehlhandlung einer Person (z.B. Codierungsfehler) zu einem bestimmten Zeitpunkt

Fehlhandlung

- Die menschliche Handlung (des Entwicklers), die zu einem Fehlerzustand in der Software führt.
- Eine menschliche Handlung (des Anwenders), die ein unerwünschtes Ergebnis (im Sinne von Fehlerwirkung) zur Folge hat (Fehlbedienung).
- Unwissentlich, versehentlich oder absichtlich ausgeführte Handlung oder Unterlassung, die unter gegebenen Umständen (Aufgabenstellung, Umfeld) dazu führt, dass eine geforderte Funktion eines Produkts beeinträchtigt ist.

Fehlerzustand und Fehlerwirkung

- Fehlerzustand/Defekt (fault)
  - Inkorrektes Teilprogramm (z.B. mit inkorrekter Anweisung oder Datendefinition), das Ursache für eine Fehlerwirkung sein kann.
  - Zustand eines (Software-)Produkts oder einer seiner Komponenten, der unter spezifischen Bedingungen (z.B. bei einer hohen Belastung) eine geforderte Funktion des Produkts beeinträchtigen kann bzw. zu einer Fehlerwirkung führt.

- Fehlerwirkung (failure)
  - Wirkung eines Fehlerzustands, die bei der Ausführung eines Programms (Testobjekt) nach »außen« in Erscheinung tritt.
  - Abweichung zwischen (spezifizierten) Soll-Wert und (beobachtetem) Ist-Wert (bzw. Soll- und Ist-Verhalten).
  - Abweichung einer Komponente/eines Systems von der erwarteten Lieferung, Leistung oder dem Ergebnis.

Was tun gegen Fehler?

- Maßnahmen zur Vermeidung von Fehlhandlungen
  - Ausbildung/Schulung
  - Standards
  - Codierrichtlinien
- Maßnahmen zum Entdecken von Fehlerzuständen
  - Reviews
  - Statische Codeanalysen (Statische Tests)
- Maßnahmen zum Auffinden von Fehlerwirkungen
  - Testen (Dynamische Tests)

Was ist Testen?

- Prozess, der sich (sowohl statisch als auch dynamisch) mit der Planung, Vorbereitung und Bewertung einer Software und den hierzu in Beziehung stehenden Arbeitsergebnissen befasst, um die Software mit dem Ziel zu bewerten,
  - dass diese allen festgelegten Anforderungen und
  - ihren Zweck erfüllt und
  - um etwaige Fehlerzustände zu finden.

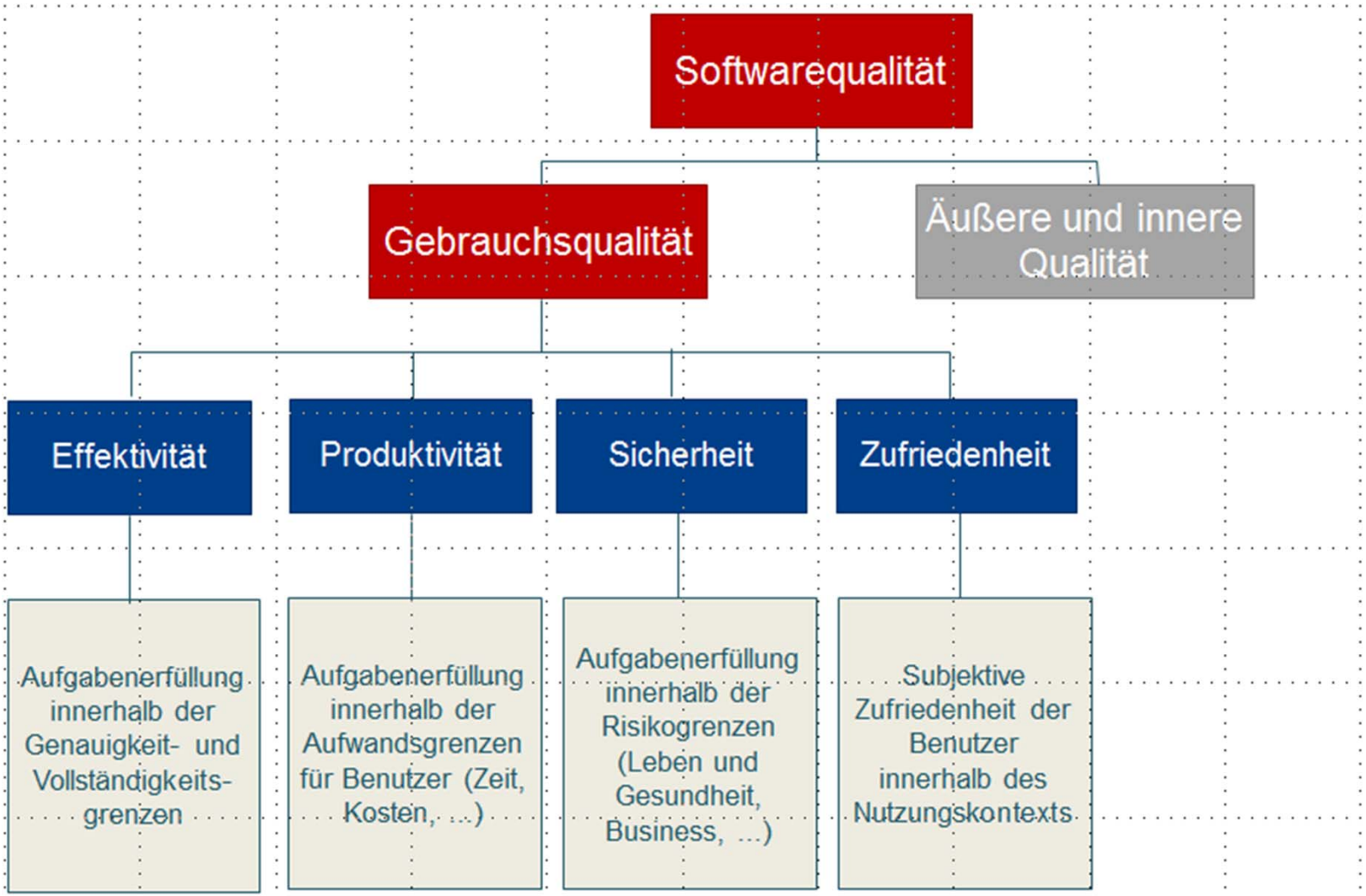
- Validierung ist Prüfung gegen die Anforderungen
  → Wurde das **richtige System** entwickelt?

  Genauer: Prüfung, ob ein Entwicklungsergebnis die individuellen Anforderungen bezüglich einer speziellen beabsichtigten Nutzung erfüllt.

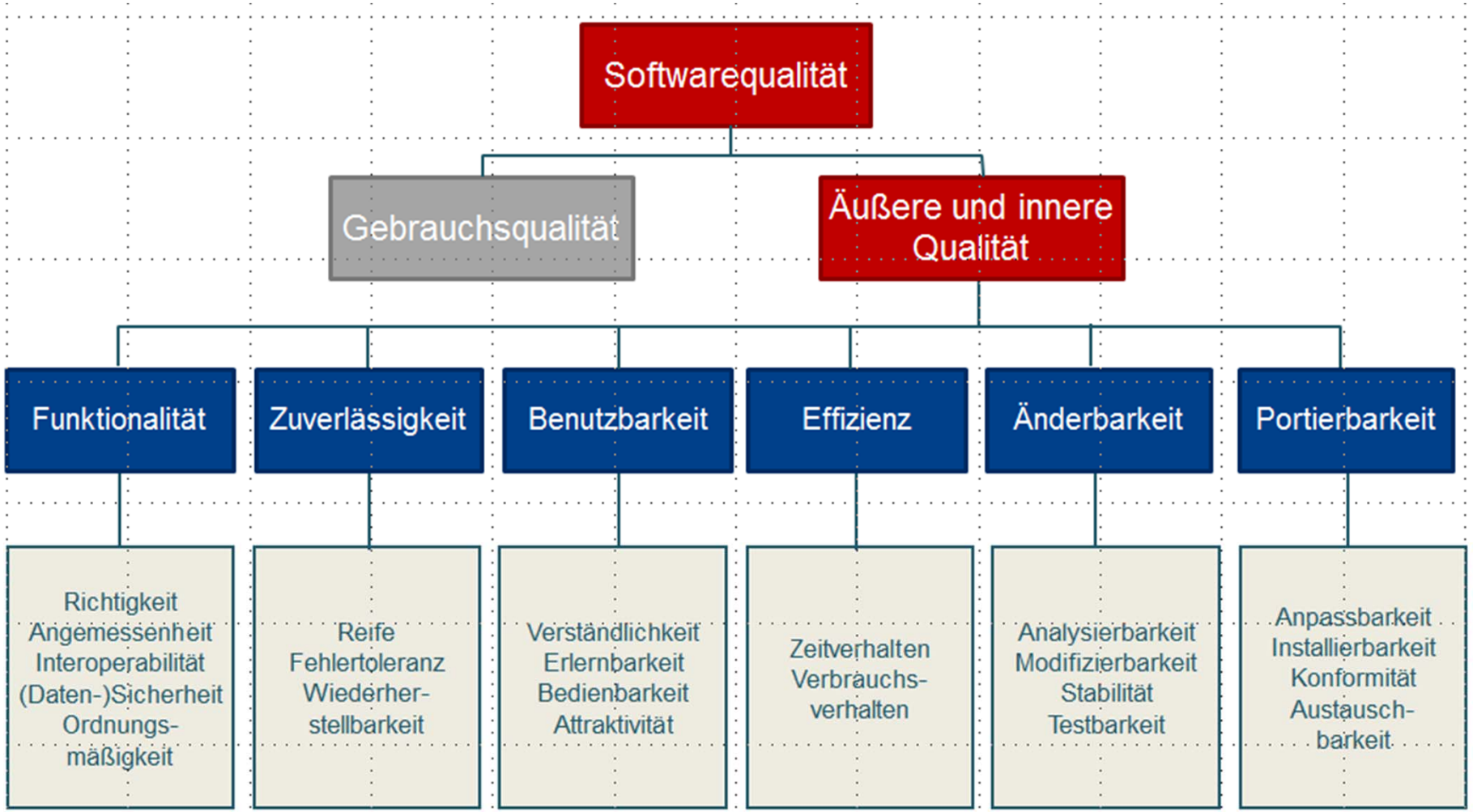
- Verifikation ist Prüfung gegen die technische Spezifikation
  → Wurde das System **richtig entwickelt**?

  Genauer: Prüfung, ob die Ergebnisse einer Arbeitsphase die Anforderungen der Eingangsdokumenten erfüllen

Softwarequalitätsbegriff



Softwarequalitätsbegriff



Erhöht Testen die Qualität?

- Testen ist ein Prozess zum Messen (Abschätzen) der Qualität.
  → Erfassung von Fehlerwirkungen!
  → Werden alle Fehler erfasst?

- Testen erhöht indirekt die Qualität, indem (im Nachgang) Fehlerzustände beseitigt werden können.

- Testen erhöht indirekt die Qualität des Entwicklungsprozesses, da systematische Fehler erkannt und konstruktiv beseitigt werden können.

- Testen kann dazu dienen, das Vertrauen (des Kunden/der Anwender) in die Qualität des Produktes zu erhöhen (oder zu verlieren…)

1. **Testen zeigt die Anwesenheit von Fehlern**
2. **Vollständiges Testen ist nicht möglich**
3. **Mit dem Testen frühzeitig beginnen**
4. **Häufung von Fehlern (Fehlerschwerpunkte)**
5. **Wiederholungen haben keine Wirksamkeit (Pesticide Paradox)**
6. **Testen ist abhängig vom Umfeld**
7. **Trugschluss: Keine Fehler heißt, dass das System brauchbar ist**

Testfallarten

- Positivtestfall
  Testet das spezifizierte und erwartete Verhalten für den Normalfall ab.

- Negativtestfall
  Testet das Verhalten auf fehlerhafte Eingaben oder auf erwartete Ausnahmebehandlung ab.

- Robustheitstestfall
  Testet auf unerwartete Eingaben oder nicht spezifizierte Ausnahmen.

- In der Praxis müssen alle Testfallarten zur Anwendung kommen!
  Meist werden zunächst nur Positivtestfälle spezifiziert
  → Falsche Sicherheit

Abstrakte und konkrete Testfälle

- Abstrakte (Logische) Testfälle spezifizieren einen Test ohne konkrete Werte. Vorgabe von Bereichen oder allgemeinen Abhängigkeiten

  Beispiele (Test Dreieck)

Abstrakter Testfall	**1**	**2**	**3**
Daten	A = B = C; A,B,C > 0	0 < A < B; C=B	A < 0; B>0; C>0
Erwartetes Ergebnis	Gleichseitiges Dreieck	Gleichschenkliges Dreieck	Fehlermeldung

- Konkrete Testfälle spezifizieren einen Test mit eindeutigen Werten.

  Beispiele (Test Dreieck)

Konkreter Testfall	**1**	**2**	**3**
Daten	A = 5; B = 5; C=5	A=2; B=4; C=4	A=-2; B=3; C=5
Erwartetes Ergebnis	Gleichseitiges Dreieck	Gleichschenkliges Dreieck	Fehlermeldung

Testorakel

- Nach Durchführung der Testfälle muss entschieden werden, ob der Testfall erfolgreich abgeschlossen wurde.
- Die Sollwerte müssen aus der Spezifikation abgeleitet werden.
  - Spezifikation ist diesbezüglich eindeutig
    → Übliche Vorgehensweise
  - Erzeugung der Sollwerte durch einen ausführbaren Prototypen
    → Beispiel modellgestütztes Vorgehen
    → Auch der Prototyp muss auf einer formalen Spezifikation beruhen (oder diese darstellen)
  - Entwicklung mehrerer Lösungen parallel
    → Gegenseitiger Vergleich (Back-to-back-Test)

- Man spricht auch vom Testorakel, das befragt werden muss.
- Weitere Quellen:
  Benutzerhandbuch, Vorgängerversionen, Plausibilität, Erfahrung

- Fehler sind in jedem Softwaresystem vorhanden
- Irren ist menschlich
  → aber wer gibt das gerne zu?

- Wer sollte testen?

- Entwicklertests
  - Keine fachliche Einarbeitung notwendig
  - Keine schlechten Nachrichten von anderen
  - Blindheit gegenüber eigenen Fehlern

- Unabhängiges Testteam
  - Unvoreingenommenheit
  - Methodisches Testwissen
  - Einarbeitung notwendig

- Abstufungen
  - Entwickler selbst
  - Anderes Teammitglied
  - Mitarbeiter aus gleicher Abteilung
  - Mitarbeiter aus Testabteilung

- Aufteilung ist in der Testplanung festzulegen
- Abhängig von Produkt und Projekt
- Eine Mischung ist anzustreben