Software Testing

POS1 Schwerpunkt

Inhalt

- **Software Testing allgemein**
- **Testentwurfsverfahren**
- **Testgüte**
- **Teststufen**
- **Testprozess**

Software Testing allgemein

Was ist Software Testen?

- **Testing**: activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component. [IEEE 829 2008]

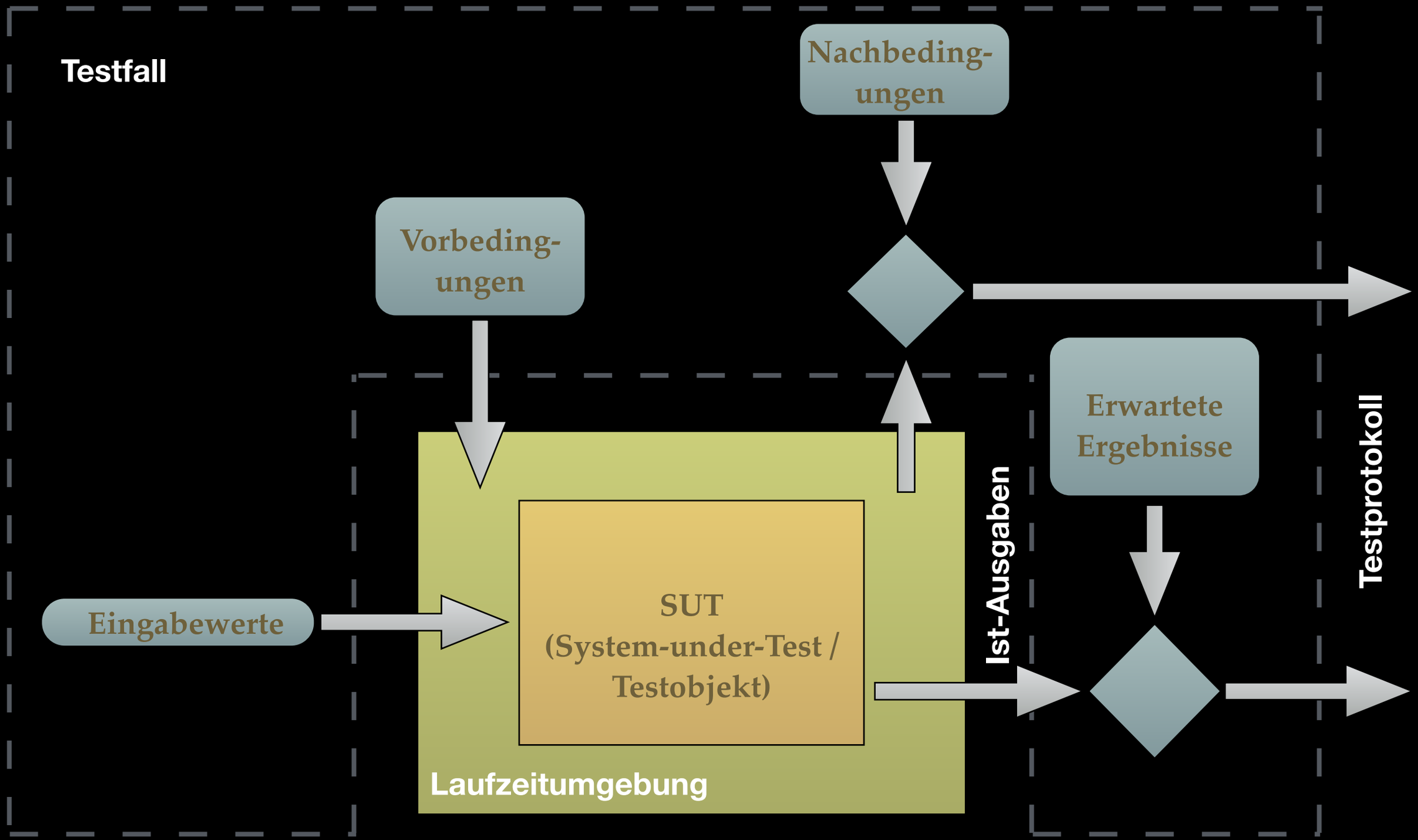
- **Testen**: Der Prozess, der aus allen Aktivitäten des Lebenszyklus besteht (sowohl statisch als auch dynamisch), die sich mit der Planung, Vorbereitung und Bewertung eines Softwarepro- dukts und dazugehöriger Arbeitsergebnisse befassen. Ziel des Prozesses ist sicherzustellen, dass diese allen festgelegten Anforderungen genügen, dass sie ihren Zweck erfüllen, und etwaige Fehlerzustände zu finden. [ISTQB Glossar 2011]

Kategorisierung von Software Testen

- **statische Analysen**, welche den zu untersuchenden bzw. zu prüfenden Gegenstand (Teil-, Zwischen- oder Endprodukt, = Testobjekt) nicht ausführen, sondern als solches analysieren und somit auf alle Entwicklungsprodukte wie z.B. Anforderungs- und Entwurfsspezifikationen sowie den Programmcode „als Text“ anwendbar sind

- **dynamische Tests**, welche „ausführbare“ Testobjekte, in der Regel also Programmcode, erfordern und diese unter kontrollierten Bedingungen mit dem Ziel ausführen, Fehler zu finden

System-under-Test / Testobjekt



Testentwurfsverfahren

Testentwurfsverfahren

- Black-Box Verfahren
- White-Box Verfahren

Black-Box Verfahren

- Testbasis ist *rein die Spezifikation*, es wird kein Wissen über Aufbau und andere Interna der Anwendung für die Testfallerstellung verwendet

- Konzentration auf *funktionale Sicht*, es wird auf *charakteristische bzw. fehlerträchtige Eingabewerte* fokussiert

- Ausprägungen des Black-Box Verfahrens

  A. Äquivalenzklassentest:
     Gruppierung von Eingabewerten in Klassen mit gleichem Verhalten, 1 Wert der Klasse pro Test

  B. Grenzwertanalyse:
     Grenzwerte und deren Nachbarwerte werden verwendet

White-Box Verfahren

- Tests werden *basierend auf der Programmstruktur* entworfen -> Wissen über Programmaufbau wird ausgenutzt
- Überdeckung von *Anweisungen, Verzweigungen und Schleifen* als Gegenstand der Testfall-Erstellung
- Ausprägungen des White-Box Verfahrens

  A. Kontrollfluss-basierter Test:
  Betrachtung möglicher Wege durch das Programm

  B. Datenfluss-basierter Test:
  Betrachtung der Variablenverwendung, aufbauend auf Kontollflussorientierten Tests

  C. Bedingungstest:
  alle Teilbedingungen testen

White- oder Black-Box Verfahren?

- White-Box
  Adressiert nicht Spezifikation -> es können funktionale Probleme trotz erfolgreicher Tests bestehen

- Black-Box
  Adressiert nicht mögliche Probleme außerhalb der Spezifikation, z.B. nicht erreichbaren Code etc. (oft sind das Probleme mit negativen Auswirkungen auf Security und Zuverlässigkeit von Software)

- **Fazit: Zweckmäßig ist eine sinnvolle Kombination!**

Testgüte

Messung Testüberdeckung

- Andere Bezeichnung: Testabdeckung
- Metriken zur Messung der Intensität ausgeführter Tests
- Messung des Testfortschritts
  = Anhaltspunkte zur Entscheidung, wann "genug" getestet wurde
- z.B. Fehlerdichte (=Fehler pro Zeile), Fehlerrate (= Fehler pro Zeit), Testfälle pro Anwendungsfall, …

Mutationstests

- Testabdeckungsmessung ist bei Abwesenheit von Fehlern nicht eindeutig: Entweder ist das Testobjekt wirklich von hoher Qualität, oder aber die Testfälle sind von schlechter Qualität bzw. falsch gewählt!

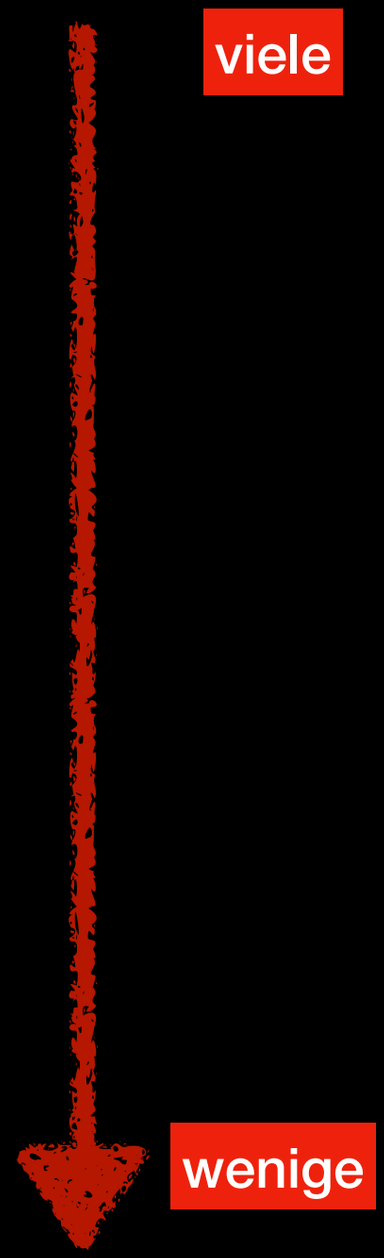
- Provokation bestimmter Fehlerzustände durch Änderungen im Testobjekt (= Code)
  -> *Fehlerinjektion* bzw. “*fault-injection*”

- Ausführung Tests auf diese "Mutanten" (= geänderter Code) mit Messung der tatsächlich von den Testfällen erkannten Anzahl der injizierten Fehler

Teststufen bzw. Testpyramide

Teststufen

- Modul-/Komponententest ("*unit test*")
- Integrationstest
- Systemtest
- Abnahmetest
  -> manchmal zusätzliche Unterscheidung in
    - "*factory acceptance test*" (FAT) und
    - "*user acceptance test*" (UAT)



Modul- / Komponententest

- = "unit test"
- Test *elementarer*, *isolierter* Programmbausteine
- Meist vom *Entwickler* selbst durchgeführt
- Black-und White-Box Tests anwendbar, meist aber *sehr nahe am Code (also eher White-Box)*
- *Automatisierung ist ein MUSS in der modernen Entwicklung! (-> JUnit + Mockito o. ä.)*

Integrationstest

- Test des korrekten *Zusammenspiels mehrerer Programmbausteine* (die jeweils für sich oft bereits erfolgreich dem Unit Test unterzogen wurden)
- Es wird also hauptsächlich getestet, ob die definierten *Schnittstellen* korrekt spezifiziert und implementiert sind
- Ausführung in *speziellen Testumgebungen*
- Einsatz von *Automatisierungswerkzeugen* äußerst sinnvoll

Systemtest

- Test auf *Funktionsfähigkeit des gesamten Systems* entsprechend der Anforderungen (Spezifikation)

- Reiner *funktionaler Black-Box* Test

- Zusätzlich: Security-, Performance-, Last-, Stress-, Robustheits-Test und auch andere Tests zu *nicht-funktionalen* Anforderungen

- Spezialfall *System-Integrationstest*:
  Es wird das Zusammenspiel des Systems *mit anderen Systemen* in seiner Umwelt getestet

- Automatisierung erstrebenswert, oft aber nicht sinnvoll möglich

Abnahmetest

- In Zielumgebung unter echten *Einsatzbedingungen*
- Test auf Erwartungen der *Anwender* bzw. des *Kunden*
- Bestätigung der *Gebrauchstauglichkeit* und der äußeren Qualität der Software

- Reiner *Black-Box* Test
- Ideal: *Abnahmekriterien* aus Auftrag
- Endgültiger *System-Integrationstest* (Zusammenspiel des Systems mit anderen Systemen in Zielumgebung)
- Automatisierung erstrebenswert (Regressionstests zu alten Funktionen), allerdings neue Funktionen meist manuell durch Kunde

Testausführung und Testumgebung

Testausführung und Testumgebung

- Bis auf Abnahmetests werden Tests in eigenen **Testumgebungen** gefahren -> nicht in Produktion wegen *Gefahr von Störung* durch Fehler!

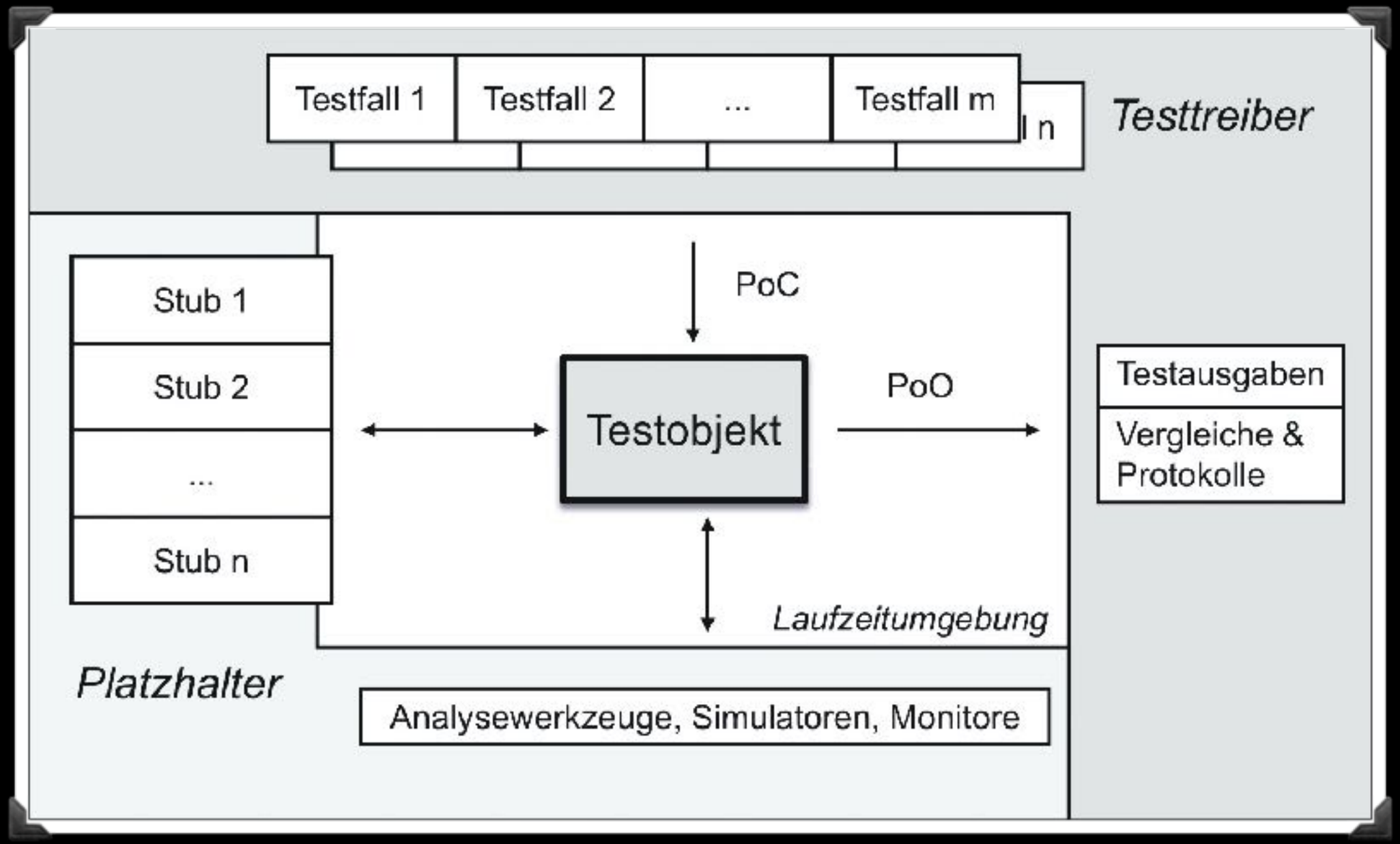
- **Testtreiber** zur Ausführung, **Testdatenbereitstellung** für Versorgung mit Eingangswerten

- **Stubs** als "Stellvertreter" für noch nicht fertig gestellte Programmteile

    A. *Dummy-Objekte*: sehr einfach, meist nur Daten (keine Information, ob ein Dummy-Objekt auch aufgerufen wurde)

    B. *Mock-Objekte*: stellen auch aufwendigere Test-spezifische Funktionalität zur Verfügung (“behaviour tests” möglich durch Aufrufinformationen)

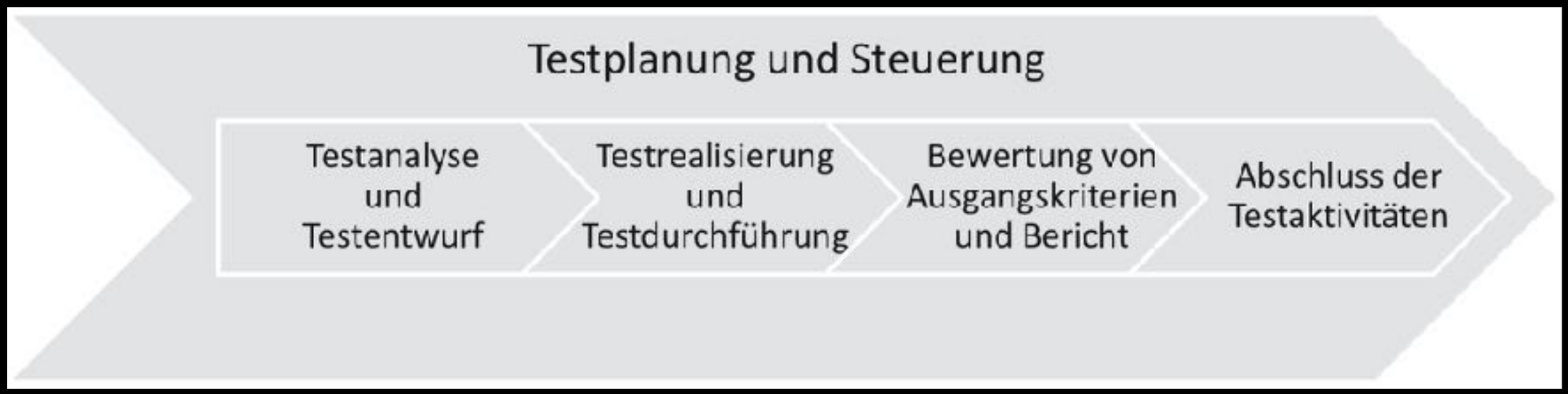
Aufbau Testumgebung



PoC = Point-of-Control. ***PoO = Point-of-Observation***

Testprozess

Testprozess



Testprozess nach ISTQB

„Be a yardstick of quality. Some people aren't used to an environment where excellence is expected.“

-Steve Jobs

Referenzen

- Wikipedia: http://en.wikipedia.org/wiki/Software_testing
- SWEBOK v3, Kapitel 4: http://www.computer.org/portal/web/swebok/swebokv3
- Winter, Ekssir-Monfared, Sneed, Seidl, Borner: "Der Integrationstest", Hanser, 2012.
- Sneed, Baumgartner, Seidl: "Der Systemtest", Hanser, 2011.