


- Einführung
- Software Fehler
- Konstruktive Qualitätssicherung
- ➔ ■ Software Test
- Statische Analyse



- Andreas Spillner, Tilo Linz: Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester - Foundation Level nach ISTQB-Standard (iSQI-Reihe), dpunkt.verlag
- Dirk W. Hoffmann: Software-Qualität, 2 Auflage, Springer Vieweg



- 
- Motivation
 - Testklassifikation
 - Black Box Testtechniken
 - White Box Testtechniken
 - Testmetriken
 - Grenzen des Software Tests
 - Testautomatisierung



Motivation

- Siehe z.B.
http://de.wikipedia.org/wiki/Liste_von_Programmfehlerbeispielen
- https://jaxenter.de/top-10-der-software-katastrophen-181?utm_source=nl&utm_medium=email



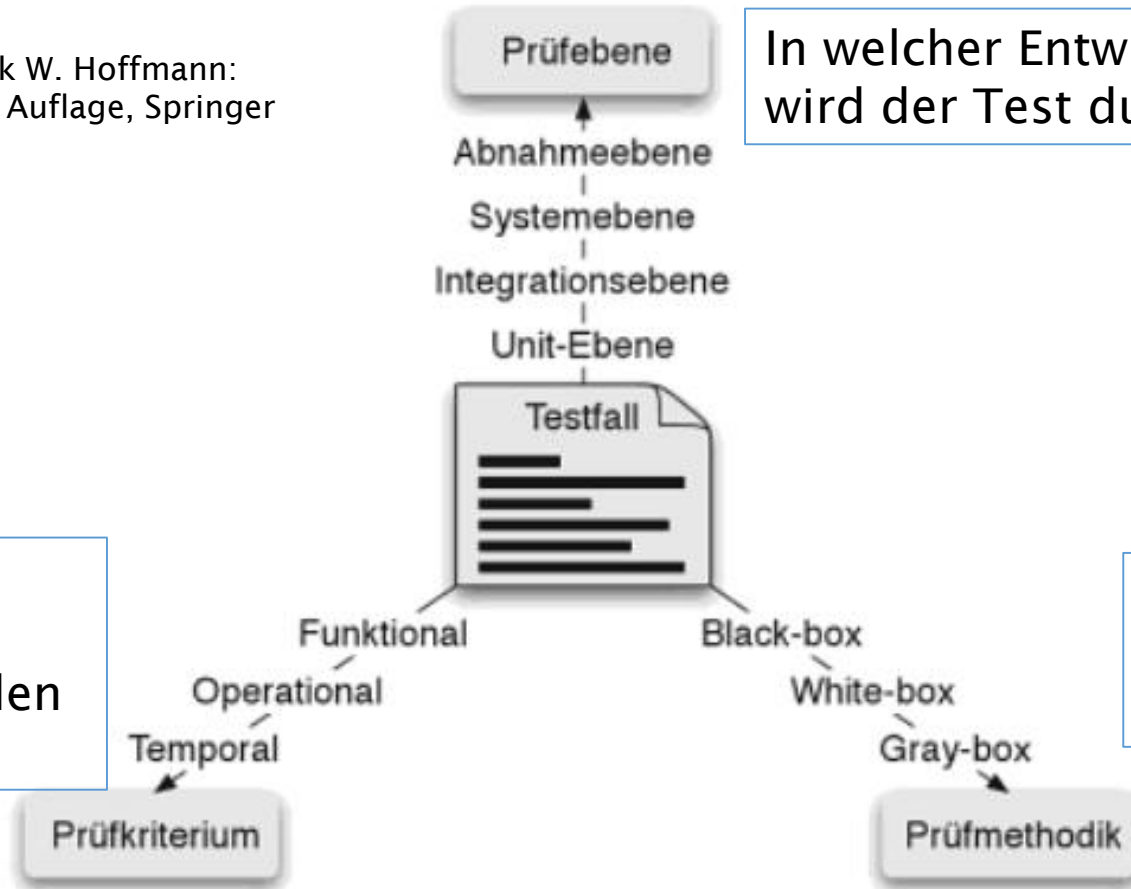
- Motivation
- ➔ ■ Testklassifikation
- Black Box Testtechniken
- White Box Testtechniken
- Testmetriken
- Grenzen des Software Tests
- Testautomatisierung



Testklassifikation

Quelle der Abb.: Dirk W. Hoffmann:
Software-Qualität, 2 Auflage, Springer
Vieweg

Welche
inhaltlichen
Aspekte werden
geprüft?

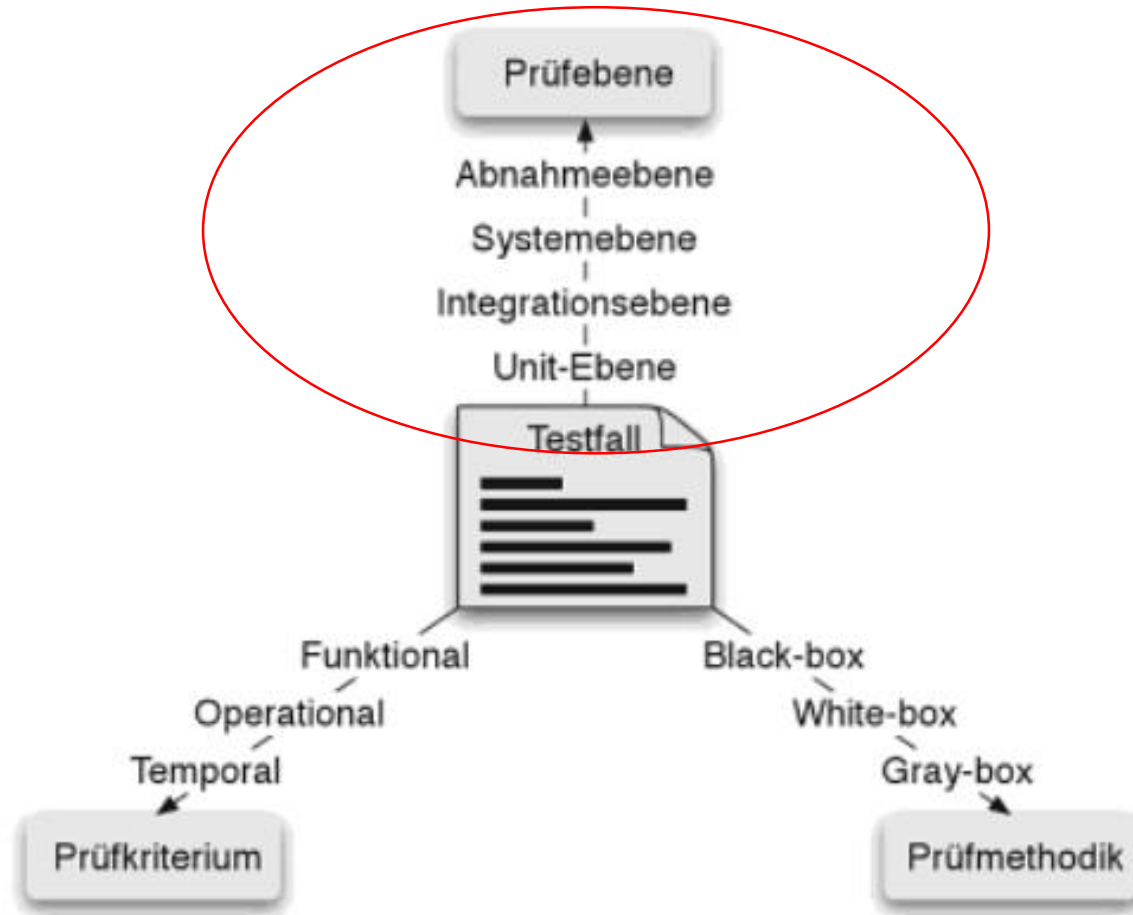


In welcher Entwicklungsphase
wird der Test durchgeführt?

Wie werden die
Testfälle
konstruiert?



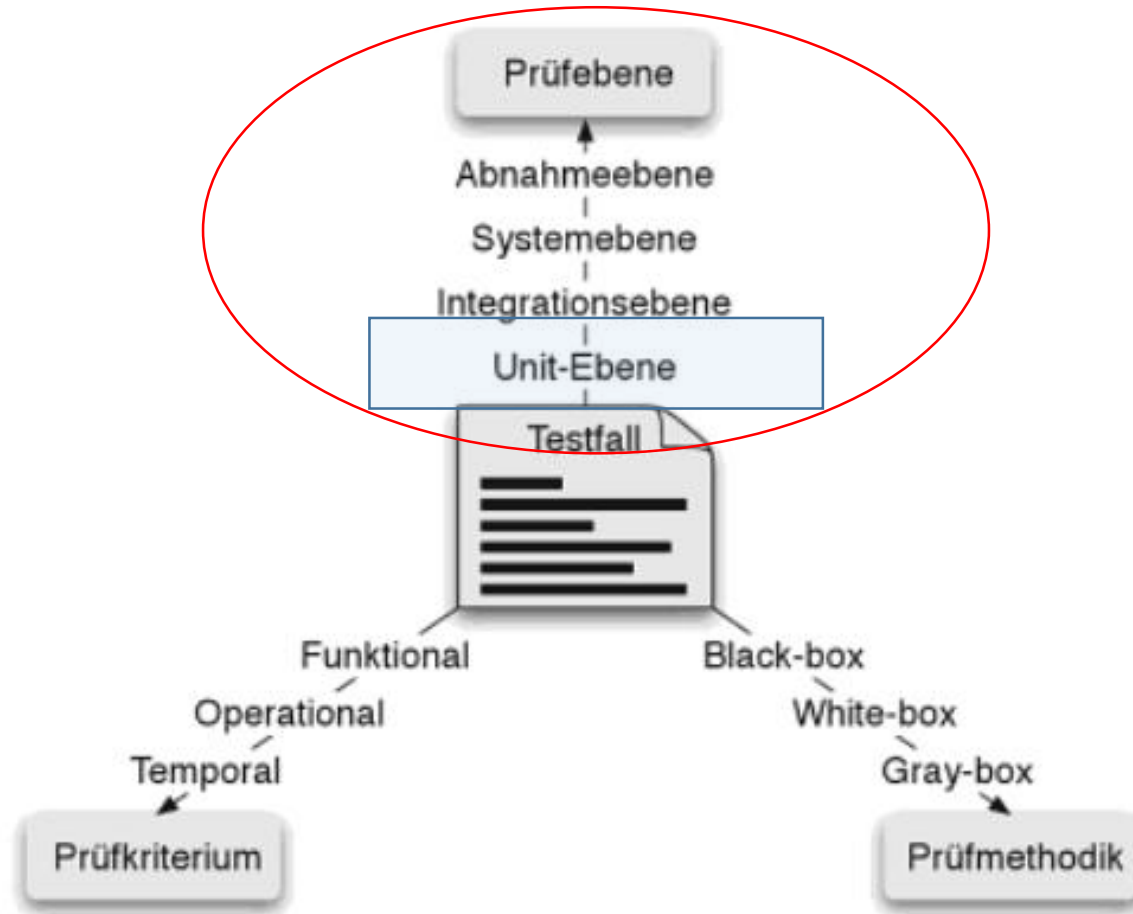
Testklassifikation



Prüfebenen

- **Unit Tests:** Test von atomaren Einheiten, die groß genug sind, um eigenständig getestet zu werden.
- **Integrationstests:** einzelne Einheiten werden zu größeren Komponenten zusammengeführt → Test, ob das Zusammenspiel funktioniert.
- **Systemtest:** Test des Systems als Ganzes auf Einhaltung der im Pflichtenheft festgelegten Eigenschaften.
- **Abnahmetest:** Ist ein Systemtest in der Umgebung und der Verantwortung des Auftraggebers.

Unit Tests



- Sie testen einzelne Einheiten isoliert.
- Einheiten sind typischerweise:
 - Einzelne Methoden
 - Klassen mit mehreren Attributen und Methoden.
 - Zusammengesetzte Komponenten mit definierten Schnittstellen.



Unit Tests

- Die Testfälle sollen zeigen, dass die Komponente tut, was sie soll.
- Wenn Defekte enthalten sind, sollen sie aufgezeigt werden.

Zwei Arten von Unit Test Cases

- Normale Programmausführung: Die Komponente tut, was sie soll.
- Ungewöhnliche Eingaben, falsche Eingaben etc: Die Komponente geht entsprechend der Spezifikation damit um.



Automatisierte Unit Tests

- **Ziel:** Möglichst viele Modultests automatisieren.
 - **Durchführung:** Verwendung eines Testautomatisierungs-Frameworks (z.B. JUnit) um Tests zu schreiben und durchzuführen.
- ➔ Ermöglicht, bei jeder Änderung **ALLE** Tests laufen zu lassen und das Ergebnis graphisch anzuzeigen.



Komponenten:

- **Aufbau:**

Sie definieren die Testfälle mit Eingaben und erwarteten Ergebnissen.

- **Aufruf**

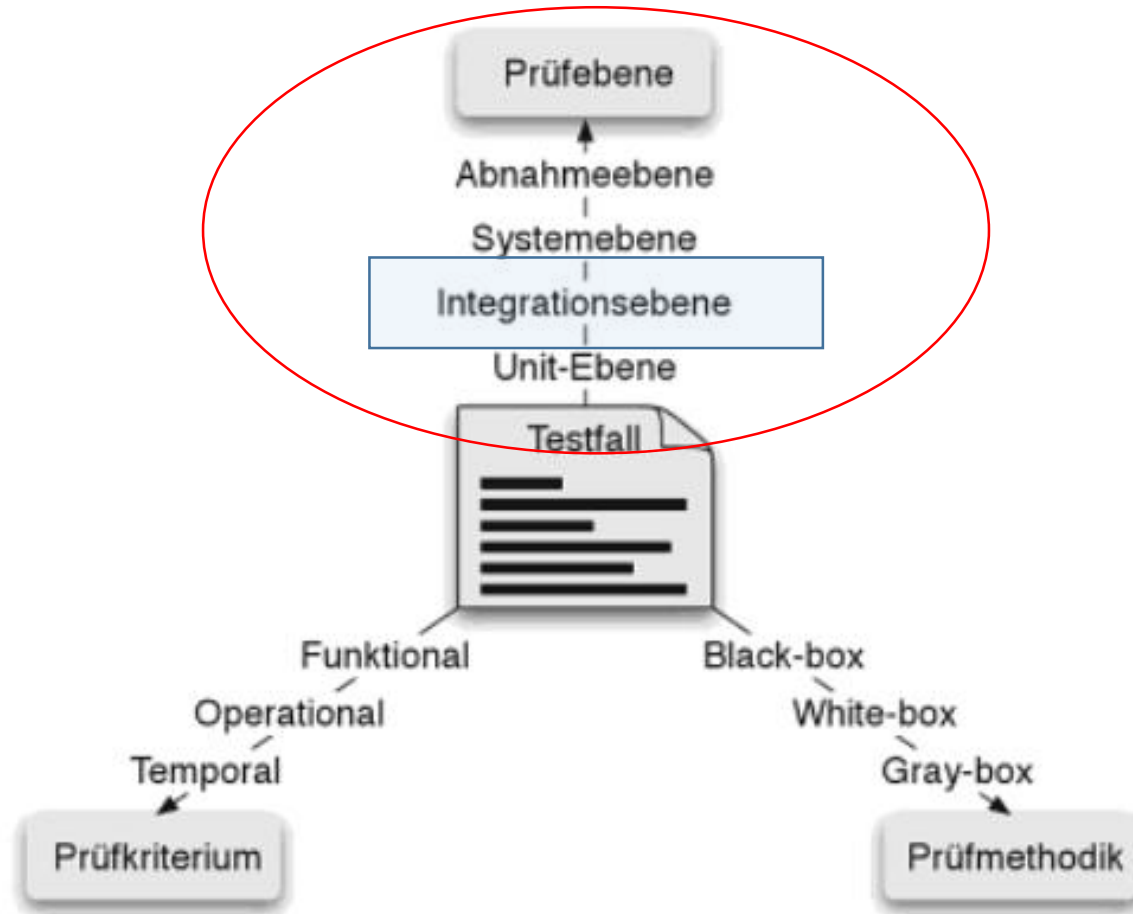
Sie rufen die zu testenden Objekte oder Methoden auf.

- **Auswertung**

Vergleich des wirklichen mit dem erwarteten Ergebnis.



Integrationstests



- Nächsthöhere Abstraktionsebene gegenüber Unit Test.
- Wird eingesetzt, wenn einzelne Programmmodule zu größeren Software Komponenten zusammengesetzt werden.
- Stellt sicher, dass die Komposition der separat getesteten Komponenten ein funktionsfähiges System ergibt.



- **Big Bang Integration**
- **Strukturorientierte Integration**
 - Bottom-Up
 - Top-Down
 - Outside-in
 - Inside-Out
- **Funktionsorientierte Integration**
 - Termingetrieben
 - Risikogetrieben
 - Testgetrieben
 - Anwendungsgetrieben



Big Bang Integration

Entwicklung sämtlicher Module, anschließend Integration auf einen Schlag.

▪ Nachteile

- Beginn erst wenn alle Module fertig sind.
- Gleichzeitige Integration aller Komponenten führt zu schwieriger Fehlersuche.

▪ Vorteil:

- Testtreiber und Mocks sind nicht nötig.



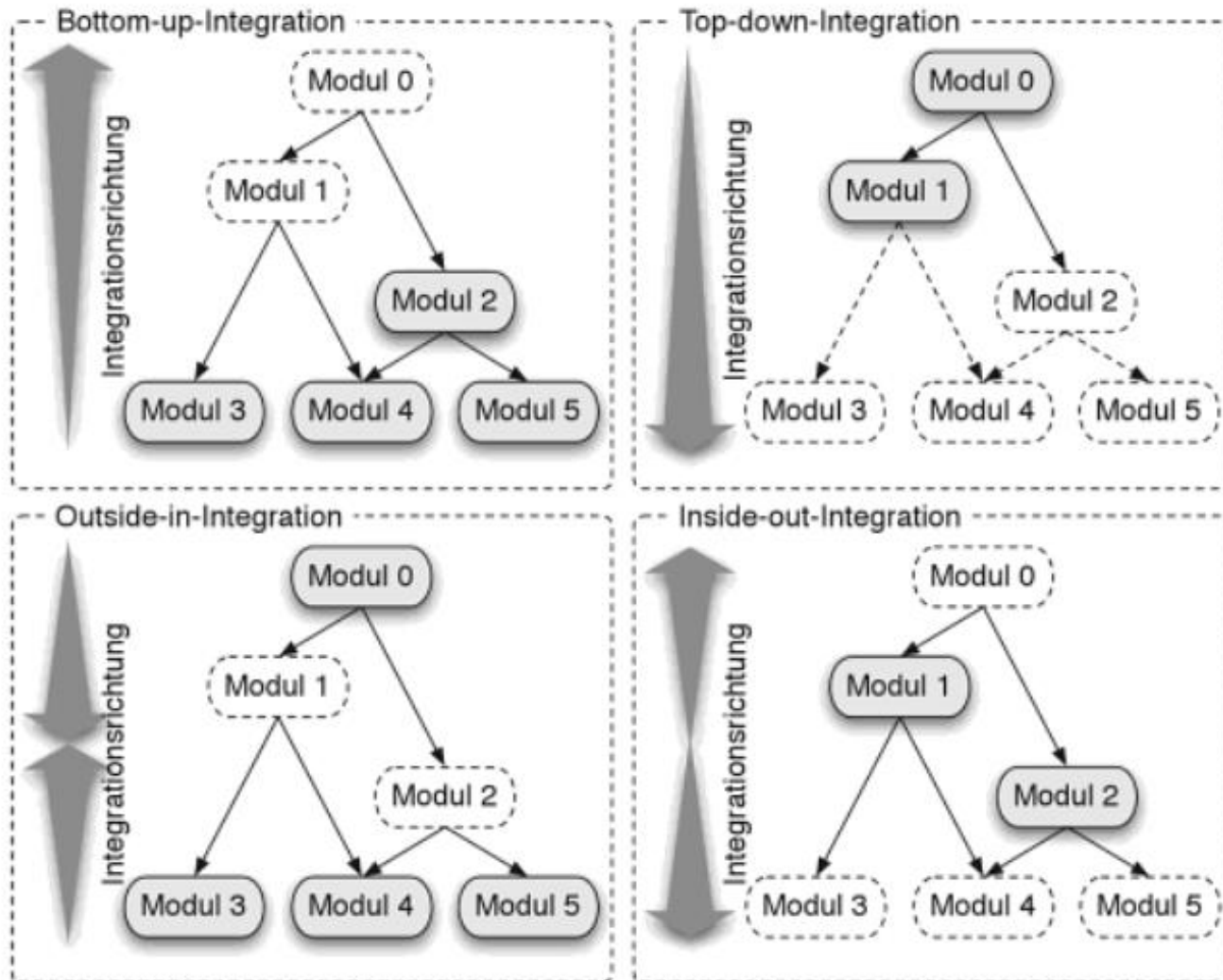
Strukturorientierte Integration

Inkrementelle Integration der Module zum Gesamtsystem. Reihenfolge der Integration richtet sich nach den Abhängigkeiten der Module

- **Bottom Up** – Ausgangspunkt: Basiskomponenten, Verwendung von Testtreibern.
- **Top Down** – Ausgangspunkt: Module der höchsten Schicht, Verwendung von Stubs.
- **Outside-In** – Integration von beiden Seiten nach innen.
- **Inside Out** – Integration von innen nach außen.



Integrationsstrategien im Vergleich

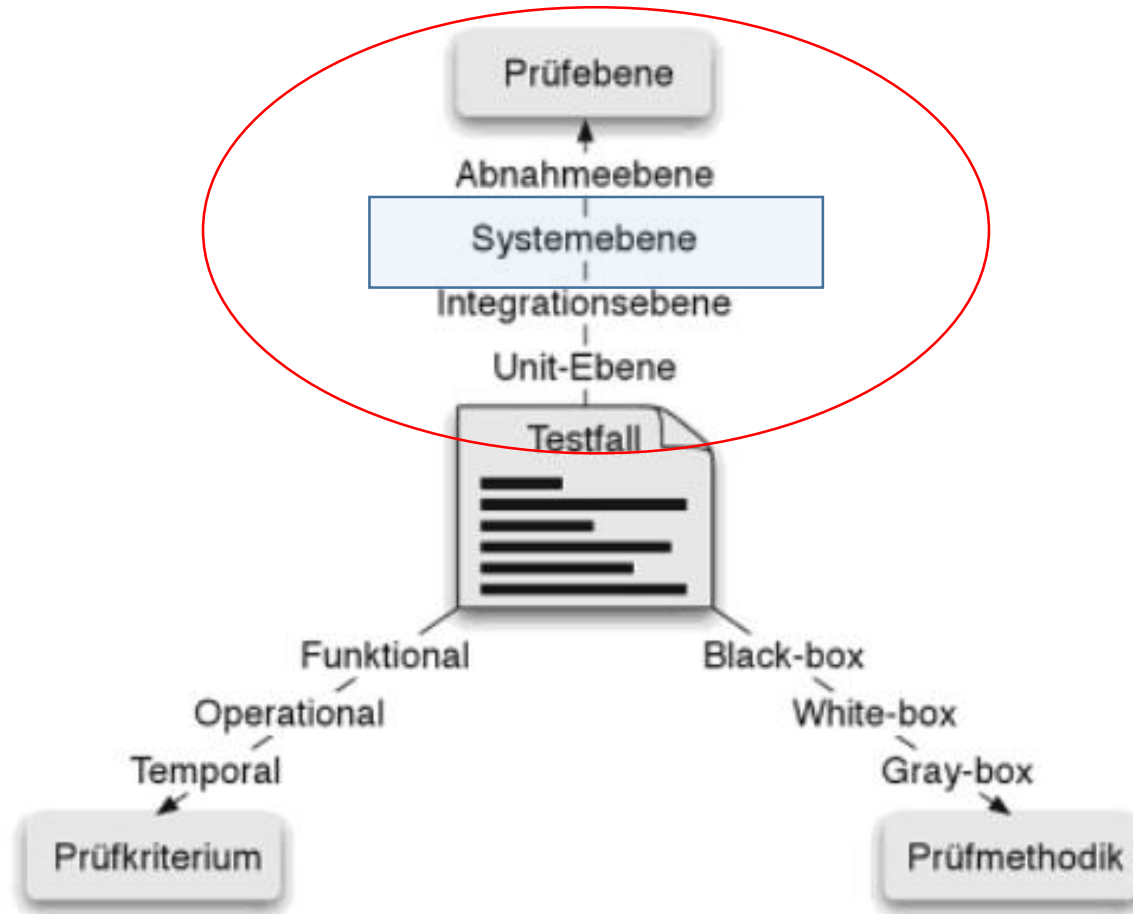


Quelle des Bilds:
D. Hoffmann,
Software-Qualität,
Springer, 2013

Funktionsorientierte Integration Integration anhand funktionaler oder operationaler Kriterien

- Termingetriebene Integration – entsprechend der Verfügbarkeit
- Risikogetriebene Integration – riskanteste als erstes
- Testgetriebene Integration – Integration für bestimmte Testfälle
- Anwendungsgetriebene Integration – Integration für bestimmte Usecases

Systemtests



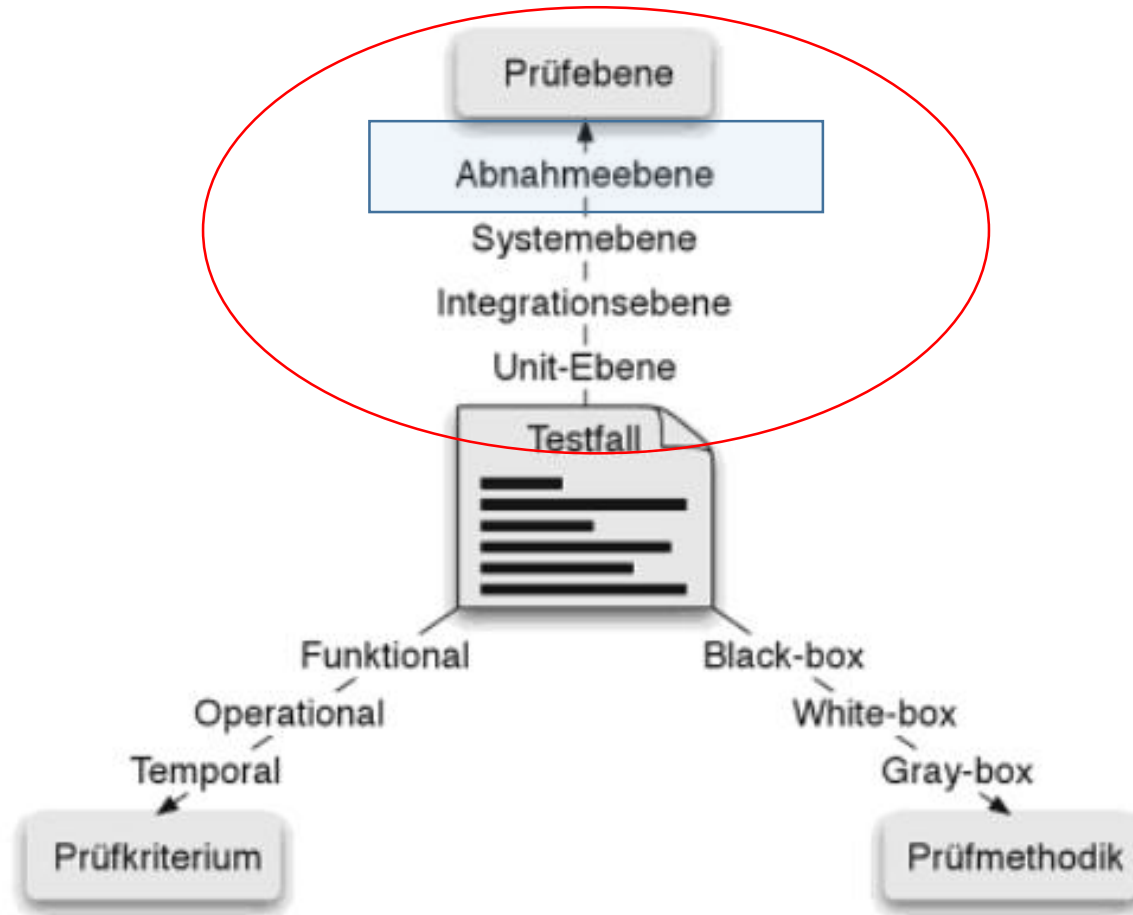
Merkmalsräume der Testklassifikation



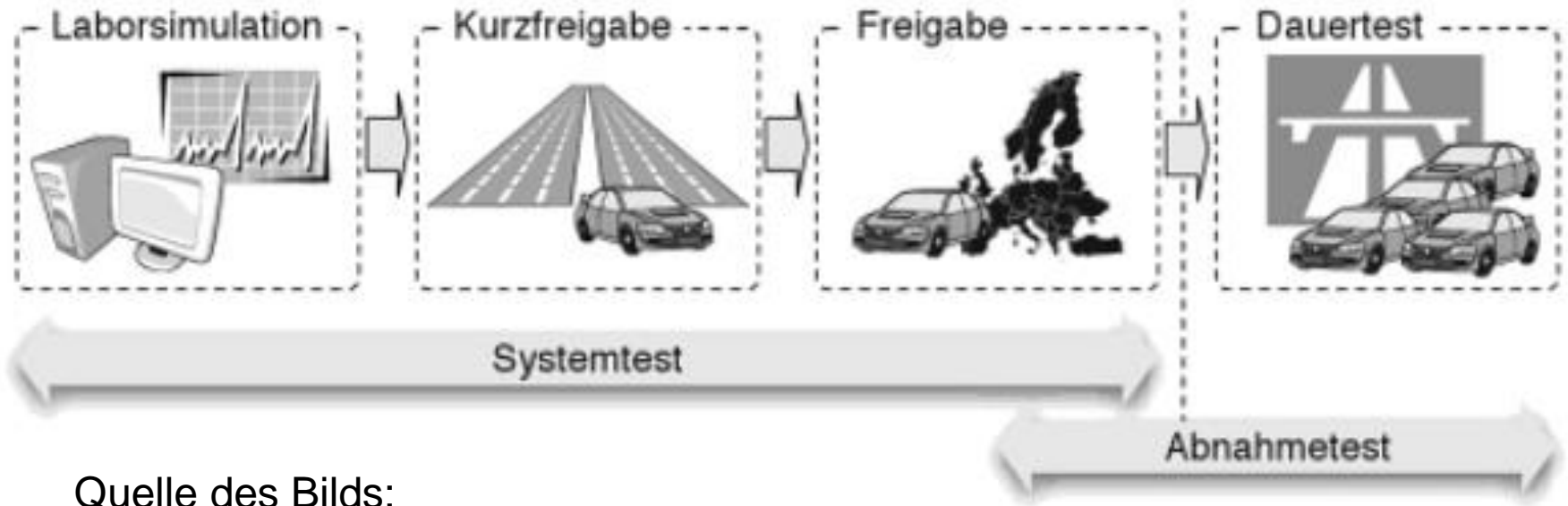
- Start, sobald alle Komponenten erfolgreich integriert sind.
- Findet in einer separaten Testumgebung statt, die der Produktivumgebung ähnelt.



Abnahmetests



Systemtest - Abnahmetest



Quelle des Bilds:
D. Hoffmann,
Software-Qualität,
Springer, 2013

Typische Phasen des System- und Abnahmetests
eines KFZ-Steuergeräts.



Abnahmetests sind ähnlich dem Systemtest

Unterschiede:

- Abnahmetest unter Federführung des Auftraggebers.
- Abnahmetest findet in der realen Einsatzumgebung des Kunden statt. Durchführung mit authentischen Daten.

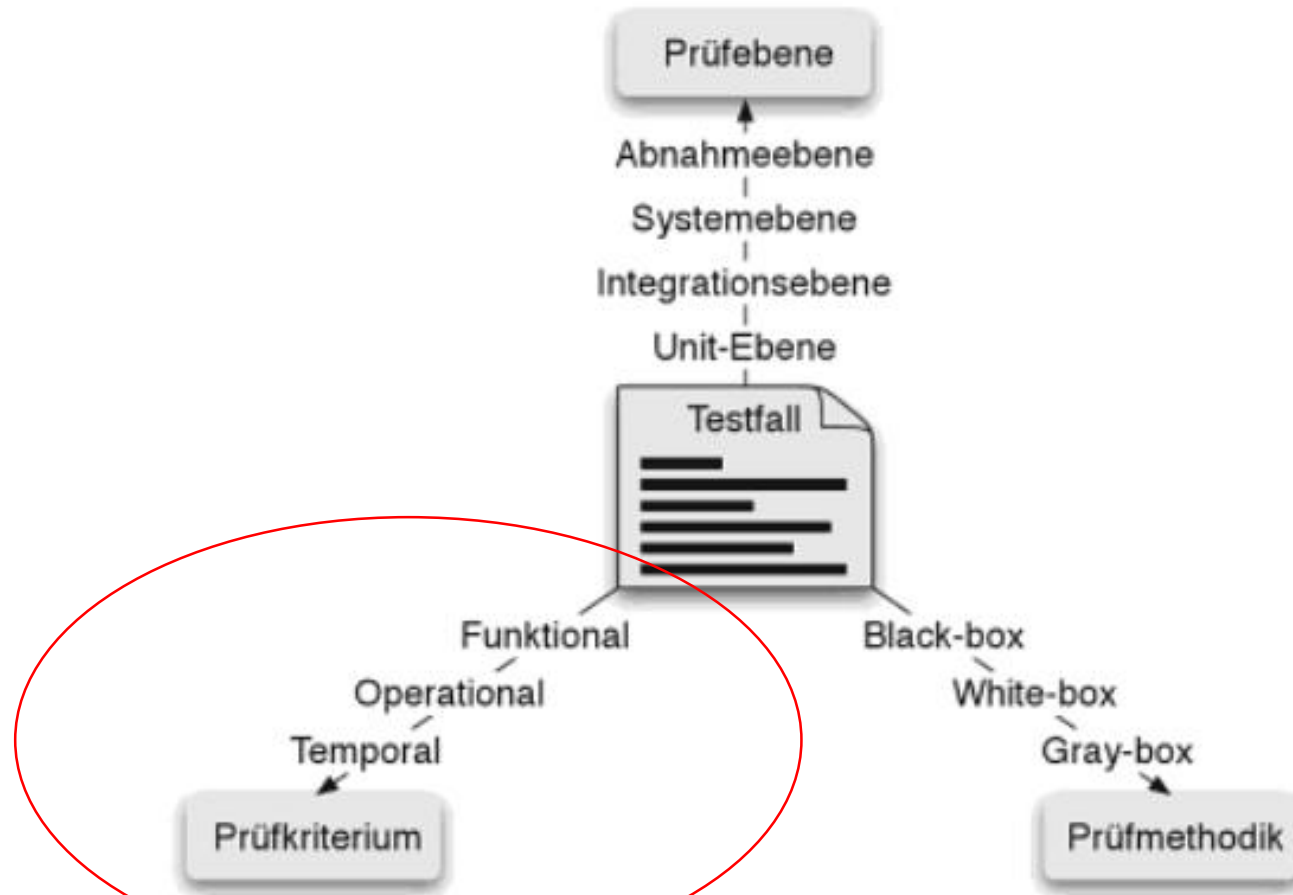
Abnahmetests sind juristisch relevant.

Empfehlenswert: Kunden bereits in die Systemtests einzubinden.

- ➔ Kunde ist früh informiert
- ➔ Teilabnahmen sind möglich



Testklassifikation

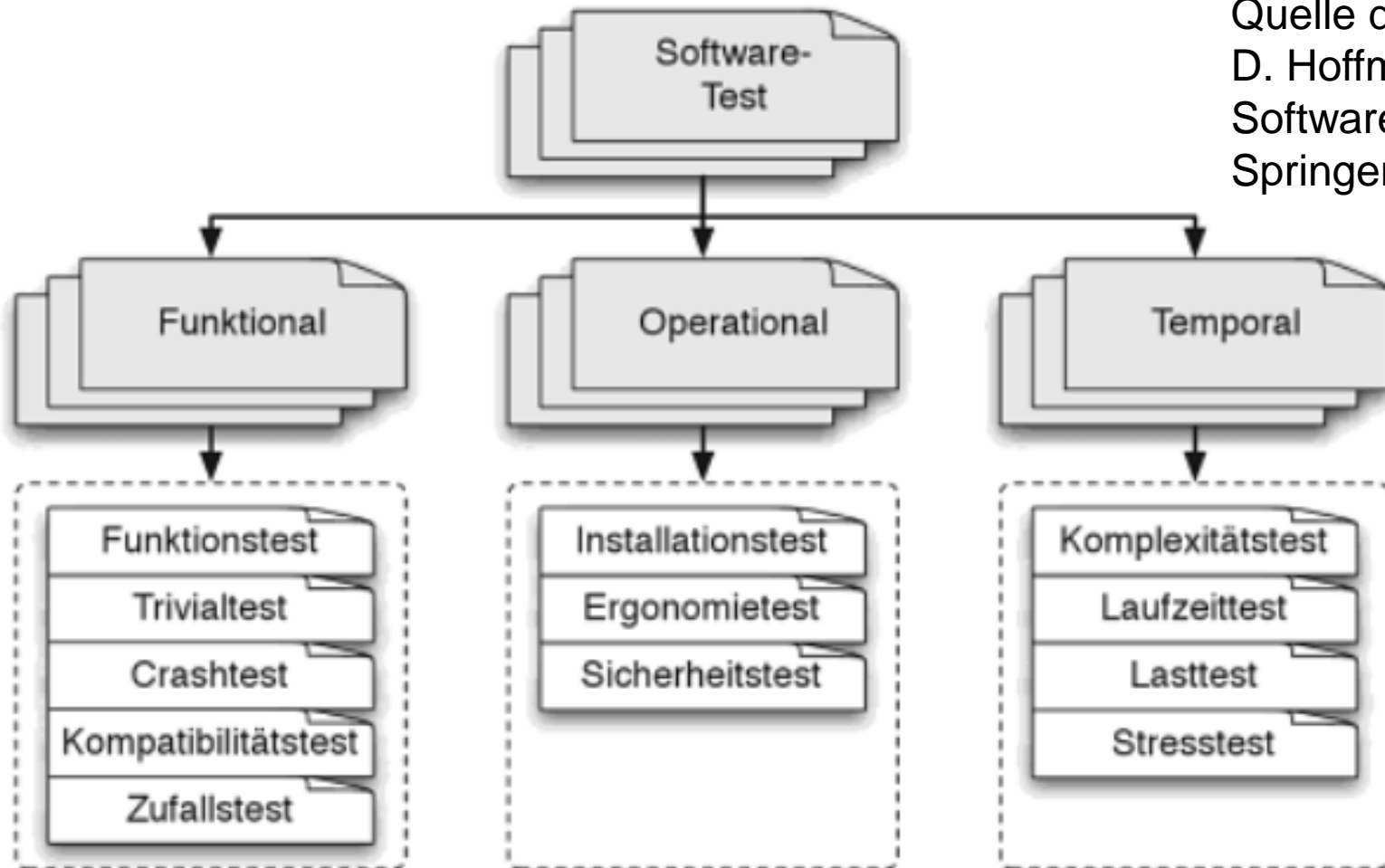


Merkmalsräume der Testklassifikation

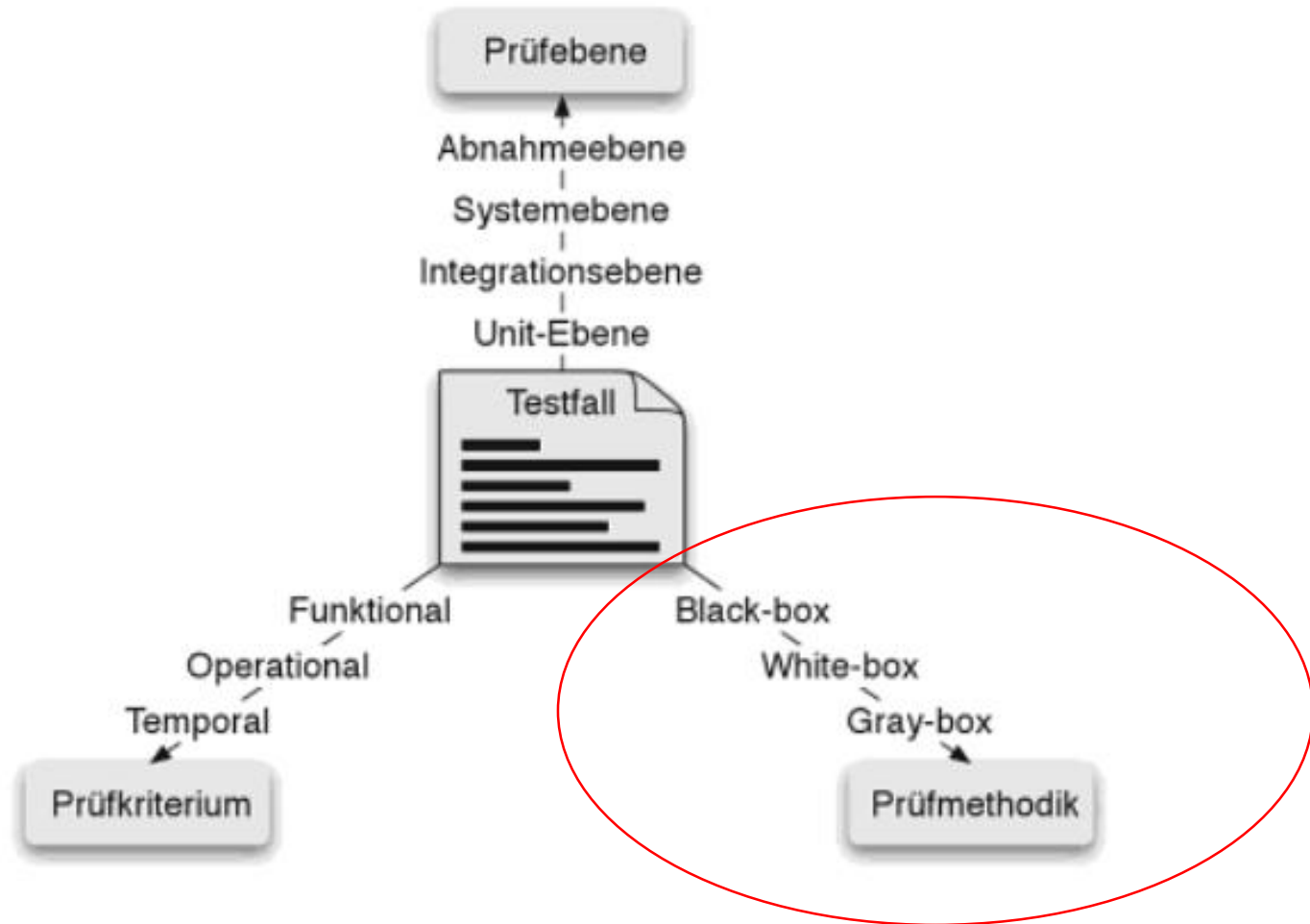


Prüfkriterien

Quelle der Abb:
D. Hoffmann,
Software-Qualität,
Springer, 2013



Testklassifikation



Erstellen von Testfällen

Ziel: Systematisches Vorgehen um mit möglichst wenig Aufwand möglichst viele Anforderungen zu überprüfen bzw. Fehler zu finden.

Vorgehen:

1. Die durch den Test verfolgten Ziele sowie Bedingungen und Voraussetzungen festlegen.
2. Testfälle spezifizieren.
3. Testausführung festlegen.



- **Black Box Tests**

Testfälle werden aus der Anforderungs- und Schnittstellenanalyse hergeleitet.

- **White Box Tests**

Testfälle werden systematisch aus der inneren Programmstruktur hergeleitet

- **Gray Box Tests**

Testfälle werden sowohl aus der Anforderungs- und Schnittstellenanalyse als auch aus der Kenntnis der inneren Programmstruktur hergeleitet.

