

## 2 – Testen während des Softwareentwicklungslebenszyklus

1. Testen im Kontext eines Softwareentwicklungslebenszyklus
2. Teststufen und Testarten
3. Wartungstest



## 2.1 Testen im Kontext eines Softwareentwicklungslebenszyklus

- FL-2.1.1 (K2) Sie können die Auswirkungen des gewählten Softwareentwicklungslebenszyklus auf das Testen erklären
- FL-2.1.2 (K1) Sie können gute Praktiken für das Testen, die für alle Softwareentwicklungszyklen gelten, wiedergeben
- FL-2.1.3 (K1) Sie können die Beispiele für Test-First-Ansätze in der Entwicklung wiedergeben
- FL-2.1.4 (K2) Sie können die möglichen Auswirkungen von DevOps auf das Testen zusammenfassen
- FL-2.1.5 (K2) Sie können Shift-Left erklären
- FL-2.1.6 (K2) Sie können den Einsatz von Retrospektiven als Mechanismus zur Prozessverbesserung erklären

# Modelle des Softwareentwicklungslebenszyklus

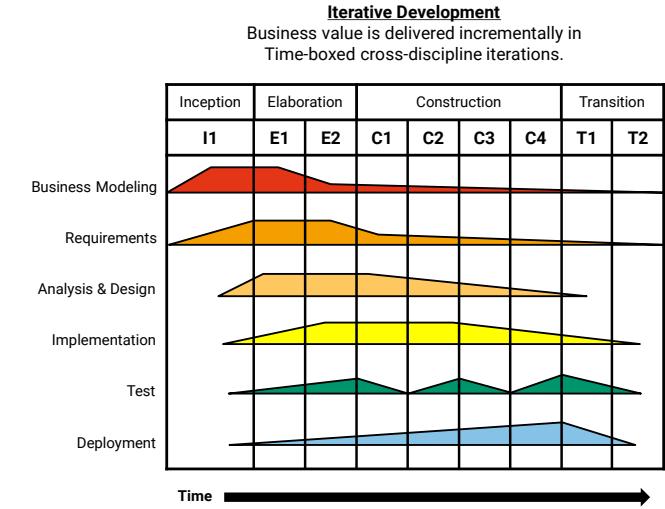
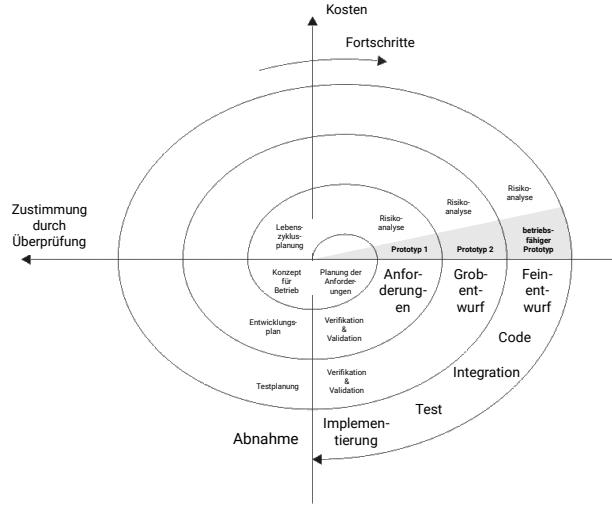
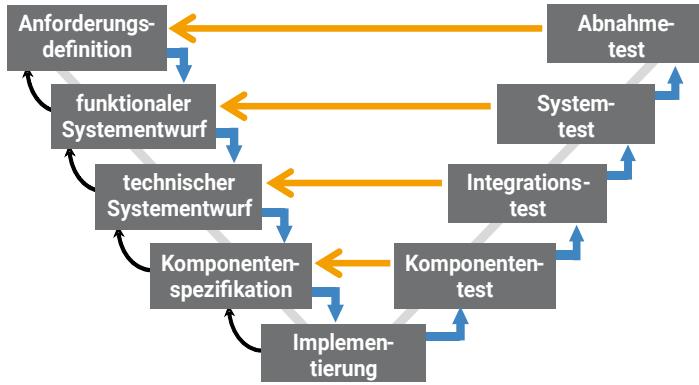
Ein Modell des Softwareentwicklungslebenszyklus (SDLC: Software Development Lifecycle) ist eine abstrakte, übergeordnete Darstellung des Softwareentwicklungsprozesses.

- Modelle des Softwareentwicklungslebenszyklus beschreiben
  - ... die Arten von Aktivitäten, die in jeder Phase durchgeführt werden.
  - ... wie diese Aktivitäten logisch und zeitlich zueinander in Beziehung stehen.

Verschiedene Softwareentwicklungslebenszyklus-Modelle erfordern verschiedene Testansätze.



# Softwareentwicklungslebenszyklus-Modelle (SDLC)



## sequenzielle Entwicklungsmodelle

- Wasserfallmodell
- V-Modell

## iterative Entwicklungsmodelle

- Spiralmodell
- Prototyping

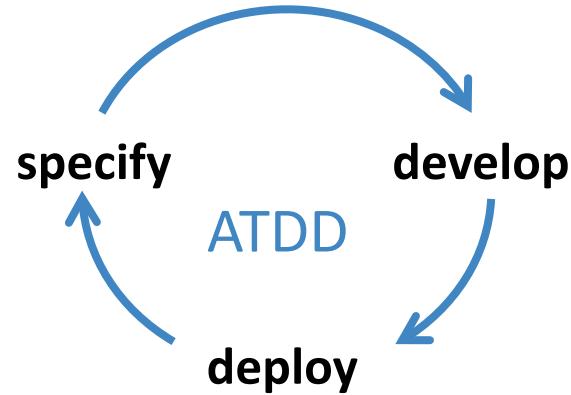
## inkrementelle Entwicklungsmodelle

- Unified Process



# Softwareentwicklungsmethoden und agile Praktiken I

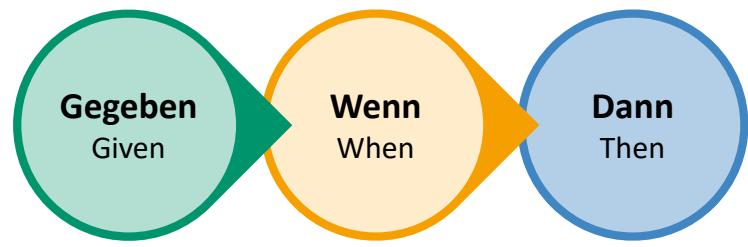
Einige Aktivitäten innerhalb von Softwareentwicklungsprozessen können auch durch detailliertere Softwareentwicklungsmethoden und agile Praktiken beschrieben werden.



## ATDD

### Abnahmetestgetriebene Entwicklung

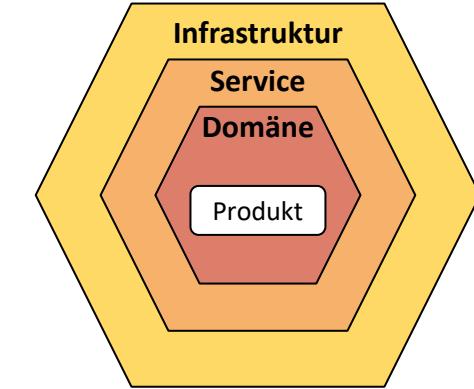
Ein auf Zusammenarbeit basierender Test-First-Ansatz, der Abnahmetests in der Fachsprache der Stakeholder definiert.



## BDD

### Verhaltensgetriebene Softwareentwicklung

Eine kollaborative Entwicklungs-vorgehensweise, bei der das Team den Schwerpunkt auf die Lieferung des erwarteten Verhaltens einer Komponente oder eines Systems für den Kunden legt, welches die Basis des Testens bildet

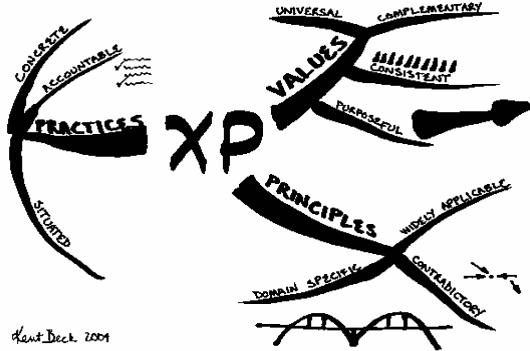


## DDD

### Domain-Driven Design

Beim DDD werden komplexe Zusammenhänge mithilfe klar abgegrenzter Domänen modelliert. Sie bilden Fachlichkeit und Fachlogik des Anwendungsgebietes ab.

# Softwareentwicklungsmethoden und agile Praktiken II



## XP

### Extreme Programming (XP)

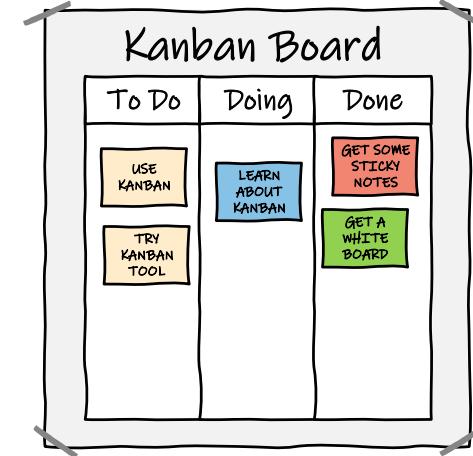
Eine agile Softwareentwicklungs-methodik zur Verbesserung der Softwarequalität und der Reaktionsfähigkeit auf sich ändernde Kundenanforderungen.



## FDD

### Feature Driven Development

Ein iterativ inkrementeller Softwareentwicklungsprozess, der mit Blick auf die Funktionalitäten mit Kundenwert (Features) betrieben wird. Feature-getriebene Entwicklung wird meist bei agiler Softwareentwicklung genutzt.



## Kanban

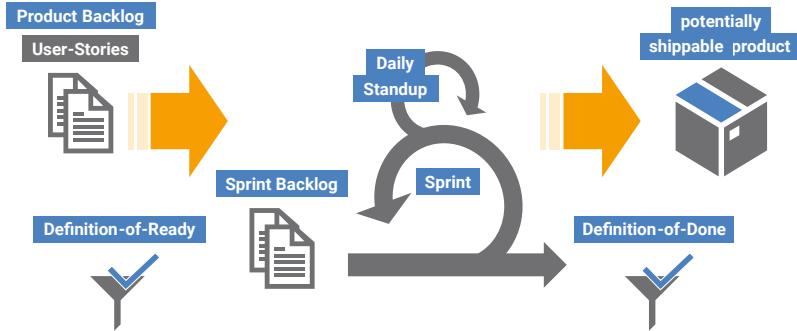
Ziel ist es, einen stetigen und geordneten Workflow zu etablieren. Wird oft mit anderen agilen Methoden wie Scrum kombiniert.

# Softwareentwicklungsmethoden und agile Praktiken III



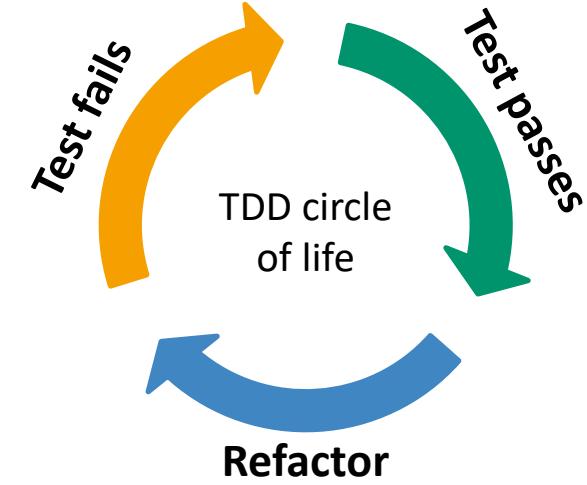
## Lean IT

Ein Managementansatz der Aufbau- und Ablauforganisation einer Organisation, in einem kontinuierlichen Verbesserungsprozess.



## Scrum

Ein Projektmanagement-framework für empirische Softwareentwicklung (agile).



## TDD

Test Driven Development  
Ein Softwareentwicklungsverfahren, bei der die Testfälle entwickelt und automatisiert werden und die Software dann schrittweise entwickelt wird, um diese Testfälle zu bestehen.



# Auswirkungen des Softwareentwicklungslebenszyklus auf das Testen

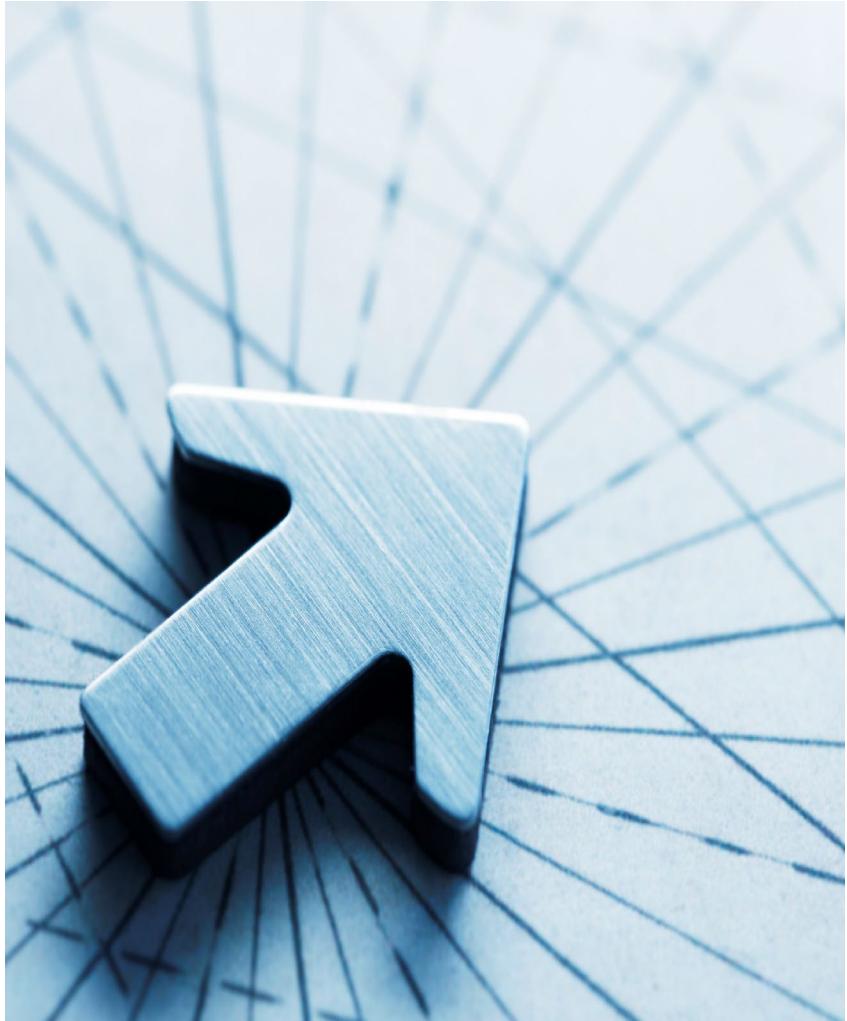
# SWE-Lebenszyklus-Modelle im Kontext

**Das Testen muss an das SWE-Lebenszyklus-Modell angepasst werden, um erfolgreich zu sein.**

Die Auswahl des Modells hat Auswirkungen auf

- Umfang und Zeitpunkt der Testaktivitäten (z. B. Teststufen und Testarten)
- Detaillierungsgrad der Testdokumentation
- Wahl der Testverfahren und des Testansatzes
- Umfang der Testautomatisierung
- Rolle und Aufgaben der Tester

# Sequenzielle Entwicklungsmodelle



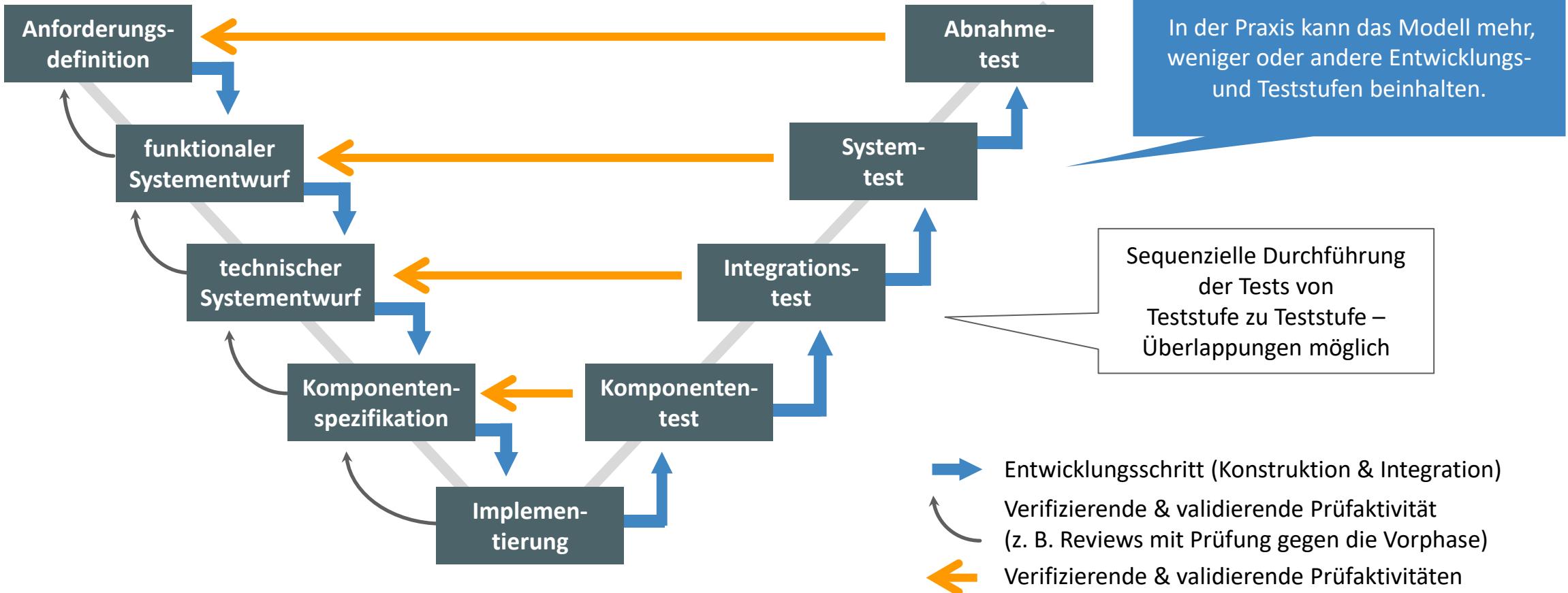
Beteiligung der Tester in den Anfangsphasen des SWE-Lebenszyklus meist bei ...

- Reviews der Anforderungen
- Testanalyse
- Testentwurf

Da der ausführbare Code meist in den späteren Phasen des SWE-Lebenszyklus erstellt wird, werden dynamische Tests oft erst später ausgeführt.

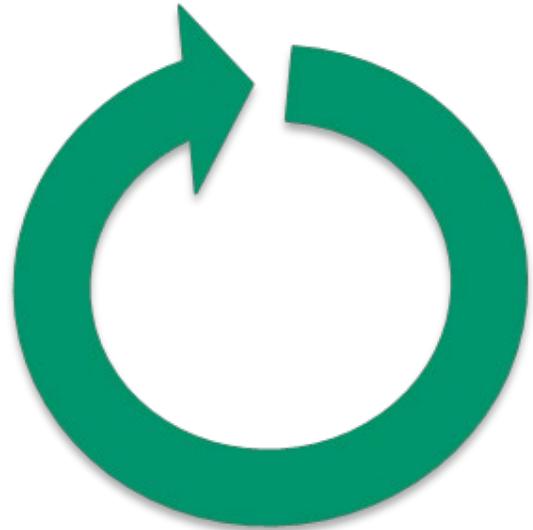


# Sequenzielle Softwareentwicklung – V-Modell



s.a: A.-P. Bröhl, Das V-Modell, Oldenbourg Verlag

# Iterative und inkrementelle Entwicklungsmodelle I



Oft wird davon ausgegangen, dass jede Iteration einen funktionierenden Prototyp oder ein Inkrement des Produkts liefert.

→ In jeder Iteration können statische Tests und dynamische Tests auf allen Teststufen durchgeführt werden.

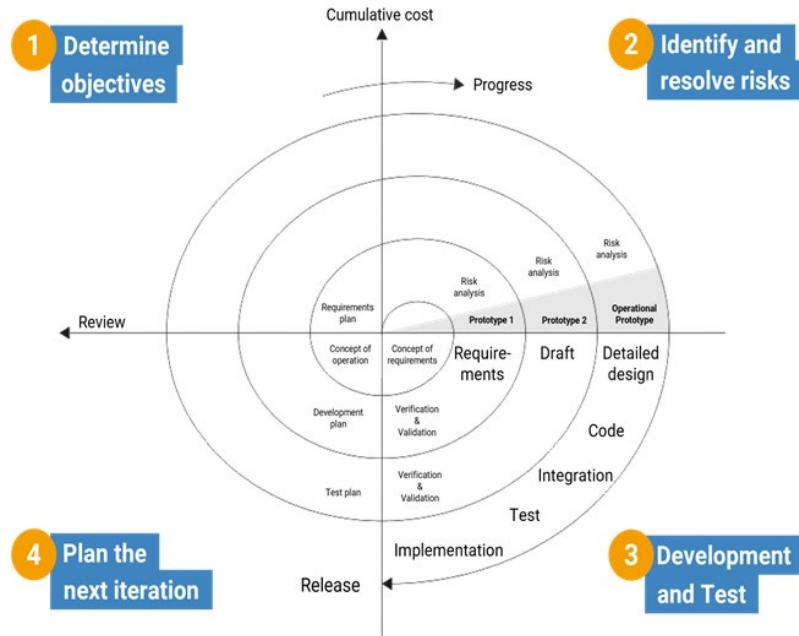
Häufige Lieferung von Inkrementen.

→ Erfordert schnelle Rückmeldungen und umfangreiche Regressionstests.



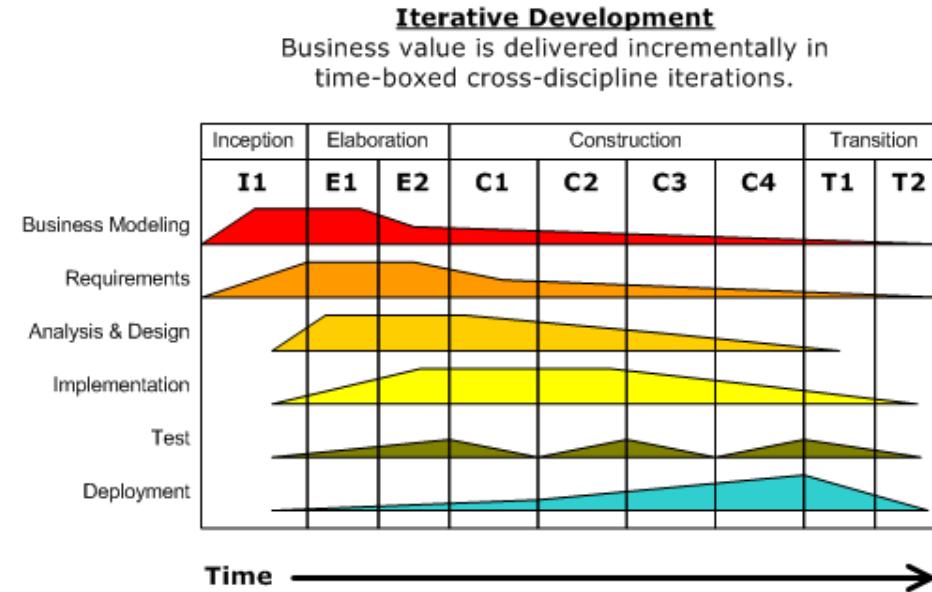
# Iterative und inkrementelle Entwicklungsmodelle II

## Iteratives Entwicklungsmodell: Spiralmodell



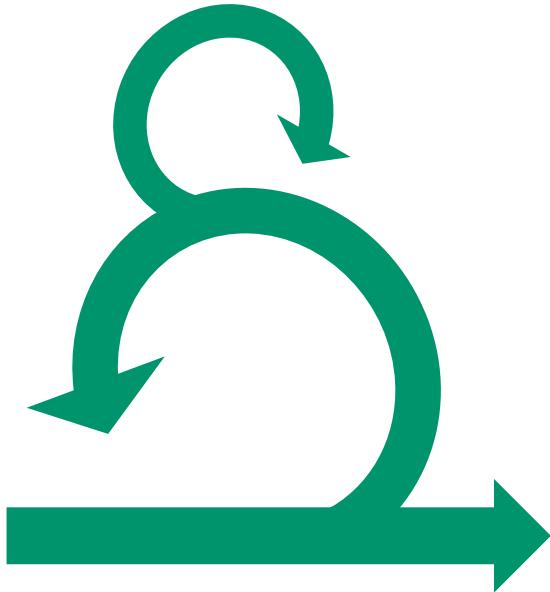
Von Conny, CC BY-SA 3.0,  
<https://commons.wikimedia.org/w/index.php?curid=3697528>

## Inkrementelles Entwicklungsmodell: Unified Process



Von Dutchguilder - Eigenes Werk, Gemeinfrei,  
<https://commons.wikimedia.org/w/index.php?curid=37249677>

# Agile Entwicklungsmodelle



Änderungen können sich während des gesamten Projekts ergeben.

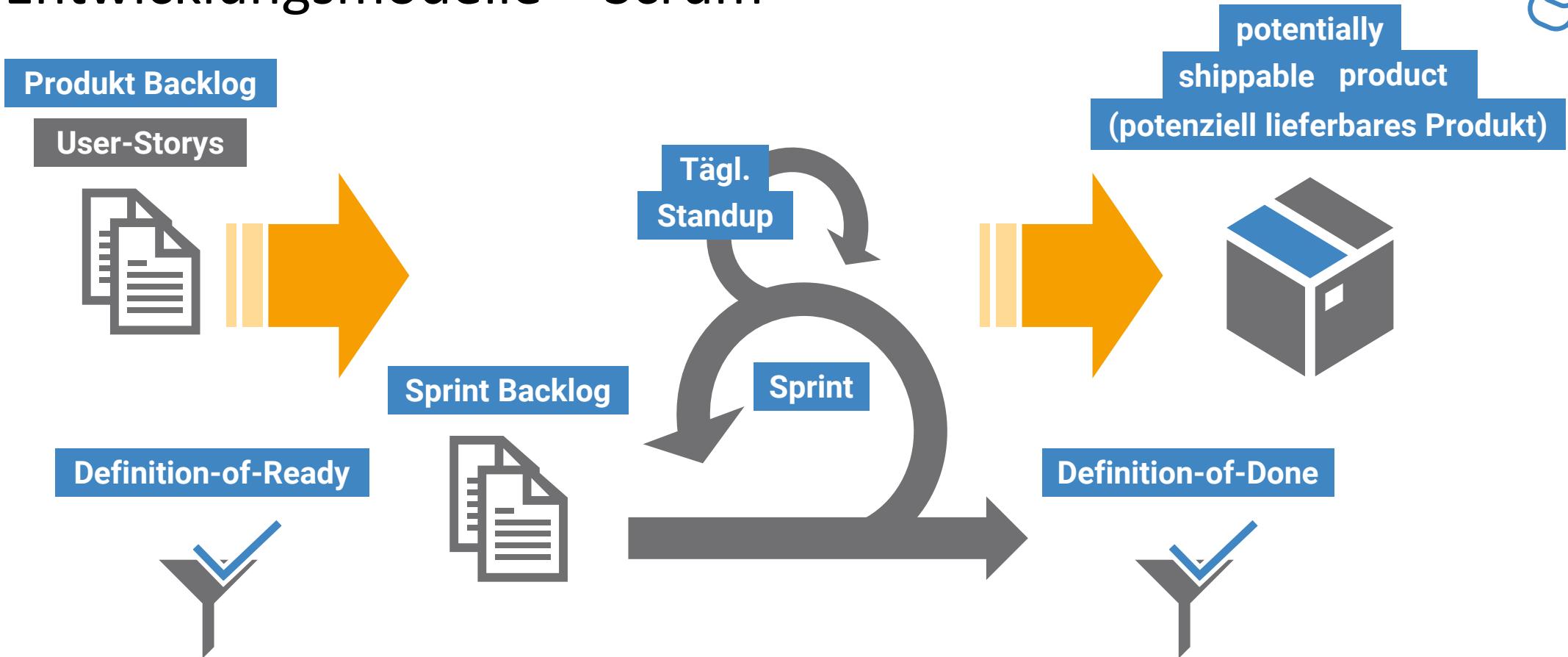
- Schlanke Dokumentation der Arbeitsergebnisse
- Umfassende Testautomatisierung, um Regressionstests zu erleichtern.

Der Großteil der manuellen Tests kann mit erfahrungsbasierteren Testverfahren durchgeführt werden.

- Keine umfangreiche Testanalyse und kein ausführlicher Testentwurf notwendig.



# Agile Entwicklungsmodelle – Scrum



- Jede Iteration ist eher kurz ([eine bis vier Wochen](#)).
- Die Inkremeante der Features sind entsprechend klein ([z. B. einige Verbesserungen und/oder zwei oder drei neue Features](#)).



## Gute Praktiken für das Testen

# Gute Praktiken für das Testen

## Zusammenarbeit

Für jede Softwareentwicklungsaktivität gibt es eine entsprechende Testaktivität.

→ Alle Entwicklungsaktivitäten unterliegen dem Qualitätsmanagement.

## Testziele

Unterschiedliche Teststufen haben spezifische und unterschiedliche Testziele.

- Der jeweilige Test ist angemessen und entsprechend umfassend.
- Redundanzen werden vermieden.

## Reviews

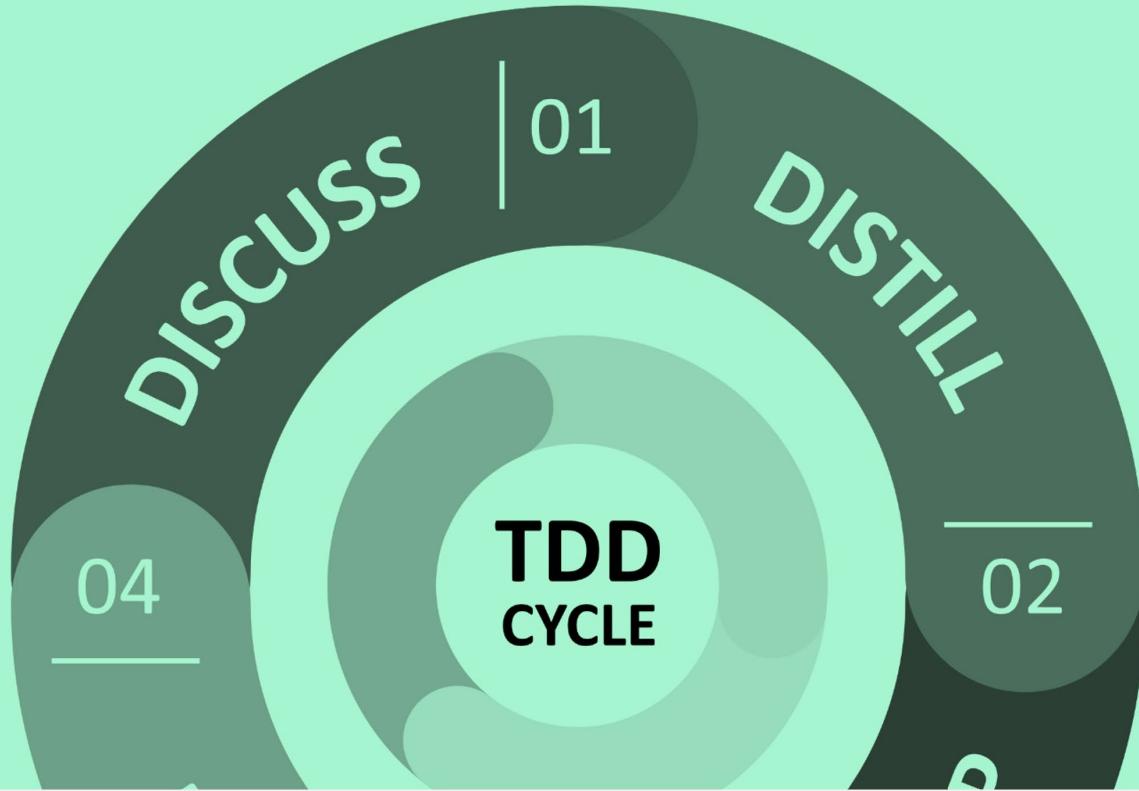
Tester werden in das Review von Arbeitsergebnissen einbezogen, sobald erste Entwürfe verfügbar sind.

- Frühes Testen und die Fehlerentdeckung können Shift-Left unterstützen.

## Frühes Testen

Testanalyse und Testentwurf für eine bestimmte Teststufe beginnen bereits in der entsprechenden Entwicklungsphase.

→ So wird beim Testen der Grundsatz des frühen Testens eingehalten.



## Testen als Treiber für die Softwareentwicklung

# Test-First-Ansätze

Test-First-Ansätze sind Entwicklungsansätze,  
bei denen Tests als Mittel zur Lenkung der Entwicklung definiert werden.

Jeder dieser Ansätze setzt das Prinzip des frühen Testens um und folgt Shift-Left.

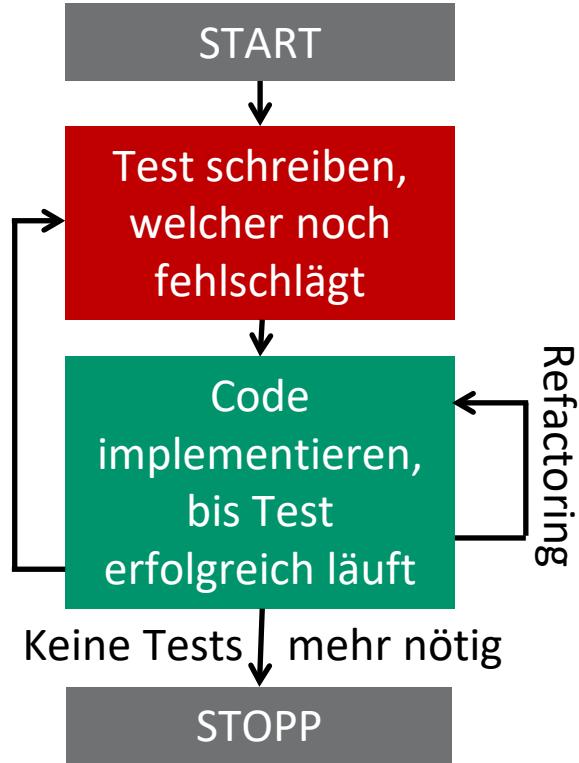
Die Tests werden definiert, bevor der Code geschrieben wird.

Um die Codequalitt bei zuknftigen nderungen am Code sicherzustellen, knnen die Tests als automatisierte Tests beibehalten werden.

Test-First-Ansätze unterstützen iterative Entwicklungsmodelle.

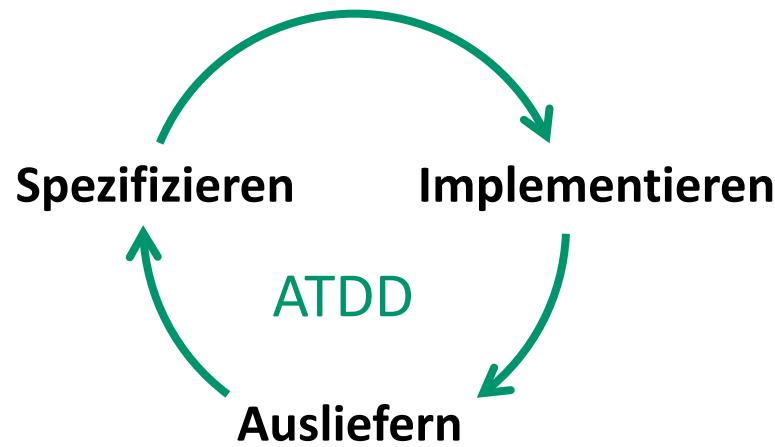
## Test-First-Ansätze sind:

# TDD – Testgetriebene Entwicklung



- Lenkt die Codierung durch Testfälle (unter Verzicht auf einen umfangreichen Softwareentwurf).
- Zuerst werden Tests geschrieben, dann wird der Code geschrieben, um die Tests zu erfüllen, und dann werden Tests und Code überarbeitet (Refactoring).

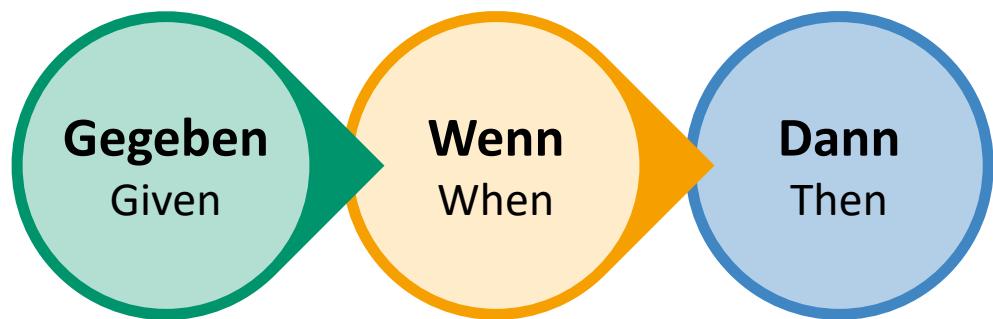
# ATDD – Abnahmetestgetriebene Entwicklung



Leiten Tests aus Akzeptanzkriterien als Teil des Systementwurfs ab.

Tests werden geschrieben, bevor der Teil der Anwendung entwickelt wird.

# BDD – Verhaltensgetriebene Entwicklung



Drückt das gewünschte Verhalten einer Anwendung mit Testfällen aus.

Verwendet eine einfache, natürlichsprachliche Form, die von Stakeholdern leicht zu verstehen ist – meist unter Verwendung des Gherkin-Formats.

Die Testfälle sollten dann automatisch in ausführbare Tests übersetzt werden.

# Akzeptanzkriterien im Gherkin Format



## User-Story 4: Darlehensbetrag, Zinssatz und Ratenhöhe anzeigen

Als Kunde möchte ich mir die monatliche Ratenhöhe der Autofinanzierung anzeigen, um zu beurteilen, ob ich mir die Autofinanzierung leisten kann.

Bei Eingabe von Anzahlung und Laufzeit in Monaten wird der Darlehensbetrag, Zinssatz und die Ratenhöhe ermittelt und angezeigt.

### Akzeptanzkriterien:

language: de  
Funktionalität: Virtual Showroom II - Finanzierung

Vorbedingungen:  
Gegeben seien ein Kunde ist in "VIRTUAL SHOWROOM II" eingeloggt  
Und ein Fahrzeug ist konfiguriert #Testfahrzeug 1= 21.600 Euro  
Und der Dialog „VSR-II EASY Finance - Ratenkredit“ ist geöffnet

Szenario 1: Finanzierung ermitteln  
Wenn Eingabe von <Anzahlung>  
Und Eingabe von <Laufzeit in Monaten>

Dann wird der <Darlehensbetrag> ermittelt und angezeigt  
Und die <monatliche Ratenhöhe> wird ermittelt und angezeigt  
Und der <Zinssatz> wird ermittelt und angezeigt

Beispiele:  

Anzahlung	Laufzeit in Monaten	Darlehensbetrag	mtl. Ratenhöhe	Zinssatz
1	10	21.599	2.289,49	6
21.599	20	1	0,05	3
10.000	40	11.600	295,80	2

Szenario 2: Finanzierung - ungültige Anzahlung  
Wenn Eingabe von Anzahlung ist „0,5“  
Dann wird die Fehlermeldung "Bitte geben Sie einen Betrag zwischen Kaufpreis minus 1,00 € und 21.600 € ein." angezeigt

Szenario 3: Finanzierung - ungültige Laufzeit in Monaten  
Wenn Eingabe von Laufzeit in Monaten ist „5“  
Dann wird die Fehlermeldung "Bitte geben Sie eine Laufzeit zwischen 6 und 48 Monaten ein." angezeigt

Hinweis: Konkrete Testwerte für die Übung 3-1: Äquivalenzklassen und Übung 3-2: Längenprüfung Preis: Testfahrzeug

language: de

Funktionalität: Virtual Showroom II - Finanzierung

### Vorbedingungen:

Gegeben seien ein Kunde ist in "VIRTUAL SHOWROOM II" eingeloggt  
Und ein Fahrzeug ist konfiguriert #Testfahrzeug 1= 21.600 Euro  
Und der Dialog „VSR-II EASY Finance - Ratenkredit“ ist geöffnet

### Szenarien 1: Finanzierung ermitteln

Wenn Eingabe von <Anzahlung>  
Und Eingabe von <Laufzeit in Monaten>

Dann wird der <Darlehensbetrag> ermittelt und angezeigt  
Und die <monatliche Ratenhöhe> wird ermittelt und angezeigt  
Und der <Zinssatz> wird ermittelt und angezeigt

### Beispiele:

Anzahlung	Laufzeit in Monaten	Darlehensbetrag	mtl. Ratenhöhe	Zinssatz
1	10	21.599	2.289,49	6
21.599	20	1	0,05	3
10.000	40	11.600	295,80	2

siehe Seite 65 im Übungsheft

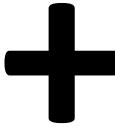


# DevOps und Testen

# Was ist DevOps? I



**Developers & Testers**



**IT Operations**

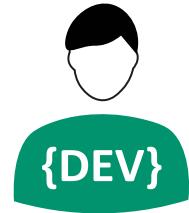
DevOps ist ein organisatorischer Ansatz, der darauf abzielt, Synergien zu schaffen.

→ Entwicklung (einschließlich Testen) und Betrieb arbeiten zusammen,  
um gemeinsame Ziele zu erreichen.

DevOps erfordert einen Kulturwandel innerhalb eines Unternehmens um ...

→ die Kluft zwischen Entwicklung (und Test) und Betrieb zu überbrücken und  
die jeweiligen Aufgaben gleichwertig zu behandeln.

# Was ist DevOps? II



**Developers & Testers**



**IT Operations**

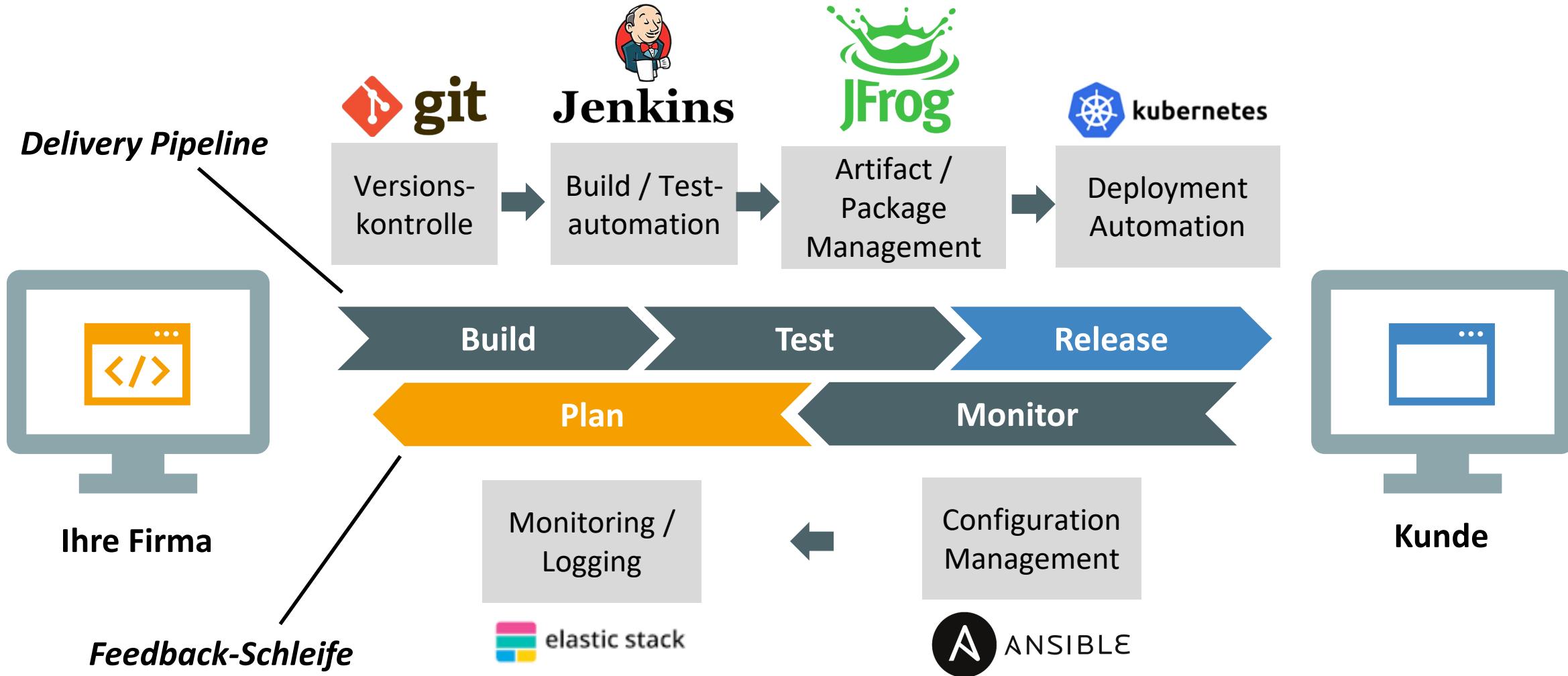
DevOps fördert Teamautonomie, schnelle Rückmeldungen, integrierte Werkzeugketten und technische Praktiken.

→ Kontinuierliche Integration (CI) und kontinuierliche Auslieferung (CD)

DevOps ermöglicht dem Team, qualitativ hochwertigen Code schneller zu erstellen, zu testen und freizugeben.

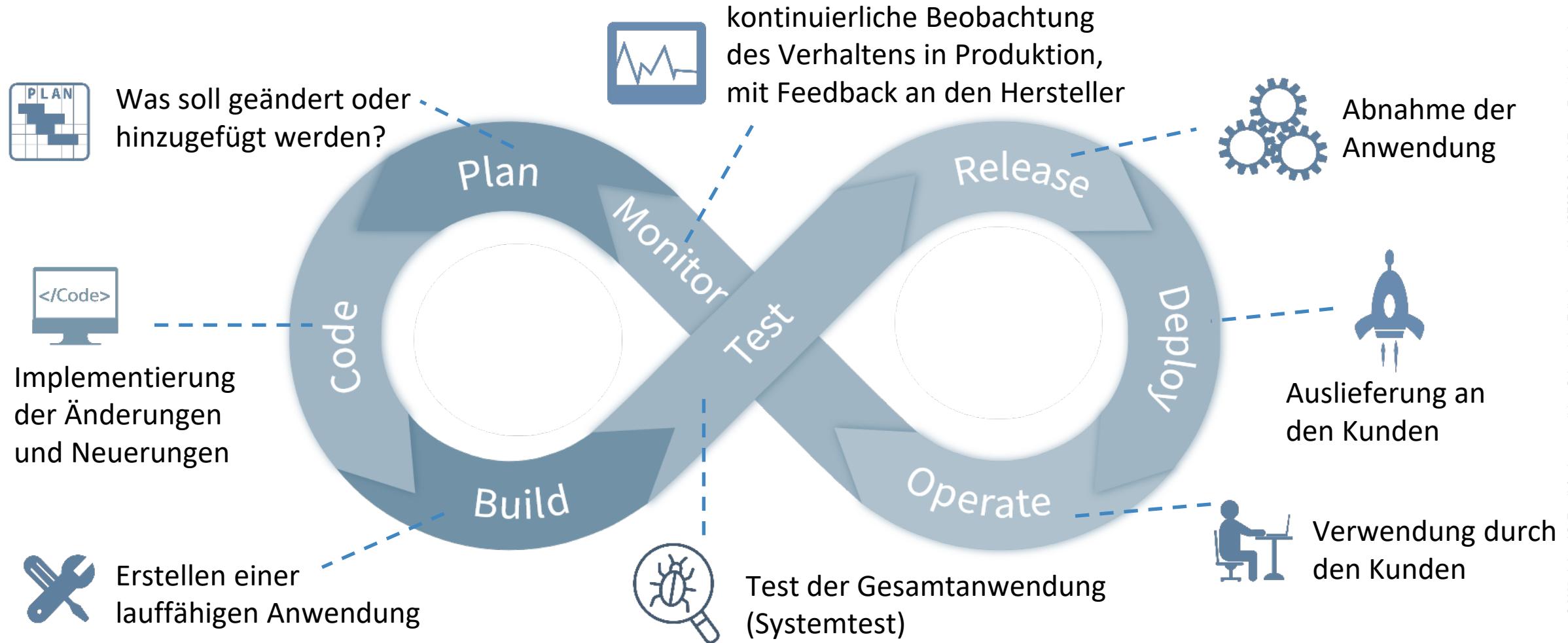
→ DevOps-Auslieferungskette (Delivery-Pipeline)

# DevOps – Umgebung





# Beispiel DevOps und kontinuierliches Testen



# DevOps – Vorteile I



## DevOps fördert ...

- Teamautonomie
- schnelle Rückmeldung (Fast Feedback)
- integrierte Werkzeugketten
- kontinuierliche Integration (CI = "Continuous Integration")
- kontinuierliche Auslieferung (CD = "Continuous Delivery")

→ Ermöglicht qualitativ hochwertigen Code durch eine DevOps-Auslieferungskette (Delivery-Pipeline) schneller zu erstellen, zu testen und freizugeben.

# DevOps – Vorteile II



## DevOps – Vorteile aus Sicht des Tests

- Schnelle Rückmeldung (Fast Feedback) zu Codequalität und Änderungen
- CI fördert Shift-Left (Komponententests und statische Analyse)
- Förderung automatisierter Prozesse wie CI/CD  
→ erleichtert den Aufbau stabiler Testumgebungen
- Erhöhter Fokus auf nicht-funktionale Qualitätsmerkmale
- Auslieferungskette reduziert durch Automatisierung die sich wiederholenden manuellen Tests
- Minderung des Regressionsrisikos durch automatisierte Regressionstests

# DevOps – Herausforderungen



## Risiken und Herausforderungen

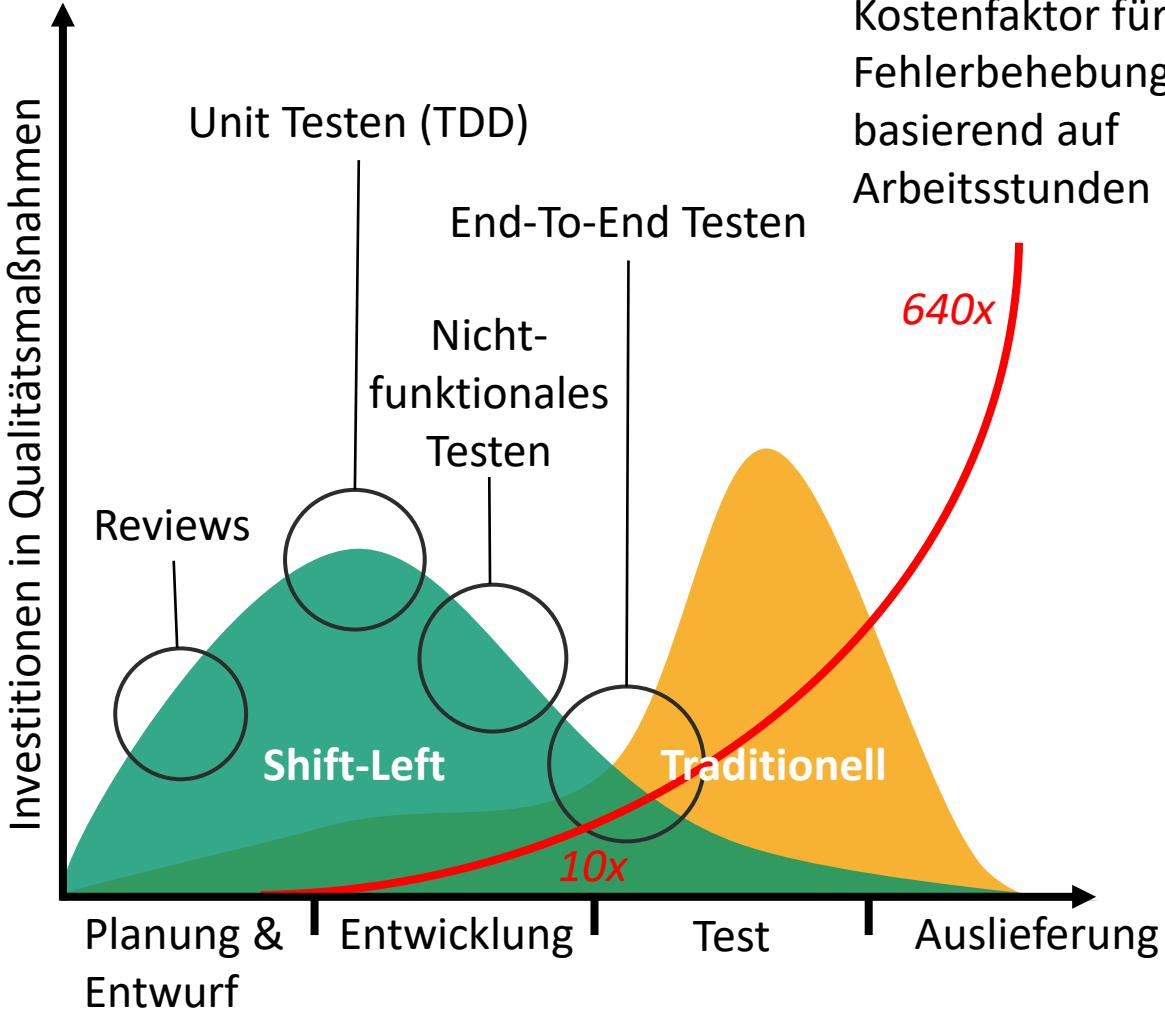
- Auslieferungskette muss definiert und etabliert werden
- CI/CD Werkzeuge müssen eingeführt und gewartet werden
- Testautomatisierung fordert Ressourcen und kann schwierig einzurichten und zu warten sein

### ➤ Achtung!

Auch bei einem hohen Automatisierungsgrad von Tests kann auf manuelle Tests (insbesondere aus Benutzerperspektive) nicht verzichtet werden.

# Shift-Left

# Shift-Left



Kostenfaktor für die Fehlerbehebung basierend auf Arbeitsstunden

- Bearbeitung von Aufgaben in der Prozesskette möglichst zeitlich nach vorne verlagern.
- Shift-Left beim Test (dynamisch und statisch) ermöglicht es, Fehlerzustände so früh wie möglich im Entwicklungsprozess zu finden.
- Shift-Left ersetzt aber keine End-To-End Tests im späteren Verlauf.
- Die Etablierung von Shift-Left führt oft am Beginn des Prozesses zu mehr Kosten, spart aber später im Prozess mehr Kosten ein.
- Es ist wichtig, dass die Stakeholder von Shift-Left überzeugt sind und es annehmen.



# Shift-Left Testen – bewährte Verfahren

## Review von Spezifikationen aus Sicht der Tester

- Findet früh potenzielle Fehlerzustände, wie Mehrdeutigkeiten, Unvollständigkeiten und Inkonsistenzen

## Schreiben von Testfällen, bevor der Code geschrieben wird und Ausführen des Codes in einem Testrahmen während der Coderealisierung (TDD)

## Verwendung von Continuous Integration (CI) und noch besser auch Continuous Delivery (CD)

- statische Analyse des Quellcodes vor dem dynamischen Testen
- schnelle Rückmeldungen durch automatisierte Tests
- kontinuierliche Integration anstatt Big Bang

## Durchführung von nicht-funktionalen Tests, beginnend wenn möglich ab Komponententest

- zu späte nicht-funktionale Tests sind potentielle Projektrisiken



# Schlüsselbegriff – Shift-Left

## Shift-Left

Ein Ansatz zur Durchführung von Test- und Qualitätssicherungsaktivitäten so früh wie möglich im Softwareentwicklungslebenszyklus.

# Retrospektiven und Prozessverbesserung

# Retrospektiven I



**Testeffektivität?  
Qualität der Testfälle?  
Zufriedenheit des Teams?  
Qualität der Anforderungen?  
Zusammenarbeit?**

Teilnehmer (Tester, Entwickler, Architekten, Product Owner, Businessanalysten, ...) diskutieren:

- Was war erfolgreich und sollte beibehalten werden?
- Was war nicht erfolgreich und könnte verbessert werden?
- Wie können die Verbesserungen eingearbeitet und die Erfolge in Zukunft beibehalten werden?



**Retrospektiven können bei Bedarf,  
am Ende eines Projekts oder einer Iteration,  
oder bei einem Release-Milestein abgehalten werden.**

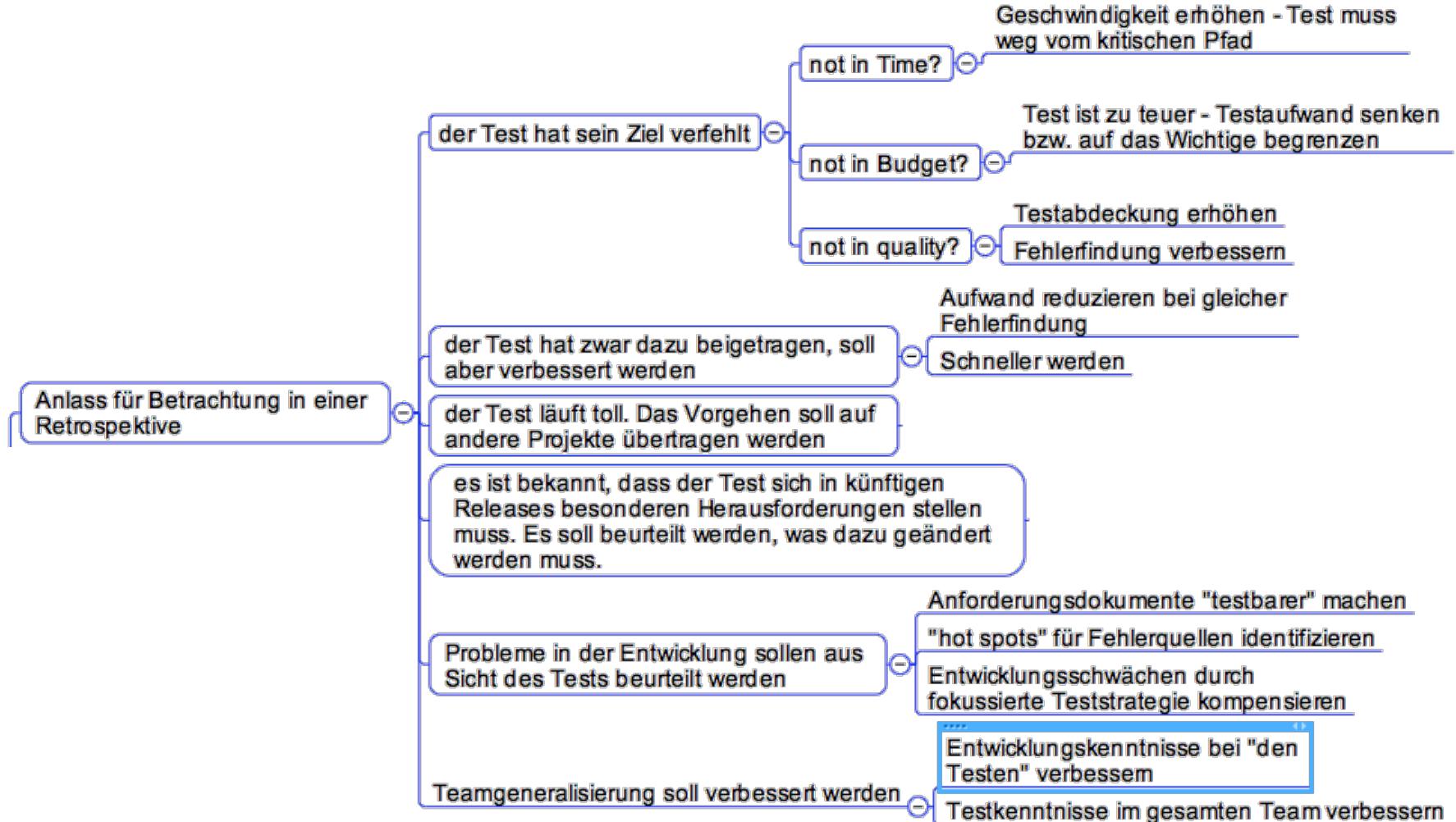
- Die Ergebnisse sollten festgehalten werden und sind normalerweise Teil des Testabschlussberichts.
- Retrospektiven sind entscheidend für die erfolgreiche Umsetzung der kontinuierlichen Verbesserung.
- Die empfohlenen Verbesserungen sollten weiterverfolgt werden.



## Typische Vorteile für das Testen

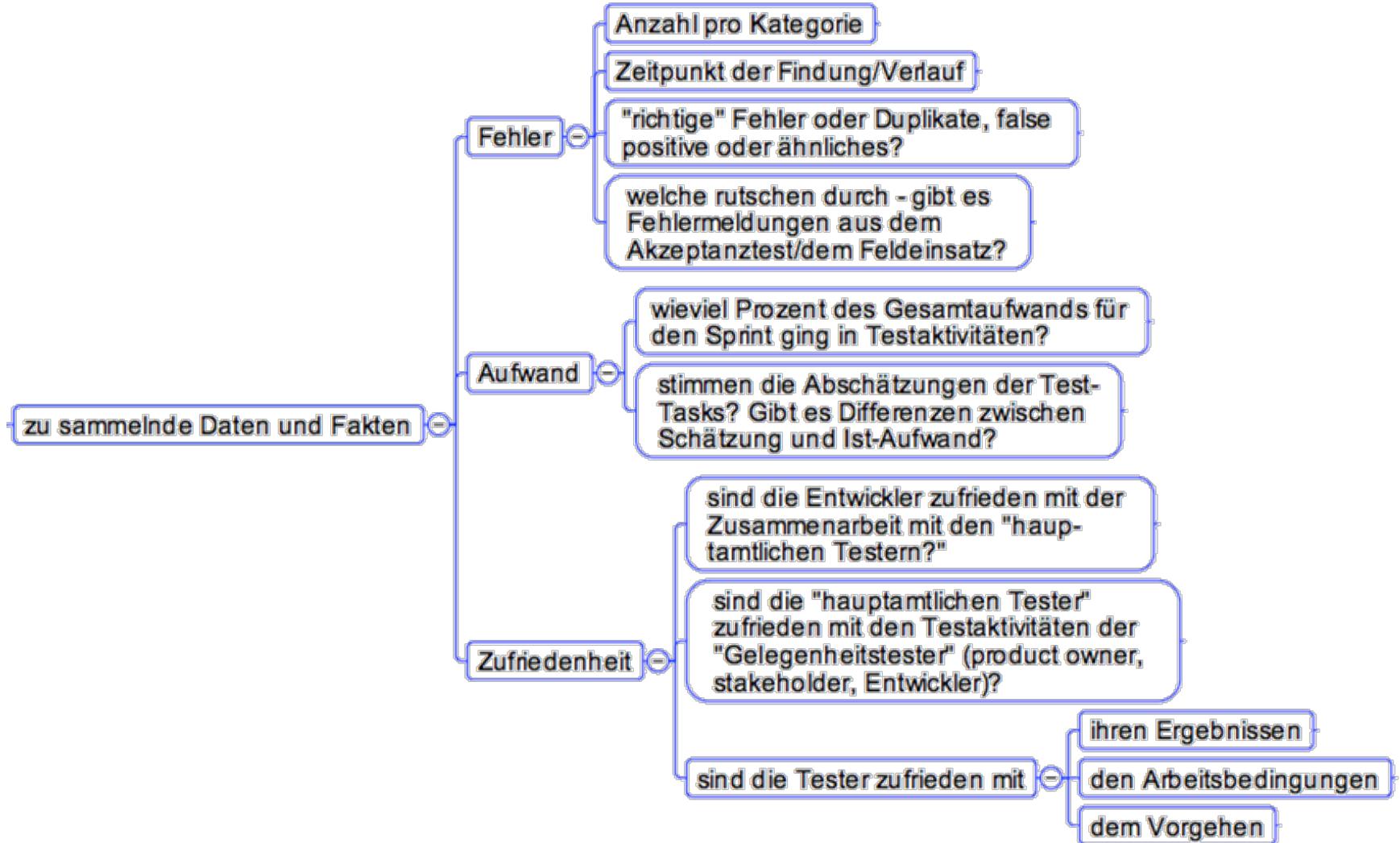
- Erhöhte Effektivität und Produktivität des Testens
  - z. B. Vorschläge zur Prozessverbesserung umsetzen
- Höhere Qualität der Testmittel
  - z. B. gemeinsames Review der Testprozesse
- Teamzusammenhalt und Lernen
  - z. B. Probleme ansprechen und Verbesserungspunkte vorschlagen
- Verbesserte Qualität der Testbasis
  - z. B. Mängel in Anforderungen werden angesprochen und behoben
- Bessere Zusammenarbeit zwischen Entwicklung und Test
  - z. B. Zusammenarbeit regelmäßig überprüfen und optimieren

# Mögliche Themen aus der Sicht des Testers I





# Mögliche Themen aus der Sicht des Testers II





# Auswirkungen des SDLC auf das Testen

Testen muss an den SDLC angepasst werden, um erfolgreich zu sein.

## Die Auswahl des SDLC hat Auswirkungen auf:

- Umfang und Zeitpunkt der Testaktivitäten (z. B. Teststufen und Testarten)
- Detaillierungsgrad der Testdokumentation
- Wahl der Testverfahren und des Testansatzes
- Umfang der Testautomatisierung
- Rolle und Aufgaben eines Testers

### sequenzielle Entwicklungsmodelle

- Tester früh einbinden
- dynamische Tests oft erst in späten Phasen

### iterative Entwicklungsmodelle und inkrementelle Entwicklungsmodelle

- Jede Iteration liefert ein fertiges Produkt
- Testen (statisch und dynamisch) auf allen Teststufen

### agile Softwareentwicklung

- Änderungen können sich während des gesamten Projektverlaufs ergeben
- umfassende Testautomatisierung und erfahrungsbasierte Tests
- schlanke Dokumentation



# Gute Praktiken für das Testen

## Gute Praktiken

- Für jede Softwareentwicklungsaktivität gibt es eine entsprechende Testaktivität
- Unterschiedliche Teststufen haben spezifische und unterschiedliche Testziele
- Die Testanalyse und der Testentwurf für eine bestimmte Teststufe beginnen bereits in der entsprechenden Entwicklungsphase des SDLC-Modells.
- Tester werden in das Review von Arbeitsergebnissen einbezogen, sobald Entwürfe dieser Dokumentation verfügbar sind



# Test-First-Ansatz

- Entwicklungsansatz, bei dem Tests als Mittel zur Lenkung der Entwicklung definiert werden
- Tests werden definiert bevor der Code geschrieben wird
- Test-First folgt Shift-Left

## Beispiele

- Testgetriebene Entwicklung (TDD)
- Abnahmetestgetriebene Entwicklung (ATDD)
- Verhaltensgetriebene Entwicklung (Behavior-Driven Development, BDD)



# Testen in DevOps und Shift-Left

## Auswirkungen von DevOps auf Testen

- schnelle Rückmeldung zu Codequalität und negativen Auswirkungen von Änderungen
- Shift-Left-Ansatz im Test
- Förderung automatisierter Prozesse wie CI/CD → erleichtert den Aufbau stabiler Testumgebungen
- Erhöhter Fokus auf nicht-funktionale Qualitätsmerkmale
- Automatisierung reduziert den Bedarf an sich wiederholenden manuellen Tests
- Risiko der Regression wird durch automatisierte Regressionstests minimiert

## Shift-Left

**Shift-Left** = Mit dem Testen wird früher begonnen

- Review von Spezifikationen aus Sicht der Tester
- Testfälle schreiben bevor Code erstellt wird
- schnelle Rückmeldungen durch automatisierte Tests
- Statische Analyse während der Entwicklung
- nicht-funktionale Tests so früh wie möglich ausführen (schon im Komponententest).



# Retrospektiven und Prozessverbesserung

## Zeitpunkt

- am Ende eines Projekts oder einer Iteration
- bei einem Release-Meilenstein
- bei Bedarf

→ Zeitpunkt und Organisation hängen von dem jeweiligen SDLC-Modell ab

## Teilnehmer, z. B.

- |               |                     |
|---------------|---------------------|
| ■ Tester      | ■ Product Owner     |
| ■ Entwickler  | ■ Businessanalysten |
| ■ Architekten |                     |

## Diskussionspunkte

- Was sollte beibehalten werden?
- Was könnte verbessert werden?
- Wie können die Verbesserungen eingearbeitet werden?

## Vorteile für das Testen

- Erhöhte Effektivität / Effizienz des Testens
- Höhere Qualität der Testmittel
- Teamzusammenhalt und Lernen
- Verbesserte Qualität der Testbasis
- Bessere Zusammenarbeit

## 2 – Testen während des Softwareentwicklungslebenszyklus

1. Testen im Kontext eines Softwareentwicklungslebenszyklus
2. Teststufen und Testarten
3. Wartungstest



## 2.2 Teststufen und Testarten

- FL-2.2.1 (K2) Sie können die verschiedenen Teststufen unterscheiden
- FL-2.2.2 (K2) Sie können die verschiedenen Testarten unterscheiden
- FL-2.2.3 (K2) Sie können Fehlernachtests von Regressionstests unterscheiden

# Teststufen

# Teststufen I

Eine Teststufe besteht aus Gruppen von Testaktivitäten, die gemeinsam organisiert und verwaltet werden.

Jede Teststufe ist mindestens eine Instanz des Testprozesses.

Die Testaktivitäten beziehen sich auf die Software einer bestimmten Entwicklungsphase:

- Einzelne Einheiten oder Komponenten
- Vollständige Systeme
- Systeme von Systemen

Die Teststufen stehen mit anderen Aktivitäten innerhalb des SDLC in Verbindung.

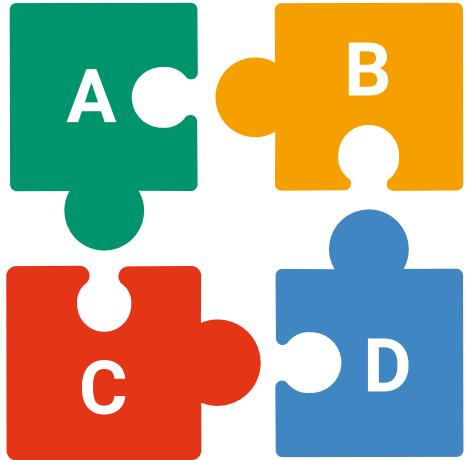
- In **sequenziellen Entwicklungsmodellen** sind die Teststufen oft so definiert, dass die Endekriterien einer Stufe Teil der Eingangskriterien für die nächste Stufe sind.
- In **iterativen Entwicklungsmodellen** können sich Entwicklungsaktivitäten über mehrere Teststufen erstrecken oder sich zeitlich überschneiden.

# Teststufen II

Um Überschneidungen von Testaktivitäten zu vermeiden,  
werden Teststufen durch die folgenden Attribute unterschieden:

- Testobjekt
- Testziele
- Testbasis
- Fehlerzustände und Fehlerwirkungen
- Vorgehensweise und Verantwortlichkeiten

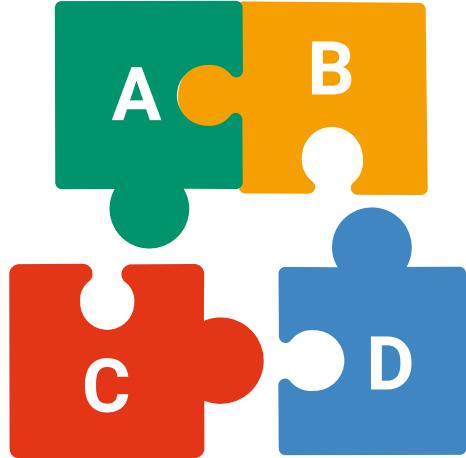
# Komponententest



## Komponententest (auch Unitests genannt)

- konzentriert sich auf isolierte Komponenten
- erfordert oft spezifische Werkzeugunterstützung, wie Testrahmen oder Unitest-Frameworks
- wird normalerweise von Entwicklern in der Entwicklungsumgebung durchgeführt

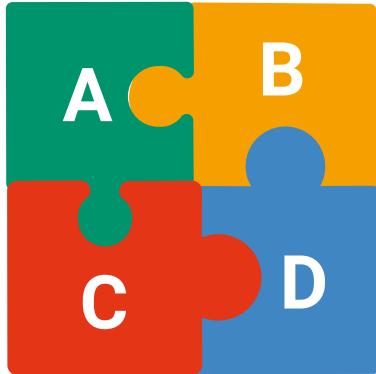
# Komponentenintegrationstest



## Komponentenintegrationstest

- konzentriert sich auf Schnittstellen und Interaktionen zwischen Komponenten
- ist stark von der Integrationsstrategie abhängig, z. B. Bottom-up, Top-down, Big-Bang

# Systemtest

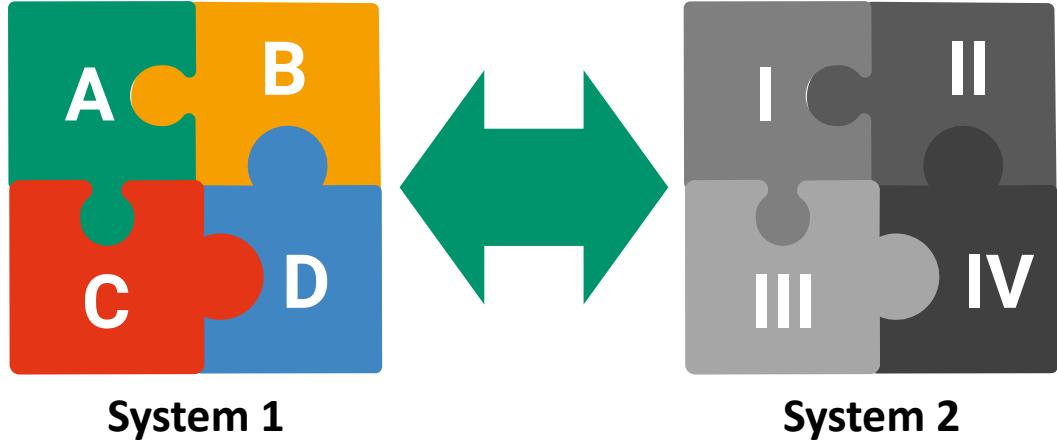


## Systemtest ...

- kann von einem unabhängigen Testteam durchgeführt werden.
- bezieht sich auf Anforderungsspezifikationen für das System.
- konzentriert sich auf das Gesamtverhalten und die Leistungsfähigkeit des gesamten Systems oder Produkts.
- umfasst häufig funktionale Tests von End-To-End-Aufgaben.
- umfasst häufig nicht-funktionale Tests.\*

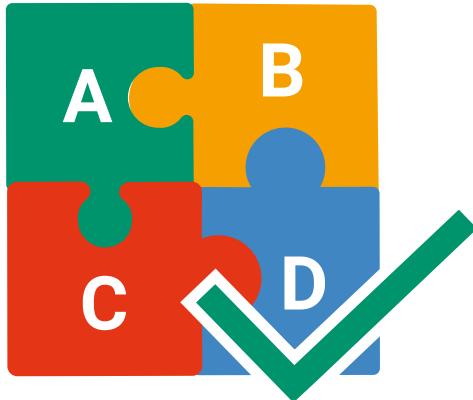
\* Bei einigen nicht-funktionalen Qualitätsmerkmalen ist es jedoch besser, sie an einem vollständigen System in einer repräsentativen Testumgebung zu testen (z. B. Gebrauchstauglichkeit). Die Verwendung von Simulationen von Teilsystemen ist ebenfalls möglich.

# Systemintegrationstest



## Systemintegrationstest

- konzentriert sich auf die Schnittstellen zwischen dem System unter Test und anderen Systemen und externen Diensten
- erfordert geeignete Testumgebungen, vorzugsweise entsprechend der Betriebsumgebung



## Abnahmetest

- konzentriert sich auf die Validierung und den Nachweis der Einsatzfähigkeit, d.h., das System muss die Geschäftsanforderungen der Benutzer erfüllen
- idealerweise sollten Abnahmetests von den vorgesehenen Benutzern durchgeführt werden
- Die wichtigsten Formen von Abnahmetests sind:
  - Benutzerabnahmetest (User Acceptance Testing, UAT)
  - betrieblicher Abnahmetest
  - vertraglicher Abnahmetest
  - regulatorischer Abnahmetest
  - Alpha-Test und Beta-Test

# Glossarvideo – Teststufe



IMBUS SOFTWARETEST

GLOSSAR VIDEOS



# Schlüsselbegriffe – Teststufe

## Teststufe

Eine spezifische Instanziierung eines Testprozesses.

## Systemtest

Eine Teststufe mit dem Schwerpunkt zu verifizieren, dass ein System als Ganzes die spezifizierten Anforderungen erfüllt.

## Komponententest

Eine Teststufe mit dem Schwerpunkt auf einer einzelnen Hardware- oder Software-komponente.

## System-integrationstest

Der Integrationstest von Systemen.

## Komponent-integrationstest

Der Integrationstest von Komponenten.

## Abnahmetest

Eine Teststufe mit dem Schwerpunkt zu bestimmen, ob ein System abgenommen werden kann.

## Integrationstest

Eine Teststufe mit dem Schwerpunkt auf dem Zusammenwirken zwischen Komponenten oder Systemen.

## Testobjekt

Das zu testende Arbeitsergebnis.

# Testarten

# Testarten

Eine Testart ist eine Gruppe von Testaktivitäten, basierend auf bestimmten Testzielen mit dem Zweck, eine Komponente oder ein System auf spezifische Merkmale zu prüfen.

In diesem Seminar werden die folgenden vier Testarten behandelt:

- Funktionaler Test
- Nicht-funktionaler Test
- Black-Box-Test
- White-Box-Test

Jede Testart kann auf jeder Teststufe angewandt werden.  
Der Schwerpunkt ist aber auf jeder Stufe anders.

Alle Testarten können unterschiedliche Testverfahren  
zur Ableitung von Testbedingungen und Testfällen verwenden.

# Funktionales – Nicht-funktionales Testen



## Klassifizierung der Qualitätsmerkmale nach ISO/IEC 25010

### Funktionale Qualitätsmerkmale



Funktionalität

### Nicht-funktionale Qualitätsmerkmale



Gebrauchstauglichkeit  
(Interaktionsfähigkeit)



Zuverlässigkeit



Sicherheit  
(engl. safety)



Kompatibilität



Performanz  
(Effizienz)



Übertragbarkeit  
(Flexibilität)



IT-Sicherheit  
(engl. security)



Wartbarkeit

# Funktionaler Test



Beim funktionalen Test werden die Funktionen bewertet, die eine Komponente oder ein System erfüllen soll.

Die Funktionen sind, „**was**“ das Testobjekt tun soll.

**Hauptziel** ist die Prüfung von

- funktionaler Vollständigkeit
- funktionaler Korrektheit
- funktionaler Angemessenheit

# Nicht-funktionaler Test



Nicht-funktionale Tests prüfen "**wie gut**" sich das System verhält.

Nicht-funktionale Tests sollten schon früh im Lebenszyklus beginnen (z. B. Komponententest, Reviews, ...).

Die späte Entdeckung von nicht-funktionalen Fehlerzuständen kann den Erfolg eines Projekts ernsthaft gefährden.

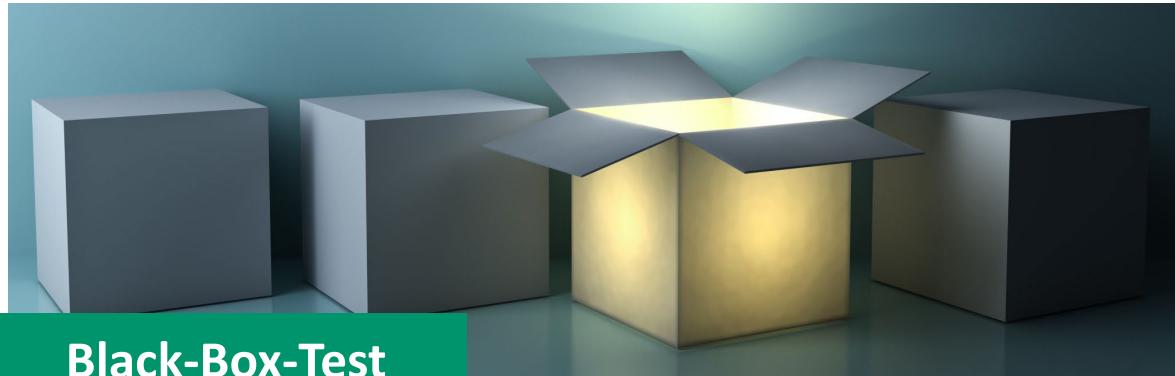
Viele nicht-funktionale Tests leiten sich von funktionalen Tests ab, da sie dieselben funktionalen Tests verwenden. Bei der Ausführung der Funktionen wird aber geprüft ob eine nicht-funktionale Bedingung erfüllt ist.

Z. B. Prüfung von Effizienz oder Übertragbarkeit

Nicht funktionale Tests können eine spezielle Testumgebung erfordern.

Z. B. ein Labor für Gebrauchstauglichkeitstests

# Black-Box-Test – White-Box-Test



## Black-Box-Test

Black-Box-Tests testen auf der Grundlage einer Analyse der Spezifikation der Komponente oder des Systems, die sich nicht auf die interne Struktur des Testobjekts bezieht.

### Hauptziel

Das Verhalten des Systems gegen seine Spezifikationen zu prüfen.



## White-Box-Test

White-Box-Test sind Tests, die auf Analyse der internen Struktur einer Komponente oder eines Systems basieren (z. B. Code, Architektur, Arbeitsabläufe und Datenflüsse).

### Hauptziel

Die ausgewählte Struktur durch Tests bis zu einem akzeptablen Grad überdecken.

# Komponententest – Testarten



The screenshot shows a car configuration application. At the top, there are language and user icons, and a 'Logout' button. Below that is a navigation bar with links: 'Grundmodell', 'Motor', 'Ausstattungspaket', 'Zusatzausstattung', 'Farbe', 'Zusammenfassung' (which is underlined), and '>'. The main area displays a configuration table:

Grundmodell: Modell 4 (Rassant Family)		18.500,00 €
Rabatt in %	0	-0,00 €
Motor: 2.0 Diesel		2.200,00 €
Ausstattungspaket: Keine:		0,00 €
Zusatzausstattung: Sportfederung:		900,00 €
Farbe: Hellblau:		0,00 €
Glanz:		0,00 €
<b>Gesamtpreis:</b>		<b>21.600,00 €</b>

On the right, there's a preview image of a blue car labeled 'Testfahrzeug 1' and 'Modell 4 (Rassant Family)'. Below the table, there's a 'Konfigurationsname' field containing 'Testfahrzeug 1' with a save and delete icon. At the bottom are three green buttons: 'Bestellung', 'Finanzierung', and 'Versicherung'.

## Funktionaler Test

Wird der Preis für ein Fahrzeug korrekt berechnet?

## Nicht-funktionaler Test

Entspricht das Layout der Eingabemaske den Vorgaben zur Gebrauchstauglichkeit?

## Black-Box Test

Entspricht das Verhalten der Buttons den Anforderungen?

## White-Box Test

Werden im Test alle Anweisungen im Code mindestens einmal aufgerufen?

# Komponentenintegrationstest – Testarten



Grundmodell: Modell 4 (Rassant Family) 18.500,00 €  
Motor: 2.0 Diesel 2.200,00 €  
Ausstattungspaket: Keine: 0,00 €  
Zusatzausstattung: Sportfederung: 900,00 €  
Farbe: Hellblau: 0,00 €  
Gesamtpreis: 21.600,00 €  
Testfahrzeug 1  
Modell 4 (Rassant Family)

Ratenkredit  
Leasing  
Kaufpreis: 21.600,00 €  
Darlehensbetrag: 21.600,00 €  
Anzahlung\*: Ratenhöhe: 0,00 €  
Zinssatz: 0 %  
Laufzeit in Monaten\*: Zurück zur Zusammenfassung Versicherung Bestellung

## Funktionaler Test

Wird der Fahrzeugpreis korrekt von VSR-II DREAMCAR nach VSR-II EASYFINANCE übergeben?

## Nicht-funktionaler Test

Effizienz – Wie lange dauert die Übergabe der Daten?

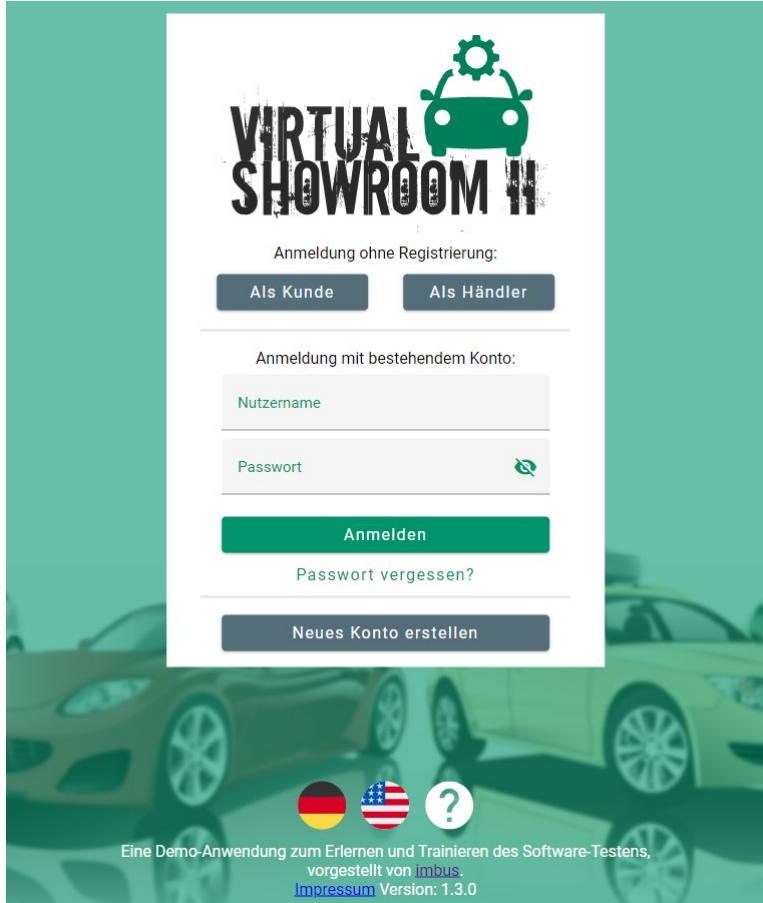
## Black-Box Test

Ist die Schnittstelle den Anforderungen entsprechend anpassbar?

## White-Box Test

Werden alle implementierten Fehlermeldungen korrekt verarbeitet?

# Systemtest – Testarten



## Funktionaler Test

Kann ein Fahrzeug konfiguriert, finanziert und bestellt werden?

## Nicht-funktionaler Test

Werden verschiedene Browser (Chrome, Firefox, Safari, Edge) unterstützt?

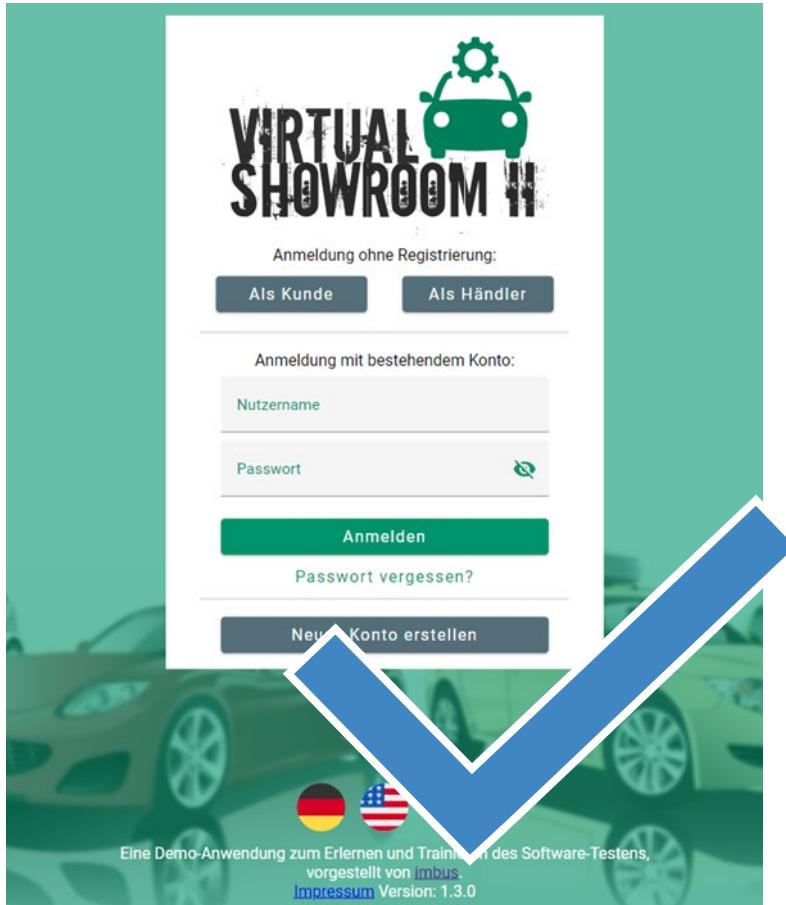
## Black-Box Test

Ist der Arbeitsablauf zum Kauf des Fahrzeugs so wie in den Epics und User-Storys beschrieben?

## White-Box Test

Wurden alle Webseiten-Dialoge wenigstens 1x getestet?

# Abnahmetest – Testarten



## Funktionaler Test

Sind die funktionalen Akzeptanzkriteriender User-Storys erfüllt?

## Nicht-funktionaler Test

Gebrauchstauglichkeitstest, um die Barrierefreiheit des Virtual Showroom II zu prüfen.

## Black-Box Test

Kann ein Fahrzeug nach Wunsch konfiguriert werden?

## White-Box Test

Sind alle möglichen Änderungen an den Konfigurationsdateien getestet?



# Schlüsselbegriffe – Testarten

## Testart

Eine Gruppe von Testaktivitäten basierend auf bestimmten Testzielen mit dem Zweck, eine Komponente oder ein System auf spezifische Merkmale zu prüfen.

## Black-Box-Test

Testen auf der Grundlage einer Analyse der Spezifikation der Komponente oder des Systems.

## funktionaler Test

Testen, welches durchgeführt wird, um die Erfüllung der funktionalen Anforderungen durch eine Komponente oder ein System zu bewerten.

## White-Box-Test

Ein Test, der auf der Analyse der internen Struktur einer Komponente oder eines Systems basiert.

## nicht-funktionaler Test

Testen, welches durchgeführt wird, um die Erfüllung der nicht-funktionalen Anforderungen durch eine Komponente oder ein System zu bewerten.



# Fehlernachtest und Regressionstest

# Fehlernachtest und Regressionstest

In der Regel werden Änderungen an einer Komponente oder einem System vorgenommen, um

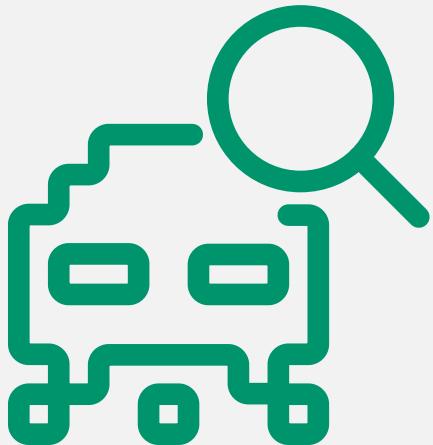
- eine Verbesserung durch Hinzufügen eines neuen Features zu erreichen.
- einen Fehlerzustand durch eine Korrektur zu beseitigen.

Das Testen sollte dann auch Fehlernachtests und Regressionstests beinhalten.

Fehlernachtests und/oder Regressionstests für das Testobjekt sind auf allen Teststufen erforderlich, wenn Fehlerzustände behoben und/oder Änderungen für diese Teststufen vorgenommen wurden.

# Fehlernachtest I

## Fehlernachtests bestätigen die Fehlerbehebung



**Der Fehlernachtest bestätigt,  
dass ein ursprünglicher Fehlerzustand erfolgreich behoben wurde.**

Je nach Risiko kann man die Fehlerkorrektur in der neuen Version der Software auf verschiedene Arten testen, z. B.:

- Ausführen aller Tests, die zuvor aufgrund des Fehlerzustands fehlgeschlagen sind
- Hinzufügen neuer Testfälle, um alle Änderungen zu überdecken, die zur Fehlerkorrektur erforderlich waren

Wenn das Budget oder die Zeit für die Fehlerkorrektur knapp ist:

Lediglich die Testschritte ausführen, die Fehler entdeckt haben, um zu prüfen, ob die Fehlerwirkung behoben wurde.

# VSR-II Fehlernachtest



- Autopilot Don'tCare
- Winterreifen (anstatt Sommerreifen)
- Navigationssystem
- Fußmatten
- Lederlenkrad
- Sound System OverDrive

## 1. Fehler finden

- Beim Test des VSR-II wird ein Fehler entdeckt
- Dazu wird der Fehlerbericht *"Zusatzausstattungen sind nicht alphabetisch sortiert"* in der Fehlerdatenbank erstellt

```
content: '';
content: none;
}
table {
    border-collapse: collapse;
    border-spacing: 0;
}
button, input, select, textarea { margin: 0;
:outline { outline: 0 }
a:link { -webkit-tap-highlight-color: #FF5E99 }
img, video, object, embed {
    max-width: 100%;
    height: auto!important;
}
iframe { max-width: 100% }
blockquote {
    font-style: italic;
    font-weight: normal;
    font-family: Georgia,Serif;
    font-size: 15px;
    padding: 0 10px 20px 27px;
    position: relative;
    margin-top: 25px;
}
small { font-size: 100% }
figure { margin: 10px 0 }
code, pre {
    font-family: monospace,consolas,sans-serif;
    font-weight: normal;
    font-style: normal;
}
pre {
    margin: 5px 0 20px 0;
    line-height: 1.3em;
    padding: 8px 10px;
    overflow: auto;
}
img {
    width: 100px;
    height: 100px;
    border-radius: 50px;
    border: 2px solid black;
}
```

## 2. Fehlerkorrektur

- Der gefundene Fehler wird in der Entwicklung analysiert und korrigiert
- Der Status des Fehlerberichts wird aktualisiert

- Autopilot Don'tCare
- Fußmatten
- Intelligentes Lichtsystem
- Lederlenkrad
- Navigationssystem
- Sound System OverDrive

## 3. Fehlernachtest

- Beim Fehlernachtest werden die Testschritte, die den Fehler entdeckt haben, erneut ausgeführt
- Der Fehler wurde korrigiert, der Fehlerbericht wird geschlossen



# Regressionstest I

## Werden benötigt, wenn Funktionen geändert oder hinzugefügt wurden



**Regressionstests bestätigen, dass eine Änderung (einschließlich einer bereits getesteten Fehlerbehebung) keine nachteiligen Folgen hat.**

Die nachteiligen Folgen könnten verschiedene Bereiche betreffen:

- die Komponente, an der die Änderung vorgenommen wurde
- andere Komponenten desselben Systems
- andere verbundene Systeme

Regressionstests können sich auf das Testobjekt und auch auf die Umgebung beziehen.

Es ist ratsam, zunächst eine Auswirkungsanalyse durchzuführen, um

- den Umfang der Regressionstests zu optimieren.
- festzustellen, welche Teile der Software betroffen sein könnten.

# Regressionstest II

**Werden benötigt, wenn Funktionen geändert oder hinzugefügt wurden**



## Regressionstestsuiten

- laufen viele Male
- die Anzahl der Testfälle nimmt im Allgemeinen mit jeder Iteration oder jedem Release zu
- eignen sich sehr gut für Testautomatisierung

Die **Automatisierung** von Regressionstests sollte früh im Projekt beginnen.

Wenn **CI/CD** eingesetzt wird (z. B. bei **DevOps**):

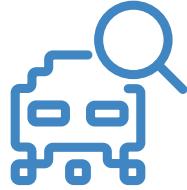
- automatisierte Regressionstests einbeziehen  
(je nach Situation auf verschiedenen Teststufen)

# VSR-II Regressionstest



Bei der **Fehlerkorrektur** wurde versehentlich ein **neuer Fehler eingebaut**. Dieser wird glücklicherweise **durch den Regressionstest entdeckt**.

Testprotokoll	
Regressionstestsuite VSR-II Systemtest, Version 2	
Testfall	Testergebnis
1: Login in VSR-II	OK
2: Fahrzeug konfigurieren	OK
3: Fahrzeugkonfiguration speichern	OK
4: Ratenkredit berechnen	OK
5: Fahrzeug bestellen	OK



Der **Fehler**  
*"Zusatzausstattungen sind nicht alphabetisch sortiert"*  
wurde beim manuellen Systemtest Version 3.0 **entdeckt** und **korrigiert**.

Testprotokoll	
Regressionstestsuite VSR-II Systemtest, Version 3	
Testfall	Testergebnis
1: Login in VSR-II	OK
2: Fahrzeug konfigurieren	OK
3: Fahrzeugkonfiguration speichern	NOK
4: Ratenkredit berechnen	OK
5: Fahrzeug bestellen	OK

VSR-II - Version 2

Entwicklung und Test

VSR-II - Version 3

Zeit

Am Ende der Teststufe werden die automatisierten **Regressionstests** in **Version 2** ausgeführt.

Am Ende der Teststufe werden die automatisierten **Regressionstests** in **Version 3** ausgeführt.



# Schlüsselbegriffe – Änderungsbezogenes Testen

## Fehlernachttest

Eine Art änderungsbezogenes Testen, das nach der Behebung eines Fehlerzustands durchgeführt wird, um zu bestätigen, dass eine Fehlerwirkung nicht mehr auftritt.

## Regressionstest

Eine Art änderungsbezogenes Testen, um festzustellen, ob in unveränderten Bereichen der Software Fehlerzustände eingebaut oder freigelegt wurden.



# Teststufen

	Komponententest	Integrationstest	Systemtest	Abnahmetest
Hauptfokus	Test einer einzelnen Hardware- oder Softwarekomponente	Zusammenwirken zwischen Komponenten oder Systemen	Verifizieren, dass ein System als Ganzes die spezifizierten Anforderungen erfüllt	Bestimmen, ob ein System abgenommen werden kann
Übliche Verantwortlichkeit	Entwickler	Komponenten-integrationstest: Entwickler  System-integrationstest: Tester	Unabhängige Tester	Kunden, Fachanwender, Product Owner, Betreiber eines Systems, andere Stakeholder

Jede Teststufe hat individuelle **Testziele, Testbasis, Testobjekte, Testvorgehensweisen, Verantwortlichkeiten sowie typische Fehlerzustände & -wirkungen.**



# Testarten

Jede Testart kann auf jeder Teststufe eingesetzt werden.

## Funktionaler Test

Prüft, „**was**“ das System tun sollte.

Tests prüfen die funktionale Vollständigkeit, die funktionale Korrektheit und die funktionale Angemessenheit.

## Nicht-funktionaler Test

Prüft, „**wie gut**“ das System sich verhält.

Tests prüfen nicht-funktionale Qualitätsmerkmale.

## Black-Box-Test

Testen auf der Grundlage einer **Analyse der Spezifikation** der Komponente oder des Systems.

## White-Box-Test

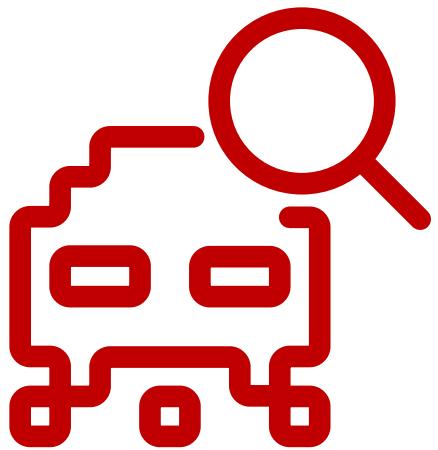
Tests, die auf der **Analyse der internen Struktur** einer Komponente oder eines Systems basieren.



# Fehlernachtest – Regressionstest

## Fehlernachtest

wird nach der Behebung eines Fehlerzustands durchgeführt, um zu bestätigen, dass eine Fehlerwirkung nicht mehr auftritt.



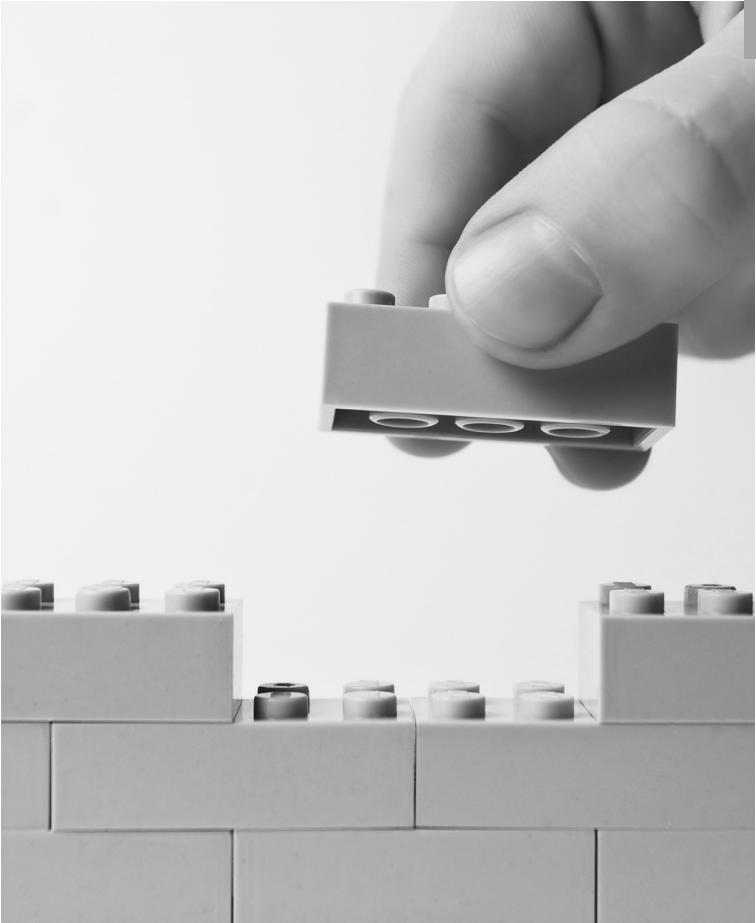
## Regressionstest

zeigt, ob in unveränderten Bereichen der Software Fehlerzustände eingebaut oder freigelegt wurden.



## 2 – Testen während des Softwareentwicklungslebenszyklus

1. Testen im Kontext eines Softwareentwicklungslebenszyklus
2. Teststufen und Testarten
3. Wartungstest



## 2.3 Wartungstest

- FL-2.3.1 (K2) Sie können den Wartungstest und dessen Auslöser zusammenfassen

# Wartungstest

# Was ist Wartung?



Wir sprechen von **Wartungstest**, wenn die zu testende Software bereits im Einsatz ist.

## Verschiedene **Kategorien von Wartung**

- Korrekturen
- Anpassung an Änderungen in der Umgebung
- Verbesserung der Leistung oder Wartbarkeit  
(Einzelheiten siehe ISO/IEC 14764)

**Wartung** kann Releases und/oder Bereitstellungen umfassen

- geplant
- ungeplant (Hotfixes)



# Wartungstests – Auslöser

## Änderungen

Release-basierte Änderungen

- [Test der geplanten Erweiterungen](#)
- [Test der Fehlerkorrekturen](#)
- [Test von Hotfixes](#)

## Upgrades oder Migrationen

Wenn von einer Betriebsumgebung auf eine andere migriert wird oder die Betriebsumgebung aktualisiert wird (Upgrade)

- [Tests der neuen Umgebung](#)
- [Tests der geänderten Software](#)

Wenn Daten aus einer anderen Anwendung in das zu wartende System migriert werden

- [Test der Datenkonvertierung](#)

## Außenbetriebnahme

Wenn eine Anwendung das Ende ihres Lebens erreicht und außer Betrieb genommen wird und

- falls lange Datenaufbewahrungsfristen erforderlich sind
  - wenn bestimmte Daten während der Archivierungszeit benötigt werden
- [Test der Datenarchivierung](#)
  - [Test des Wiederherstellungsverfahren nach der Archivierung](#)

# Wartungstest – Auswirkungsanalyse



Eine **Auswirkungsanalyse** kann **vor** einer Änderung durchgeführt werden, um auf der Grundlage der potenziellen Auswirkungen auf andere Bereiche des Systems zu entscheiden, ob die Änderung durchgeführt werden sollte.

**Testen der Änderungen im operativen System** umfasst:

- Bewertung des Erfolgs der Implementierung der Änderung
- Überprüfung auf mögliche nachteilige Folgen (Regressionstest) in Teilen des Systems, die unverändert bleiben  
(was in der Regel der größte Teil des Systems ist)

Der **Umfang des Wartungstests** hängt meist ab von:

- Grad des Risikos der Änderung
- Größe des bestehenden Systems
- Größe der Änderung



# Schlüsselbegriff – Wartungstest

## Wartungstest

Testen der Änderungen an einem laufenden System oder der Auswirkungen einer geänderten Umgebung auf ein laufendes System.



# Wartungstest

## Wartungstest prüft

- Änderungen an einem laufenden System
- Auswirkungen einer geänderten Umgebung auf ein laufendes System

Die **Auswirkungsanalyse** legt den Umfang für den Wartungstest fest.

## Auslöser für Wartungstest

- Änderungen
- Upgrades oder Migrationen der Betriebsumgebung
- Außerbetriebnahme