Prof. Dr. Knut Linke

DSBQSSP

QUALITÄTSSICHERUNG IM SOFTWAREPROJEKT

Sommersemester 2024

PRÄAMBEL

Danke an dieser Stelle an Herrn Prof. Dr. David Kuhlen und Frau Dr. Jessica Schiffmann für die bisherigen Vorarbeiten im Rahmen dieser Vorlesungsreihe.

VORSTELLUNG

Wer bin ich?

Knut Linke

- Studium WI & BWL in der EU & den USA
- Seit 2000 beruflich in der IT und mit Bezug Internet
- Professor für Informatik an der IU in Hannover

- Fun Fact: Taucht lieber als Fallschirm springen
- Vorkenntnisse: Produktmanagement, div. Tätigkeiten im Bereich Onlinemarketing, Digitalisierungsberater

VORSTELLUNG

Wer bist du?

- Person
- Arbeitgeber
- Funktion bzw. Aufgabe im Unternehmen
- Fun Fact
- Vorkenntnisse? Erwartungen?

00

INTERNATIONALE HOCHSCHULE

ORGANISATORISCHES UND RAHMENBEDINGUNG

HINWEIS

- Neben der Vermittlung von berufspraktischem Fachwissen soll Sie ein Studium zum selbstständigen Arbeiten befähigen. Ein Teil dieses selbstständigen Arbeitens besteht in der selbstständigen Organisation. Hierzu gehört auch, dass Sie sich rechtzeitig und umfänglich über die Anforderungen Ihrer Prüfungsleistungen informieren.

- Von Ihnen wird folglich erwartet, dass Sie sich über alle Regeln, Fristen, Vorschriften etc., die Sie zum erfolgreichen Bestehen Ihrer Studienleistung einhalten müssen, selbstständig informieren. Ihr Ansprechpartner diesbzgl. ist das Prüfungsamt.

HINWEIS

- Bitte beachten Sie alle Fristen, Regeln, Vorschriften, Leitfäden etc., die das Prüfungsamt ausgibt. Bei Fragen erkundigen Sie sich beim Prüfungsamt!

- In jedem Fall gelten die Bestimmungen des Prüfungsamtes! Die Ausführungen in diesem Foliensatz und in dieser Vorlesung berühren die Gültigkeit der Bestimmungen des Prüfungsamtes nicht.

KURSKONTEXT

- Zu den begleitenden Aktivitäten eines Softwareprozesses gehört die Qualitätssicherung.

- Aktivitäten zur Qualitätssicherung müssen qualitätsgesichert werden, damit die erstellen Software-Systeme in einer guten Qualität ausgeliefert werden können.

- Von Beginn an müssen erstellte Artefakte (Dokumente, Modelle, Programmcodes) qualitätsgesichert werden, denn je später ein Fehler in einem System erkannt wird, desto teurer wird die Fehlerbehebung.

KURSKONTEXT

- Der Kurs vermittelt Techniken und Vorgehensweisen zur begleitenden Qualitätssicherung: Beginnend bei der Anforderungsanalyse, über die Spezifikation, Architektur und Design, bis hin zur Implementierung.

KURSZIELE

Nach erfolgreichem Abschluss:

- kennen die Studierenden Motivation, Anwendungsfälle und Szenarien zu Aspekten des Qualitätsmanagements im Softwareprozess.

- kennen die Studierenden wichtige Begriffe und Grundlage für die Konzeption und Durchführung von Softwaretests.

- kennen die Studierenden Techniken und Methoden zum konstruktiven Qualitätsmanagement und können sie voneinander abgrenzen.

KURSZIELE

Nach erfolgreichem Abschluss:

- kennen die Studierenden Techniken und Methoden zum analytischen Qualitätsmanagement und können sie voneinander abgrenzen.

- kennen die Studierenden den allgemeinen Ablauf von Testaktivitäten und können für verschiedene Artefakte und Aktivitäten im Softwareprozess geeignete Methoden und Techniken zur Qualitätssicherung auswählen.

CHAT-GPT ETC.

Ziel ist das eigenständige reflektieren der Aufgaben im Kontext der Vorlesungsunterlagen.

Es steht Ihnen frei Chat-GPT oder ähnliche Systeme zur Unterstützung, z. B. im Kontext von Begriffsklärungen oder Themenerweiterungen zu nutzen und passend einzubringen.

Denken Sie daran, dass die Inhalte zielgerichtet und nicht zu allgemein eingebracht werden sollen. Nutzen Sie ggf. auch einen Praxistransfer oder eine Praxisreflektion als Methode.

PRÜFUNGSFORM | KONTEXT & RAHMEN

INTERNATIONALE HOCHSCHULE

Kontext der Prüfung:

Demonstrieren Sie, wie Sie durch systematische Qualitätssicherung im Softwareprozess sicherstellen, dass Sie eine geeignete Software bereitstellen, die {*Themenstellung*} löst. Erklären und begründen Sie hierbei die angewandten Prüftechniken sowie den Aufbau Ihres Qualitätsmanagements.

Themenauswahl bis zum 23.04.2024.

Anmeldung: Angabe von (allen) Teilnehmern (mit Matrikelnummern und Standort) an knutlinke@iu.org

PRÜFUNGSFORM | KONTEXT & RAHMEN

Was beinhaltet die Bearbeitung?

- Artefakte erstellen (Programmieren) und präsentieren
- QS-Planung erstellen und präsentieren
- QS-Methoden vorstellen und anwenden

- Der Quellcode, die technische Verwendbarkeit des Artefaktes sowie das dortige Thema fließen ebenfalls in die Bewertung mit ein!
- Bei der Wahl des Themas und/oder Programmiersprache sind sie frei!

PRÜFUNGSFORM | ARTEFAKTE ERSTELLEN UND PRÄSENTIEREN

- Bei der Wahl ihres Referatsthemas wählen sie auch eine Softwareanforderung.

- Führen Sie im Rahmen ihrer Bearbeitung der Prüfungsleistung den Softwareherstellungsprozess aus, indem sie die Anforderung analysieren und beschreiben, eine geeignete Software entwickeln und die Qualitätssicherung anwenden.

- Stellen Sie die Artefakte vor, welche Sie bei der Ausführung des Softwareprozesses erzeugt haben (z. B. den Quellcode oder Ergebnisberichte der Qualitätssicherung).

PRÜFUNGSFORM | QS-PLANUNG ERSTELLEN UND PRÄSENTIEREN

- Bevor sie den Softwareprozess zur Realisation ihrer Anforderung ausführen, planen sie die Qualitätssicherung → erstellen Sie eine **sinnvolle** Qualitätsplanung

- Legen Sie Qualitätsziele fest (vgl. Softwareengineering).

- Stellen Sie das Ergebnis der Planung und der Anwendung im Rahmen des Vortrags vor.

PRÜFUNGSFORM | QS-METHODEN VORSTELLEN UND ANWENDEN

- **Je Team-Mitglied muss mindestens eine eigenständige Prüftechnik im Detail beschrieben und vorgestellt werden. Innerhalb der Gruppe dürfen Prüftechniken nicht doppelt vorgestellt werden.**

- **Die Prüftechniken "Review", "Inspektion" und "Walkthrough" zählen als eine Prüftechnik. Wenn alle durchgeführt werden, kann eine zweite Person Erkenntnisse aus einem Vergleich der drei Techniken vorstellen und diskutieren.**

- Die Vorstellung schließt eine kurze Demonstration der Prüftechnik ein, auf Basis ihres Softwareartefaktes (Was hat sich real verbessert? Was war der Vorteil der Prüftechnik? Warum wurde diese sinnvoll eingesetzt?)

PRÜFUNGSFORM | METHODEN FÜR DIE QUALITÄTSPLANUNG

- Die Qualitätsplanung beinhaltet z. B.
  - Ansätze aus dem Bereich der konstruktiven Qualitätssicherungsmaßnahmen (siehe Folie: Gesammelte Maßnahmen konstruktive Qualitätssicherung in Kapitel 3) und deren Anwendung (z. B. Checklisten, Richtlinien und Standards).
  - Ebenso sind statische Reviewtechniken (Datenflussanomalieanalyse, Walkthrough, Inspektion o. ä.) oder
  - dynamische Testverfahren (z. B. White- & Blackboxtesting, Zustandstabellen etc.) und systematisches Testen (Arbeiten mit Teststufen, Testdaten, Last- und Performancetests) möglich.
  - Es können auch eigene Prüftechniken eingebracht werden (siehe nächste Folie)

HINWEIS ZUR NUTZUNG VON WEITERFÜHRENDEN METHODEN ZUR QUALITÄTSSICHERUNG

- Innerhalb dieser Vorlesung werden nicht alle möglichen Testarten berücksichtigt.

- **Im Rahmen der Qualitätsplanung und der Qualitätssicherung im euren Projekt könnt ihr auch diese Testarten (z. B. Usability-Tests, Sicherheitstest, Akzeptanztests, Software-Qualitätssicherung: Development Pattern, Arbeiten mit Git/Git Arbeitsfluss etc.) einbringen.**

- Wichtig ist, dass ihr diese richtig anwendet und den theoretischen Rahmen miterklärt/vermittelt (Quellen etc. beachten).

PRÜFUNGSFORM | NEUE QS-PRÜFUNGSFORMEN

- **Was ist, wenn ich eine QS-Prüfungstechnik vorstellen und nutzen will, welches nicht in den Folien / Curriculum des Kurses abgedeckt ist und ich mir unsicher bin, ob die Methode geeignet ist?**

→ Kurz mit mir absprechen. Da der Kurs nicht alle Methoden abdecken kann, sind eigenständige, valide Vorschläge gerne gesehen, damit wir bei in der Gruppe auch in der Prüfung voneinander lernen können.

→ Wichtig ist, dass die Methode wissenschaftlich/validiert a) vorgestellt und b) angewendet wird

PRÜFUNGSFORM | HINWEIS

- Anforderungsbeschreibungen und ähnliche Artefakte, welche benötigt werden, um die gewählten Qualitätssicherungstechniken anzuwenden, sind diese im Rahmen der Prüfungsleistung in einer geeigneten und bewertbarer Qualität herzustellen.

- Treffen Sie geeignete Annahmen, falls nötig. Sie können den Kunden (sinnvoll) definieren/annehmen.

- Ihr Ziel ist es, die Leistungsfähigkeit der Prüftechniken, in Bezug auf etwaige Artefakten und der Verbesserung des Quellcodes und der Programmierung deutlich zu machen.

PRÜFUNGSFORM | HINWEIS

- Was ist wichtig beim Referat?
  - Prüfen auf Qualität
  - Mängel aufdecken (wie hat die Methode geholfen?)
  - Wie konntet ihr den Code eleganter gestalten?

- Was wird auch erwartet:
  - Abbildung des Anforderungs-/Projektmanagement
  - Auch kurzer Eingang auf Probleme: Z. B. Definition von Anforderungen und deren Klärung.

PRÜFUNGSFORM | AUFBAU HANDOUT

- Einleitung
- Vorstellung der zu entwickelnden Software
- Qualitätsplan
- Prüftechniken (1 – n)
- Reflexion
- Fazit

PRÜFUNGSFORM | WIE KANN DIE PRÄSENTATION AUFGEBAUT SEIN

- Einleitung/Übersicht
- Anforderungen (Welche gab es? Wie waren die Qualitätsansprüche?)
- Qualitätsplanung (Wie sieht diese aus? Wie wird diese realisiert? Welche Rollen?)
- Vorstellung und Anwendung der Prüftechniken 1-n (Verantwortung der Testmethode/Umsetzung und Outcome. Wie sieht die Software nach der QS aus?)
- Reflexion (Was lief gut/Was lief schlecht/Was hättet ihr anders gemacht in Bezug auf QS)?
- Fazit

PRÜFUNGSFORM | ARTEFAKTE

- Als Software Engineer / Softwareentwickler bist du/seid ihr für die QS und das Softwareprojekt zuständig: „Die Tätigkeiten Analysieren, Entwerfen, Kodieren und Prüfen sind Bestandteil aller Prozessmodelle“ in der Softwareentwicklung.

- Im Referat sollten die für das Softwareprojekt notwendigen Artefakte passend vorkommen (Technische Dokumentation, Use Cases etc.), damit das Programm schnell verstanden werden kann. Der Kern des Referates liegt auf der QS, deren Planung und Durchführung sowie des Ergebnisses im Kontext des Quellcodes der SW-Lösung.

- Die Reflexion des Referates kann, neben einer Reflexion der eingesetzten Methoden, auch die Reflexion des Projekt- und vor allem des Qualitätsplans beinhalten.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 389

PRÜFUNGSFORM | THEMENPOOL

Nr.	Themenstellung
Thema 1	Generierung von Passworten
Thema 2	Organisation von To-Do-Listen
Thema 3	Generierung von Produktvorschlägen auf Basis bisheriger Käufe
Thema 4	Verwaltung von Lernkarten als APP
Thema 5	Zugriff auf ChatGPT per API
Thema 6	Aufbau eine OS-KI-Systems
Thema 7	Generierung von Zufallszahlen, ggf. per Quantencomputer
Thema n	**Ein passendes Thema (Softwareprojekt/Teilprojekt)**

PRÜFUNGSFORM | KONTEXT & RAHMEN

Rahmen der Prüfung:

Art: Referat – 15 Minuten pro Teilnehmer (Handout, Präsentation und Quellcode)

Gibt es Gruppen? Ja, es gilt die Regel: 15 Minuten pro Teilnehmer

Was bedeutet das?

Anzahl Teilnehmer	Dauer Präsentation	Umfang Handout
2	30 Minuten	6 Seiten
3	45 Minuten	9 Seiten

Gruppengrößen: 2-3 Personen

Prüfungsdatum: Aufgrund der Größe der Studiengruppe gibt es 2 Prüfungsgruppen

HINWEISE ZUR PRÜFUNG | ANMELDUNG

Die Anmeldung beinhaltet:

Gruppenprüfung
Thema
Alle Namen (Vor- und Zunamen) der Personen die zu dem angemeldeten Team gehören
Matrikelnummern
Standorte

Die Anmeldung ist bis zum 23.04.2024 durchzuführen.

Was passiert, wenn keine Anmeldung bis zum genannten Datum passiert? Dann meldet der Professor die Teilnehmer als Gruppen mit einem Thema seiner Wahl an.

PRÜFUNGSFORM | THEMENPOOL

Zugelassene Programmiersprachen: PHP, Java, TypeScript, JavaScript, C/C++, .net/C#, VB, Python oder vergleichbare **(Auswahl begründen)**

Zugelassene Hilfsmittel: Verwenden von vorgefertigtem Code als Lösung führt zur Disqualifikation. Code der zur Inspiration genutzt wird, ist als Quelle anzugeben. Refactoring ist durch den Studierenden zu realisieren. Das Referactoring kann, dokumentiert, durch KI begutachten werden.

Quellcode, Präsentation und Handout sind zur mündlichen Prüfung an: knut.linke@iu.org zu schicken (.exe bitte separat schicken wg. der IU-Firewall). Die finale Präsentation (falls es Änderungen gab) ist nach der Präsentation nachzusenden.

HINWEISE ZUR PRÜFUNG | KONTEXT

INTERNATIONALE HOCHSCHULE

Hilfe, die Anforderungen sind unvollständig:

- So ist das Arbeitsleben… auch Anforderungen in IT-Projekten sind häufig unvollständig… (aber das schafft ihr)
- Unklarheiten und Unsicherheiten gehören zur Welt & wenn es klar wäre, kann es die KI

Das heißt:

- Es ist Teil der Aufgabe Lücken/Unklarheiten zu identifizieren und Lösungen herbeizuführen und bedeutet, dass
- Dokumentieren und Begründen von Annahmen (im Kontext des Gelernten und des Moduls)

HINWEISE ZUR PRÜFUNG | KONTEXT

Hinweis zur Benotung:

- Die Themen geben viele Freiheiten, innovative Lösungen zu entwickeln und Ihre Fähigkeiten im Bereich der Qualitätssicherung im Softwareentwicklungsprozess unter Beweis zu stellen.

- Diese Fähigkeiten sollen gezeigt werden, wenn eine überdurchschnittliche Benotung anvisiert wird (bitte kein Standard – zeigen Sie was Sie können!).

ZEITPLAN

INTERNATIONALE HOCHSCHULE

39 Studierende: BER, HH, HAN, LE (2-3 Gruppen)

Datum	Zeit	Kontext	Umfang
11.04.2024	9:00-13:15	Einführung	5 UE
18.04.2024	9:00-13:15	Vorlesung	5 UE
25.04.2024	9:00-13:15	Vorlesung	5 UE
16.05.2024	9:00-13:15	Vorlesung	5 UE
23.05.2024	9:00-13:15	Vorlesung	5 UE
30.05.2024	9:00-13:15	Vorlesung	5 UE
07.06.2024	14:00-18:15	Vorlesung	5 UE
21.06.2024	10:45-15:30	Vorlesung	5 UE
27.06.2024	9:00-13:15	Referate	5 UE
04.07.2024	9:00-13:15	Referate	5 UE
18.07.2024	9:00-13:15	Referate	5 UE
25.07.2024	9:00-13:15	Referate	5 UE

Anwesenheit an 2 der Referatstage ist Pflicht

Einbringen in die Referate ist erwünscht.

ZEITPLAN

39 Studierende: BER, HH, HAN, LE (2-3 Gruppen)

Datum	Zeit	Kontext	Umfang
11.04.2024	9:00-13:15	Einführung	5 UE
18.04.2024	9:00-13:15	Vorlesung	5 UE
25.04.2024	9:00-13:15	Vorlesung	5 UE
16.05.2024	9:00-13:15	Vorlesung	5 UE
23.05.2024	9:00-13:15	Vorlesung	5 UE
30.05.2024	9:00-13:15	Vorlesung	5 UE
07.06.2024	14:00-18:15	Vorlesung	5 UE
21.06.2024	10:45-15:30	Vorlesung	5 UE
27.06.2024	9:00-13:15	Referate	5 UE
04.07.2024	9:00-13:15	Referate	5 UE
18.07.2024	9:00-13:15	Referate	5 UE
25.07.2024	9:00-13:15	Referate	5 UE

Vorschläge an welchem Datum können berücksichtigt werden.
Zuordnung geschieht nach Eingang.
Bitte Erst-, Zweit- und Drittwunsch mitteilen.

AGENDA

Einführung in die Softwarequalitätssicherung	1
Organisation und Planung von Softwarequalität	2
Konstruktives Qualitätsmanagement	3
Statische Qualitätssicherung: Begutachtung und Messen	4
Dynamische Qualitätssicherung: Testen	5
Systematisches Testen von Software	6
Systematische Qualitätssicherung von Anforderungen, Architekturen und Prozessen	7

01

iu INTERNATIONALE HOCHSCHULE

EINFÜHRUNG IN DIE SOFTWAREQUALITÄTSSICHERUNG

GRUPPENIMPULS

Kleingruppendiskurs:

Welche Methoden zur Qualitätssicherung in der Softwareentwicklung setzt ihr ein?

Berücksichtigt auch das Ziel des QS

- Ergebnisse werden von ausgewählten (freiwilligen) Gruppen im Plenum vorgestellt (PowerPoint o. ä.)
- 20 Minuten Zeit

ERGEBNISSE

LERNZIELE

- Bedeutung der wichtigsten Begriffe aus dem Themenbereich Softwarequalitätssicherung und Qualitätssicherung

- Prinzipien & Grundsätze der Softwarequalitätssicherung und deren Auswirkungen (Orientierung bei den Qualitätssicherungsverfahren)

- Wirtschaftlichkeit fehlerfreier Informationssysteme

MOTIVATION UND BEGRIFFE

- „**Software**,
  - ihre Erstellung sowie ihre Wartung ist eine Schlüsselkompetenz des 21. Jahrhunderts in vielen Unternehmen zur Bewältigung der Geschäftsherausforderungen
  - im Hinblick auf Flexibilität, Time-to-Market und
  - die Fähigkeit, Innovationen kontinuierlich hervorzubringen,
  - bei niedrigen Kosten und
  - gleichzeitiger Wahrung einer exzellenten Qualität.“

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag.

MOTIVATION UND BEGRIFFE

- „**Softwarequalität** ist die Gesamtheit der Merkmale und Merkmalswerte eines Softwareprodukts, die sich auf dessen Eignung beziehen, **festgelegte** Erfordernisse zu erfüllen.“ (ISO 9126)

- **Qualitätsmanagement** alle organisierten Maßnahmen zusammengefasst, die der Verbesserung der Qualität von Produkten, Prozessen oder Leistungen jeglicher Art dienen.

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag.

WARUM IST QUALITÄTSSICHERUNG BEI DER SOFTWAREENTWICKLUNG NOTWENDIG UND SINNVOLL?

Eine Anforderung: Als IT-Service-Provider erbringen Sie Leistungen für Kunden zu unterschiedlichen Preisen, gestaffelt nach vertraglich vereinbarten Service-Niveaus. Die Abrechnung erfolgt auf Stundenbasis:

- Kunden, die den Basic-Service gebucht haben, werden 30 € pro Stunde berechnet.
- Kunden, die den Medium-Service gebucht haben, werden 40 € pro Stunde berechnet.
- Kunden, die den Premium-Service gebucht haben, werden 50 € pro Stunde berechnet.

Es soll ein Programm entwickelt werden, welches für eine angegebene Service-Kategorie und eine Anzahl an Stunden den abzurechnenden Betrag berechnet.

WARUM IST QUALITÄTSSICHERUNG BEI DER SOFTWAREENTWICKLUNG NOTWENDIG UND SINNVOLL?

```
public static void main(String[] args) {
  double anzStunden, nettoBetrag, stundensatz, serviceKategorie;
  Scanner input = new Scanner(System.in);
  System.out.println("Angabe Servicekategorie (Werte 1 , 2 oder 3): ");
  serviceKategorie = input.nextInt();
  System.out.println("Geben Sie die Anzahl an Stunden an: ");
  anzStunden = input.nextDouble();
  if(serviceKategorie < 1) {
    stundensatz = 30;
  }
  if(serviceKategorie >= 2) {
    stundensatz = 40;
  }
  else {
    stundensatz = 50;
  }
  System.out.println("Endbetrag: "+stundensatz * anzStunden);
}
```

Führen Sie eine Qualitätssicherung für das Programm durch. Identifizieren Sie, getrennt nach den Begriffen:

- Fehler
- Fehlverhalten
- Fehlerursachen

PRINZIPIEN DER SOFTWARE-QUALITÄTSSICHERUNG

- **Fehler**: „Ein Fehler oder Defekt (*fault, defect*) ist bei Software die statisch im Programmcode vorhandene Ursache eines Fehlverhaltens oder Ausfalls.“
- **Fehlverhalten**: „Ein Fehlverhalten oder Ausfall (*failure*) zeigt sich dynamisch bei der Benutzung eines Produkts. Beim dynamischen Test einer Software erkennt man keine Fehler sondern Fehlverhalten bzw. Ausfälle. Diese sind Wirkungen von Fehlern im Programm.“
- **Fehlerursache:** Der Grund, welcher den Fehler hervorruft.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 6-7.

ERGEBNISSE

WAS SAGT EIGENTLICH CHAT GPT?

In diesem Code gibt es einige Probleme. Zunächst wird die Variable `nettoBetrag` deklariert, aber nie verwendet. Außerdem gibt es ein Problem mit der Logik der if-else-Anweisungen. Wenn `serviceKategorie` kleiner als 1 ist, wird `stundensatz` auf 30 gesetzt, aber dann wird es sofort auf 50 gesetzt, da die zweite if-Anweisung nicht erfüllt ist. Dies bedeutet, dass `stundensatz` immer 50 ist, wenn `serviceKategorie` kleiner als 1 ist.

Eine Möglichkeit, dieses Problem zu lösen, besteht darin, die if-else-Anweisungen wie folgt zu ändern:

```
if(serviceKategorie < 1) {
   stundensatz = 30;
}
else if(serviceKategorie == 2) {
   stundensatz = 40;
}
else {
   stundensatz = 50;
}
```

REMINDER: DER SOFTWAREHERSTELLUNGSPROZESS

- „Ein definierter Software-Entwicklungsprozess ist eine notwendige Voraussetzung für eine reife, qualitätsorientierte Software Entwicklung.“

- „Die Tätigkeiten Analysieren, Entwerfen, Kodieren und Prüfen sind Bestandteil aller Prozessmodelle. Entwickelnde und prüfende Tätigkeiten beeinflussen einander, da die Ergebnisse der Entwicklungsphasen als Prüfreferenzen dienen.“

- „Eine sinnvolle Prüfung ist in Phasen unterteilt.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 389

QUALITÄTSMERKMALE

- „Qualitätsmerkmale stellen Eigenschaften einer Funktionseinheit dar, anhand derer ihre Qualität beschrieben und beurteilt wird, die jedoch keine Aussage über den Grad der Ausprägung enthalten.“

- „Zwischen Qualitätseigenschaften können Wechselwirkungen und Zusammenhänge existieren.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 5-6.

AKTIVITÄTEN IM QUALITÄTSMANAGEMENT IM KONTEXT VON SOFTWAREENTWICKLUNG

- Qualitätsplanung
- Qualitätslenkung
- Qualitätssicherung (auch QS)
- Qualitätsverbesserung

AKTIVITÄTEN IM QUALITÄTSMANAGEMENT IM KONTEXT VON SOFTWAREENTWICKLUNG

- **Qualitätsplanung:** Erstellung und Dokumentation der Qualitätsanforderungen gemeinsam mit dem Auftraggeber.

- **Qualitätslenkung:** Überwachung, Steuerung und Kontrolle von Aktivitäten zur Qualitätsprüfung im Softwareentwicklungsprozess.

AKTIVITÄTEN IM QUALITÄTSMANAGEMENT IM KONTEXT VON SOFTWAREENTWICKLUNG

- **Qualitätssicherung (auch QS):** Tätigkeiten, die sicherstellen, dass festgelegte Qualitätsanforderungen für Produkte, Prozesse und Leistungen erfüllt werden.

- **Qualitätsverbesserung:** Erhebungen mit dem Ziel Prozesse zu verbessern

PRINZIPIEN DER SOFTWARE-QUALITÄTSSICHERUNG

- Softwarequalitätssicherung besteht nicht nur aus ‚Testen‘ (analytisch, dynamisch), sondern auch konstruktiven und analytisch-statischen Maßnahmen

- Gute Qualitätssicherung folgt den Prinzipien und achtet bei der Umsetzung auf die grundsätzlichen Eigenschaften von Softwarequalität.

- (Immer) auf die Kosten achten!

PRINZIPIEN DER SOFTWARE-QUALITÄTSSICHERUNG

- **Qualitätsmaß**: „Die konkrete Feststellung der Ausprägung eines Qualitätsmerkmales geschieht durch so genannte Qualitätsmaße. Dies sind Maße, die Rückschlüsse auf die Ausprägung bestimmter Qualitätsmerkmale gestatten.“

- „**Qualitätsmanagement** befasst sich insbesondere mit organisatorischen Maßnahmen zur Erreichung und zum Nachweis einer hohen Produktqualität.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 7, 10.

PRINZIPIEN DER SOFTWARE-QUALITÄTSSICHERUNG

- „Software-**Qualitätssicherung** stellt Techniken zur Erreichung der gewünschten Ausprägungsgrade der Qualitätsmerkmale von Software-Systemen zur Verfügung.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 30.

KONSTRUKTIVE UND ANALYTISCHE QUALITÄTSSICHERUNG

- **Konstruktiv:**
  - Fehlervermeidend
  - Vor der Erstellung eines Softwareartefaktes

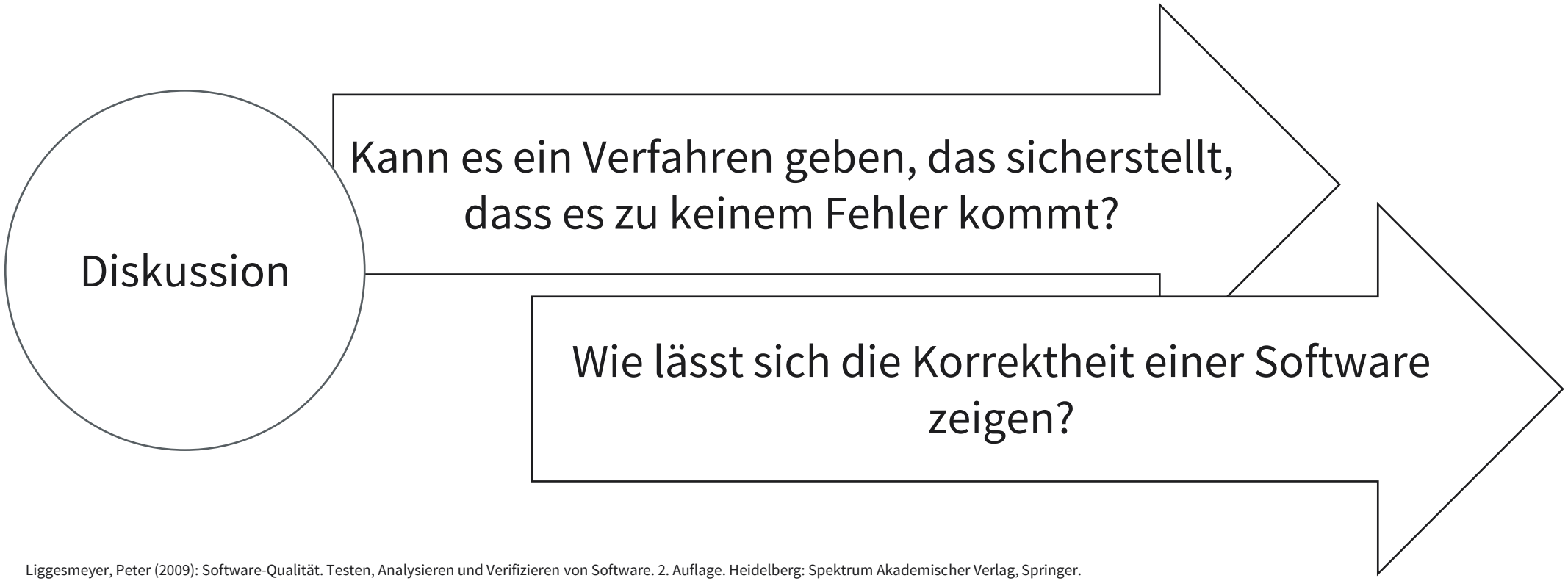
- **Analytisch:**
  - Bewertend
  - Zumeist nach der Erstellung eines Softwareartefaktes
  - Kann ein dynamisches oder statisches Vorgehen sein

ANALYTISCHE QUALITÄTSSICHERUNG

	Statische Verfahren	Dynamische Verfahren
Methode	Nur Begutachtung	Ausführung
Prüfling	Artefakte und Prozesse (auch Quellcode, Beschreibungen, Diagramme)	Ausführbare Programme (auch Komponenten)
Verfahren	Review, Metriken, statische Codeanalyse	Funktionsorientierte Tests, Zufallstest
Verantwortung	Qualitätssicherung oder Qualitätslenkung	Qualitätssicherung

NOTWENDIGKEIT DER QUALITÄTSSICHERUNG?

„Die Aufwendungen für Fehlerkorrekturen werden umso höher, je größer die Distanz zwischen der Fehlerentstehung und Fehlerfindung ist. Daher ist es sinnvoll, die Dokumente der frühen Entwicklungsphasen einer Prüfung zu unterziehen.“



Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer.

NOTWENDIGKEIT DER QUALITÄTSSICHERUNG - IMPULSE

- „Kein konstruktives Verfahren kann garantieren, dass fehlerfreie Produkte entstehen. Aus diesem Grund ist die Qualität mit analytischen Mitteln zu prüfen.“

- „Es ist erkennbar, dass einzelne Techniken keine umfassende Lösung der existierenden Probleme bieten können.
  Eine gleichzeitige Befriedigung technischer und wirtschaftlicher Ziele in einem industriellen Kontext ist durch eine sinnvolle Kombination mehrerer Techniken und erreichbar.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 9; 32.

ZIELSETZUNG DER QUALITÄTSSICHERUNG

- „Das Ziel ist möglichst viele Fehler bereits in der Prüfung zu erkennen, die auf den Konstruktionsschritt folgt, in dem der Fehler entstanden ist.“

- „Das Ziel der Prüfung muss die Erzeugung reproduzierbarer Ergebnisse unter Zugrundelegung einer klar definierten Vorgehensweise sein.“

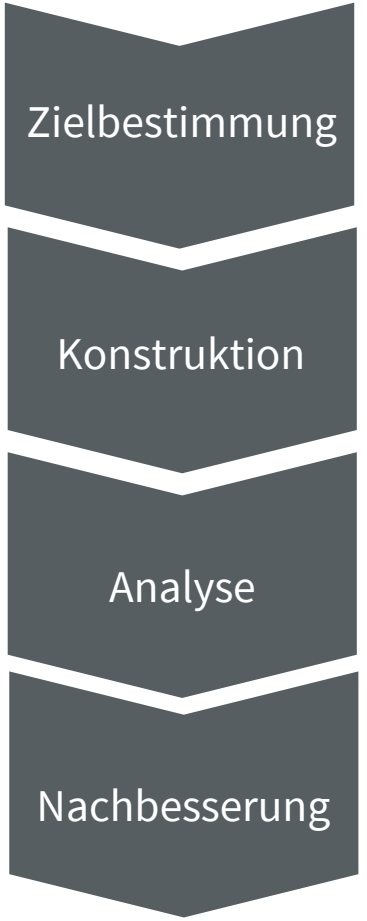
Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 3-4.

ORGANISATION DER QUALITÄTSSICHERUNG

- „Die Qualitätszielbestimmung geschieht durch Festlegung der gewünschten Ausprägungen so genannter Qualitätsmerkmale zu einem frühen Zeitpunkt der Entwicklung.“

- „Im Idealfall sollte einer konstruktiven Tätigkeit eine entsprechende analytische folgen.“

- „Der Begriff der Qualität ist in der DIN EN ISO 9000 /DIN EN ISO 9000 definiert als der Grad, in dem ein Satz inhärenter Merkmale Anforderungen erfüllt.“



Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 3-5.

GRUNDSÄTZE IM SOFTWARETEST

Eure Meinung (Gruppendiskurs) zu folgenden Grundsätzen:

1. Testen zeigt die Anwesenheit von Fehlern
2. Vollständiges Testen ist nicht möglich
3. Wiederholungen haben keine Wirksamkeit
4. Testen ist abhängig vom Umfeld (Projektumfeld)
5. Trugschluss: Keine Fehler bedingen nicht zwangsläufig ein brauchbares System

DAS TEAM IM SOFTWARETEST

Teamarbeit – 20. Minuten

Plant Maßnahmen zur Sicherung der Qualität im Anforderungsmanagement.

Stimmt hierzu im einen Prozess für eure Referat ab, der

- Anforderungen in hoher Qualität dokumentiert, um die Entwicklung vorzubereiten
- Praktikabel in seiner Anwendung ist
- Nicht zu hohe Qualitätssicherungskosten auslöst

Passende Präsentation der Ergebnisse von ausgewählten Studiengruppen

ERGEBNISSE

GRUNDSÄTZE DER QUALITÄTSSICHERUNG

Folgende Prinzipien zur Beeinflussung der Qualität haben sich bewährt:

- P1: Konkrete operationalisierbare Qualitätsmerkmale
- P2: Produkt- und projektabhängige Qualitätsplanung
- P3: Rückkopplung der Ergebnisse der Qualitätsprüfungen
- P4: Mehr-Augenkontrolle bei Qualitätsprüfungen
- P5: Maximaler Einsatz konstruktiver Engineering-Praktiken und –Maßnahmen
- P6: Frühzeitige Entdeckung und Behebung von Fehlern und Mängeln
- P7: Integriertes Qualitätsmanagement
- P8: Unabhängigkeit bei Qualitätsprüfungen
- P9: Evaluation der eingesetzten Qualitätsmaßnahmen“

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 35.

GRUNDSÄTZE DER QUALITÄTSSICHERUNG

P1: Konkrete operationalisierbare Qualitätsmerkmale

Bedeutet:

- Festlegung von Qualitätsmerkmalen
- Anforderungsabstimmung mit dem Auftraggeber

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 35-36.

GRUNDSÄTZE DER QUALITÄTSSICHERUNG

P2: Produkt- und projektabhängige Qualitätsplanung

Bedeutet:

- Verwendungszweck der Software…
- Lebensdauer des Produktes…
- Anz. Benutzer*innen…

…beeinflussen **Qualitätsfaktoren**

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 35-36.

GRUNDSÄTZE DER QUALITÄTSSICHERUNG

P3: Rückkopplung der Ergebnisse der Qualitätsprüfungen

P4: Mehr-Augenkontrolle bei Qualitätsprüfungen

Bedeutet:

- Korrekturmaßnahmen im Entwicklungsprozess
- Gruppenprozesse

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 35-37.

GRUNDSÄTZE DER QUALITÄTSSICHERUNG

P5: Maximaler Einsatz konstruktiver Engineering-Praktiken und –Maßnahmen

Bedeutet:

- Qualitätsprüfung: Identifikation der vorhandenen Qualität
- Ziel konstruktiver Praktiken: Vermeidung von Fehlern im Entwicklungsprozess
- vorbeugende Maßnahmen

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 35, 37.

GRUNDSÄTZE DER QUALITÄTSSICHERUNG

P6: Frühzeitige Entdeckung und Behebung von Fehlern und Mängeln

Bedeutet:

- Wenn Fehler und Mängel frühzeitig entdeckt werden, sorgen diese für weniger Schaden und geringere Kosten.

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 35.

GRUNDSÄTZE DER QUALITÄTSSICHERUNG

P7: Integriertes Qualitätsmanagement

Bedeutet:

- „Die Grundidee dabei ist, dass jede Entwicklungsaktivität aus einem konstruktiven und einem analysierenden/prüfenden Teil besteht.“

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 35, 38.

GRUNDSÄTZE DER QUALITÄTSSICHERUNG

P8: Unabhängigkeit bei Qualitätsprüfungen

Bedeutet:

- Interessenkonflikte, wenn Entwickler die eigene Arbeit prüfen
- Betriebsblindheit und mangelnder Objektivität vorbeugen

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 35, 38.

GRUNDSÄTZE DER QUALITÄTSSICHERUNG

P9: Evaluation der eingesetzten Qualitätsmaßnahmen

Bedeutet:

- Korrekturen des QS-Prozesse
- KAIZEN (kontinuierliche Verbesserung)

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 35, 39.

GRUNDSÄTZE DER QUALITÄTSSICHERUNG

Folgende Prinzipien zur Beeinflussung der Qualität haben sich bewährt:

- P1: Konkrete operationalisierbare Qualitätsmerkmale
- P2: Produkt- und projektabhängige Qualitätsplanung
- P3: Rückkopplung der Ergebnisse der Qualitätsprüfungen
- P4: Mehr-Augenkontrolle bei Qualitätsprüfungen
- P5: Maximaler Einsatz konstruktiver Engineering-Praktiken und –Maßnahmen
- P6: Frühzeitige Entdeckung und Behebung von Fehlern und Mängeln
- P7: Integriertes Qualitätsmanagement
- P8: Unabhängigkeit bei Qualitätsprüfungen
- P9: Evaluation der eingesetzten Qualitätsmaßnahmen“

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 35.

DAS TEAM IM SOFTWARETEST

INTERNATIONALE HOCHSCHULE

Teamarbeit – Reflektion (25 Min.):

Was würdet ihr an eurem Vorgehen in der Gruppe mit den neuen Erkenntnissen (Grundsätze der Qualitätssicherung) ändern?

Plant Maßnahmen zur Sicherung der Qualität im Anforderungsmanagement.
Stimmt hierzu im einen Prozess für eure Referat ab, der

- Anforderungen in hoher Qualität dokumentiert, um die Entwicklung vorzubereiten
- Praktikabel in seiner Anwendung ist
- Nicht zu hohe Qualitätssicherungskosten auslöst

ERGEBNISSE:

KOSTEN VON QUALITÄT – DER ÖKONOMISCHE ANSPRUCH

Solang wir über Software sprechen, **die in nicht lebenskritischen Systemen eingesetzt wird**, gilt:

- „Das Ziel ist nicht die Entwicklung einer Software mit der besten, sondern mit der richtigen Qualität. **Dies erfordert eine Festlegung der gewünschten Qualität durch eine so genannte Qualitätsziel.** Anschließend können Maßnahmen zur Erreichung der Qualität beschlossen werden.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 1-2.

KOSTEN VON QUALITÄT – DER ÖKONOMISCHE ANSPRUCH

Solang wir über Software sprechen, **die in nicht lebenskritischen Systemen eingesetzt wird**, gilt:

- „Eine Kostenreduktion im Bereich der Software-Entwicklung ist demzufolge besonders ökonomisch. Die Analyse der Kosten von Software-Entwicklungen in Abhängigkeit der Phasen des Softwarelebenszyklus führt zu dem Ergebnis, dass der weitaus größte Anteil der Anwendungen während der Wartungsphase eines im Markt befindlichen Produkts entsteht. Dies ist eine Folge der unzureichenden Qualität der Software.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 2.

WIRTSCHAFTLICHKEIT

- „Die Software Erstellung unterliegt wie jeder andere industrielle Produktions- und Entwicklungsprozess dem Zwang zu hoher Produktivität. Dies wiederum bedingt eine Berücksichtigung der Ziele Termintreue, minimale Kosten und zufriedenstellende Produktqualität. Wir sprechen dann von einer wirtschaftlichen Software-Erstellung, wenn diese Ziele verwirklicht worden sind.“

- „Beim Qualitätsmanagement sind immer auch Wirtschaftlichkeitsgesichtspunkte zu beachten.“

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 35, 43.

QUALITÄTSPLANUNG UND QUALITÄTSZIELE

- Qualitätsmanagement muss geplant und über Ziele gesteuert werden!
- Qualitätsmanagement besteht aus: Planung (Lenkung), Sicherung und Verbesserungen!
- TQM kann als Qualitätslenkung eingesetzt werden!

- **TQM bezieht explizit alle Mitarbeiter einer Organisation bei der Qualitätslenkung ein Eine hohen Produktqualität ist beim TQM das höchste Ziel aller Mitarbeiter.**

WIRTSCHAFTLICHKEIT

- „Das Qualitätswesen (eine Organisation für das QM) erbringt immer Dienstleistungen für die Produkt-, Projekt- und Prozessverantwortlichen und sollte für die Bewertung bzw. Messung der Qualität verantwortlich sein.“

- „Das Qualitätswesen muss einen unabhängigen Berichtsweg zur Leitung der Organisation bzw. zur Geschäftsleitung haben.“

- „Die Angestellten müssen über die Qualität ihrer Arbeit Bescheid wissen bzw. informiert werden.“

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 43.

DIE WIRTSCHAFTLICHKEIT

Wir kann sich der Kontext der Wirtschaftlichkeit auf die Entwicklung von Software auswirken?

- Welches Praxisbespiele fallen euch ein?
- Wann kann eine Reduktion von Qualität in der Software sinnvoll sein?
  - Wo könnt ihr die Reduktion von Qualität aus der Sicht eines Software Engineers vertreten? In welchen Bereiche fällt das schwerer?

- Wie gilt der Kontext der Wirtschaftlichkeit für euer Referat?

20 Min.

ERGEBNISSE WIRTSCHAFTLICHKEIT

02

INTERNATIONALE HOCHSCHULE

ORGANISATION UND PLANUNG VON SOFTWAREQUALITÄT

ÜBERBLICK ÜBER DEN QUALITÄTSMANAGEMENTPROZESS

Was ist Qualitätsmanagement?

- Aufeinander abgestimmte Tätigkeiten zum Leiten und Lenken einer Organisation bezüglich Qualität,
- die üblicherweise das Festlegen der Qualitätspolitik und der Qualitätsziele, die Qualitätsplanung, die Qualitätslenkung, die Qualitätssicherung und die Qualitätsverbesserung umfasst.

GRUNDSÄTZ DES QUALITÄTSMANAGEMENTS

Wallmüller formuliert folgende Grundsätze für das Qualitätsmanagement

- „Qualität muss erzeugt werden und kann nicht in ein Produkt hineingeprüft werden.“
- „[Q]ualität [sic] bezieht sich immer auf das Produkt und auf die damit verbundenen Prozesse (Entwicklung, Pflege, etc.).“
- „Die Qualitätsverantwortung ist immer auch mit der Sach-, Termin- und Kostenverantwortung verbunden und muss balanciert gemanagt werden.“

Wallmüller, Ernest (2011): Software Quality Engineering. Ein Leitfaden für bessere Software-Qualität. 3. Auflage. München: Carl Hanser Verlag. S. 43.

ÜBERBLICK ÜBER DEN QUALITÄTSMANAGEMENTPROZESS

Was ist Qualitätsplanung?

- Planung (Vorgaben zur Organisation und Durchführung) aller Maßnahmen zur Qualitätssicherung
- **Bestimmung der Qualitätsziele** und der **organisatorischen Aspekte** der Qualitätssicherung
- Legt für einen konkreten Softwareprozess fest, welche Personen für welche QS-Aktivitäten zuständig sind risikobasierter Ansatz
- Wann? frühe Projektphase
- **Qualitätsplan** → Festlegung der Qualitätsmerkmale (z. B. für eine Software)

DEFINITION DER QUALITÄTSZIELBESTIMMUNG

- „**Qualitätsmerkmal**, Eigenschaft einer Funktionseinheit, anhand derer ihre Qualität beschrieben und beurteilt wird, die jedoch keine aussage über den Grad der Ausprägung enthält“

- „**Qualitätszielbestimmung,** Definition der für ein Produkt zu erreichenden Qualität durch Festlegung der angestrebten Ausprägungen der Qualitätseigenschaften. Die Qualitätszielbestimmung wird zu Beginn der Entwicklung durchgeführt.“

→ Festlegung erfolgt z. B. im Lasten- / Pflichtenheft

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 515.

ISO 9126 REMINDER

Funktionalität
- Angemessenheit
- Genauigkeit
- Interoperabilität
- Sicherheit
- Konformität

Zuverlässigkeit
- Reife
- Fehlertoleranz
- Wiederherstellbarkeit
- Konformität

Benutzbarkeit
- Verständlichkeit
- Erlernbarkeit
- Bedienbarkeit
- Attraktivität
- Konformität

Effizienz
- Zeitverhalten
- Verbrauchsverhalten
- Konformität

Wartbarkeit
- Analysierbarkeit
- Änderbarkeit
- Stabilität
- Testbarkeit
- Konformität

Portabilität
- Anpassbarkeit
- Installierbarkeit
- Koexistenz
- Austauschbarkeit

QUALITÄTSLENKUNG UND QUALITÄTSPLANUNG: CONTROLLING

INTERNATIONALE HOCHSCHULE

Teamarbeit: Der Qualitätsplan

Kontext: Sie arbeiten für ein Ingenieurbüro, welches den Bau von Schwimmbädern, Wasserreservoirs, Abwasserleitungen, Tankanlagen etc. realisiert und kalkuliert.

Zur Arbeitserleichterung soll ein Programm von Ihrer Softwarefirma entwickelt werden, welche für verschiedene Wasserbehälter und deren Formen Volumen und mögliche Wassermenge berechnet.

Aufgabe: Entwickeln Sie einen Qualitätsplan, der sich an der ISO 9126 orientiert und präsentieren Sie ihr Ergebnis im Rahmen einer Unternehmenspräsentation für Ihren Kunden. (30 Minuten)

ERGEBNISSE QUALITÄTSPLAN

DER SOFTWAREHERSTELLUNGSPROZESS

- „Ein definierter Software-Entwicklungsprozess ist eine notwendige Voraussetzung für eine reife, qualitätsorientierte Software Entwicklung.“

- „Die Tätigkeiten Analysieren, Entwerfen, Kodieren und Prüfen sind Bestandteil aller Prozessmodelle. Entwickelnde und prüfende Tätigkeiten beeinflussen einander, da die Ergebnisse der Entwicklungsphasen als Prüfreferenzen dienen.“

- „Eine sinnvolle Prüfung ist in Phasen unterteilt.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 389

TESTMANAGEMENT GRUNDLAGEN

- Um ein Fehlverhalten von Software zu erkennen, muss der Fehlerort durch einen Testfall ausgeführt werden.
  - Bei der Ausführung des Fehlerortes muss die Fehlersituation eintreten.
  - Dies ist nicht zwangsläufig bei jeder Ausführung des Fehlerortes erfüllt, da der Eintritt der Fehlersituation möglicherweise an die Ausführung mit bestimmten Datenwerten gekoppelt ist.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 211

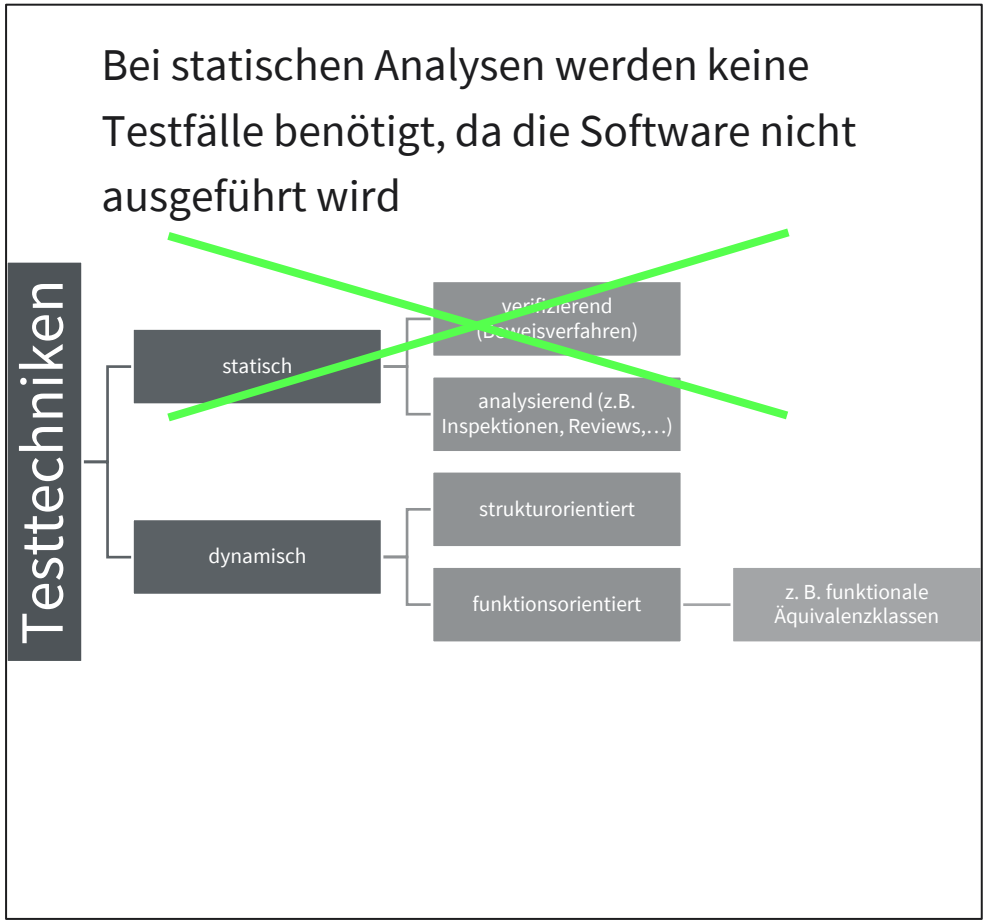
TESTMANAGEMENT GRUNDLAGEN

- Auch der Eintritt der Fehlersituation garantiert nicht, dass ein von außen wahrnehmbares Fehlverhalten eintritt.

- Daher ist es erforderlich, dass zur Erkennung des Fehlverhaltens eine Fortpflanzung der Fehlersituation an eine von außen wahrnehmbare Stelle erfolgt.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 211

- Testtechniken können in Basis-Testtechniken und höhere Testtechniken aufgeteilt werden.
  - Basistesttechniken können nicht sinnvoll in eigenständige untergeordnete Testtechniken zerlegt werden.
  - Höhere Testtechniken bestehen aus mehreren Basistesttechniken



Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 40, Bild S. 43

TESTMANAGEMENT GRUNDLAGEN

- Dynamische Testtechniken definieren unterschiedliche Regeln für die Bildung und Beurteilung der Testfälle.

- Testtechniken können nach unterschiedlichen Kriterien klassifiziert werden.
  - Weit verbreitet ist die Unterscheidung von White Box-Tests und Black Box-Tests.
  - White Box-Techniken nutzen im Gegensatz zu Black Box-Techniken die Struktur des Programmcodes.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 40

INTERNATIONALE HOCHSCHULE

Teamarbeit: Der Qualitätsplan

Diskurs:

Wie lassen sich die Ergebnisse auf eure Referate übertragen?

ZWISCHENSTAND REFERATE

- Welche Methoden habt ihr euch schon überlegt?

03

INTERNATIONALE HOCHSCHULE

KONSTRUKTIVES QUALITÄTSMANAGEMENT

LERNZIELE

- Aktivitäten und Maßnahmen konstruktive Qualitätssicherung
- 5-Why-Methode
- Timeboxing und Checklisten

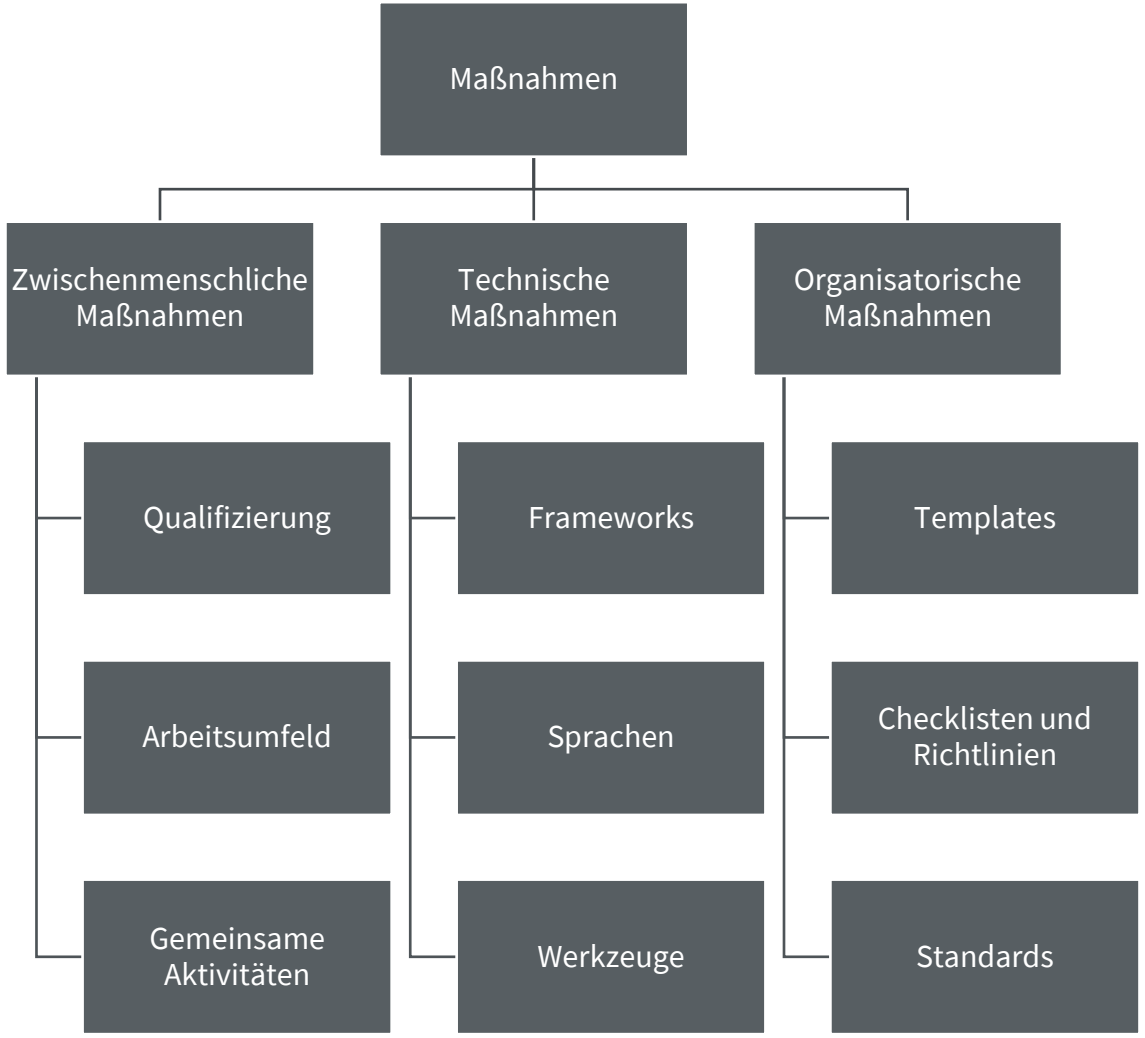
ÜBERBLICK ÜBER KONSTRUKTIVE QUALITÄTSSICHERUNG

1. Die Grundidee des Konstruktiven Qualitätsmanagements ist, dass die Qualität des Produktes von der Qualität des Erstellungsprozess abhängig ist.

2. Das Ziel? Fehlervermeidung bei Aktivitäten zur Konstruktion, zum Betrieb und der Weiterentwicklung von Softwaresystemen

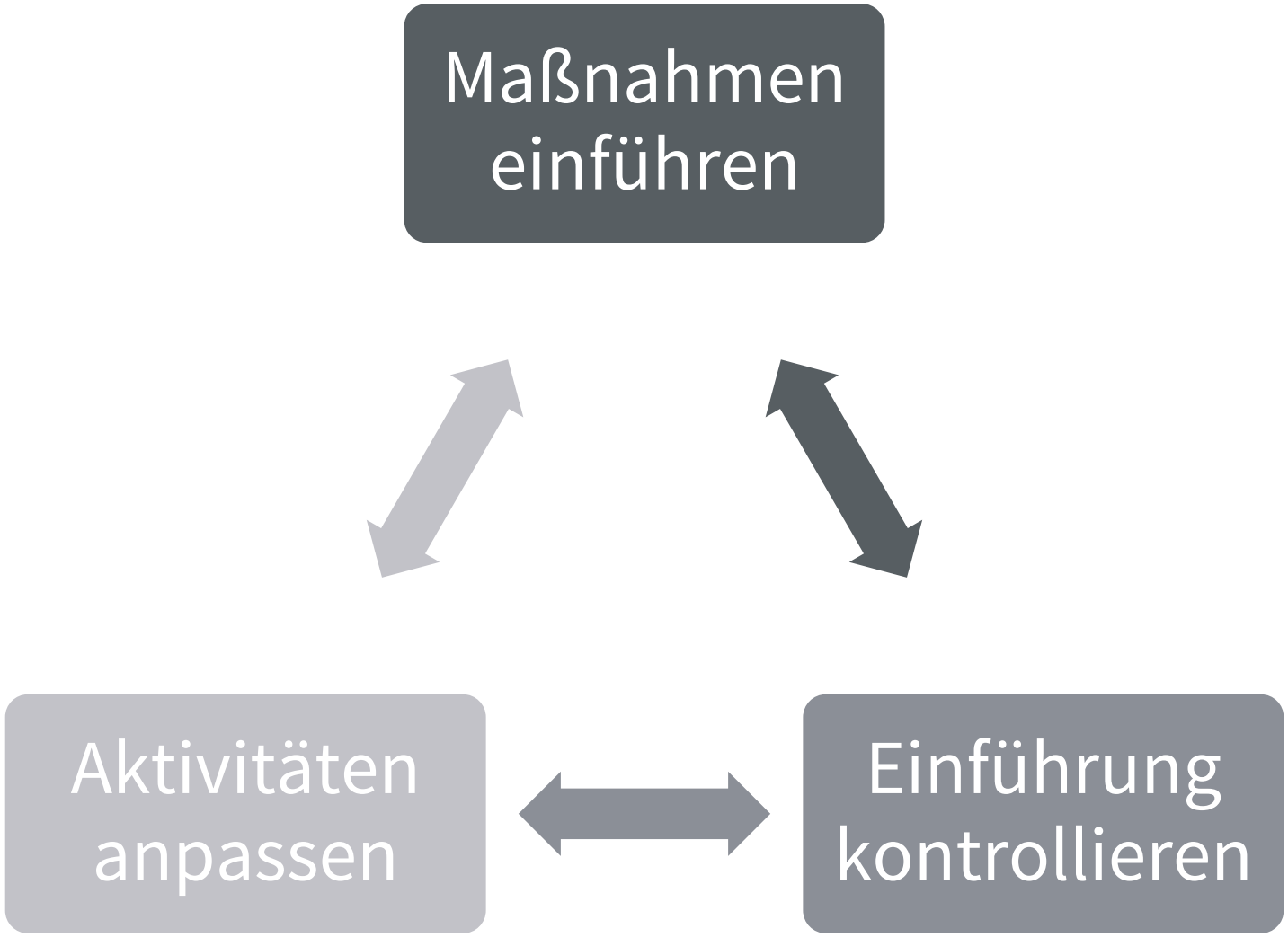
3. Teil der Qualitätsplanung ist es die konstruktiven Maßnahmen zu planen

GESAMMELTE MAßNAHMEN KONSTRUKTIVE QUALITÄTSSICHERUNG



Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Konstruktivistisches Qualitätsmanagement. IU.

AKTIVITÄTEN FÜR EINE KONSTRUKTIVE QUALITÄTSSICHERUNG



Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Konstruktivistisches Qualitätsmanagement. IU.

AKTIVITÄTEN FÜR EINE KONSTRUKTIVE QUALITÄTSSICHERUNG

1. Einsatz eines Vorgehensmodells, das in jeder Phase explizit eine Risikoabschätzung für den Entwicklungsprozess vorsieht;

2. Verwendung eines Werkzeugs zur Konfigurationsverwaltung von Quell- und Objektcode;

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Konstruktivistisches Qualitätsmanagement. IU.

AKTIVITÄTEN FÜR EINE KONSTRUKTIVE QUALITÄTSSICHERUNG

3. Dokumentenmuster für einen Qualitätsmanagementplan mit vorgegebenem Gliederungsschema und einer Anleitung für das Ausfüllen, das sicherstellt, dass alle wichtigen Punkte behandelt werden;

4. Aufstellen von Programmierrichtlinien, die unter anderem die sprechende Benennung von Variablen fordern und Unit-Tests für jede fachliche Funktion im Programmcode verpflichtend macht.

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Konstruktivistisches Qualitätsmanagement. IU.

DIE 5-WHY METHODE

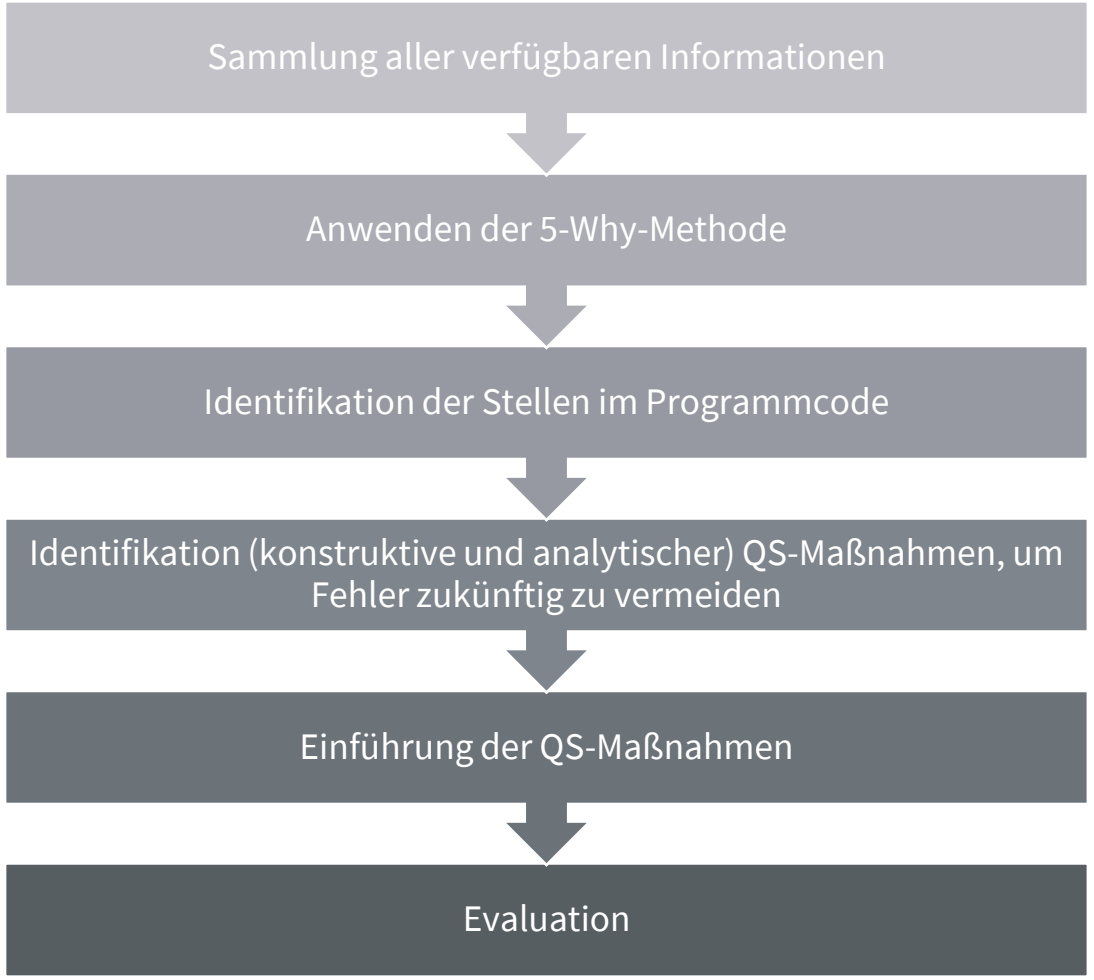
Die Kernidee dieser Analysetechnik kann im folgenden Satz auf den Punkt gebracht werden:

Bei einem Problem 5x fragen, warum das Problem existiert

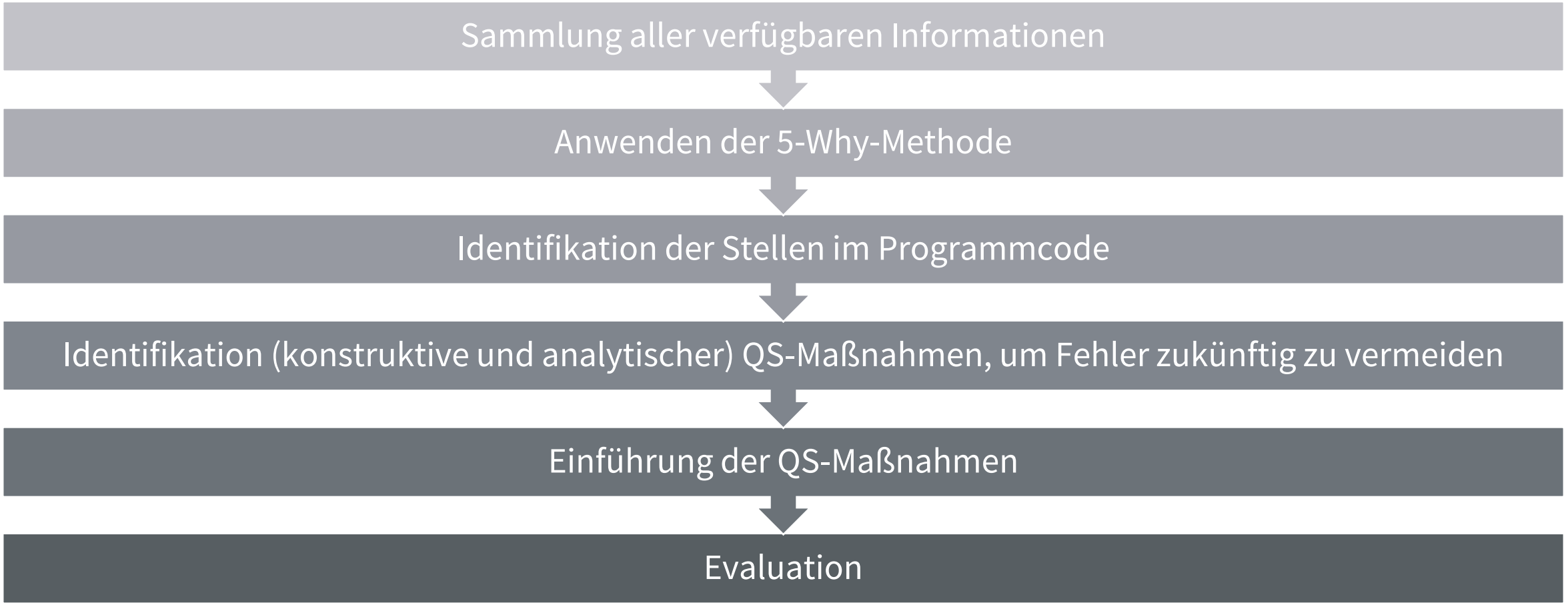
→ Zu einem Fehler soll fünfmal hintereinander die Frage „Warum“ gestellt werden, dann hat man sehr wahrscheinlich die ursprüngliche Ursache gefunden.

ROOT CAUSE ANALYSE

Im Rahmen einer Root Cause-Analyse (deutsch sinngemäß: Analyse der ursprünglichen Ursache) wird versucht, die eigentliche Fehlerursache von Fehlersymptomen zu identifizieren.



ROOT CAUSE ANALYSE



DIE 5-WHY METHODE

Die Kernidee dieser Analysetechnik kann im folgenden Satz auf den Punkt gebracht werden:

Bei einem Problem 5x fragen, warum das Problem existiert

→ Zu einem Fehler soll fünfmal hintereinander die Frage „Warum“ gestellt werden, dann hat man sehr wahrscheinlich die ursprüngliche Ursache gefunden.

BEISPIELE: 5-WHY METHODE

1. **Warum-Frage**: Warum wurde der Fehler gemeldet?
   Antwort: Weil die Berechnung Gesamtsumme häufig etwas von der Summe der Einzelpositionen abweicht.

2. **Warum-Frage**: Warum weicht die Gesamtsumme ab?
   Antwort: Weil bei der Berechnung von Rabatten für Premiumkunden Rundungsfehler auftreten.

3. **Warum-Frage**: Warum treten Rundungsfehler auf?
   Antwort: Weil bei der internen Berechnung von Zwischenergebnissen diese nicht in der erforderlichen Genauigkeit gespeichert werden.

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Konstruktivistisches Qualitätsmanagement. IU.

CHECKLISTEN – WELCHE KRITERIEN MACHEN EINE GUTE CHECKLISTE AUS?

1. Ein Punkt auf der Checkliste entspricht genau einem Ziel, zusammengesetzte Ziele sollten vermieden werden und auf mehrere Punkte aufgeteilt werden.
2. Eine klare und einfache Formulierung der Checkpunkte erhöht die Lesbarkeit und die Verständlichkeit.
3. Eine Checkliste sollte nicht größer als ein DIN A4-Blatt sein, sodass sie mit jedem Drucker einfach ausgedruckt werden kann und alle Checkpunkte auf einen Blick erfasst werden können.
4. Die zur Erfüllung eines Checkpunktes erforderlichen Kriterien müssen transparent sein, bei Bedarf werden zu einzelnen Checkpunkten Unterchеckpunkte oder eigene Checklisten erstellt.

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Konstruktivistisches Qualitätsmanagement. IU.

ÜBUNG KONSTRUKTIVES QUALITÄTSMANAGEMENT

Sie treten in ein Unternehmen ein, das Individualsoftware entwickelt:

- Es gibt ein Entwicklerteam, welches auch die Tests durchführt.
- Je nach Arbeitslast werden die Entwickler unterschiedlichen Teams zugeordnet.
- Bei Fehlermeldungen durch den Kunden wird ein beliebiger Entwickler den Fehler beheben. Jeder Entwickler arbeitet an einem selbst gewählten Arbeitspaket.
- Die Entwickler arbeiten mit unterschiedlichen Entwicklungsumgebungen, der Quellcode liegt auf einem Server.

Welche konstruktiven Maßnahmen würden Sie einführen? Gruppendiskurs

04

INTERNATIONALE HOCHSCHULE

STATISCHE QUALITÄTSSICHERUNG: BEGUTACHTUNG UND MESSEN

LERNZIELE

- Einsatzgebiete statischer Verfahren
- Rollen und Aktivitäten in den verschiedenen Review-Techniken
- Einsatzgebiete von Metriken & typische Softwaremetriken
- Statische Codeanalyse

STATISCHE CODEANALYSE

- „Grundsätzlich können alle statischen Analysen auch von Hand durchgeführt werden.

- Eine manuelle Durchführung besitzt aber deutliche Nachteile:
  - Neben dem hohen Zeitbedarf und
  - dem erforderlichen Aufwand
  - ist damit zu rechnen, dass die Analyseergebnisse eine **schlechtere Qualität besitzen werden als automatisch erzeugte Ergebnisse.** “

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 270

STATISCHE CODEANALYSE - GRUNDLAGEN

Was ist die statische Codeanalyse?

- Qualitative Bewertung von Programmcode
- Inhaltliche Auswertung des Quellcodes
- Typischer Anwendungsfall: automatische Stilüberprüfung & Analyse nach typischen Fehlermustern
- Typische Werkzeuge: CheckStyle, PMD, FindBugs

STATISCHE CODEANALYSE

- Manuelle Prüfungen sind besonders in frühen Phasen der Software-Entwicklung, z. B. für die Überprüfung von Entwurfsdokumenten wichtig.
  - Sie können auch ergänzend zu dynamischen Tests auf Quellcode angewandt werden.
  - Eine positive Eigenschaft von manuellen Prüfungen ist, dass sie in der Lage sind, semantische Aspekte zu beachten.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 305-306

STATISCHE CODEANALYSE

- Die Qualität vieler Merkmale z. B. Verständlichkeit, Änderbarkeit, Aussagefähigkeit von Bezeichnern und Kommentaren – kann nur manuell geprüft werden.

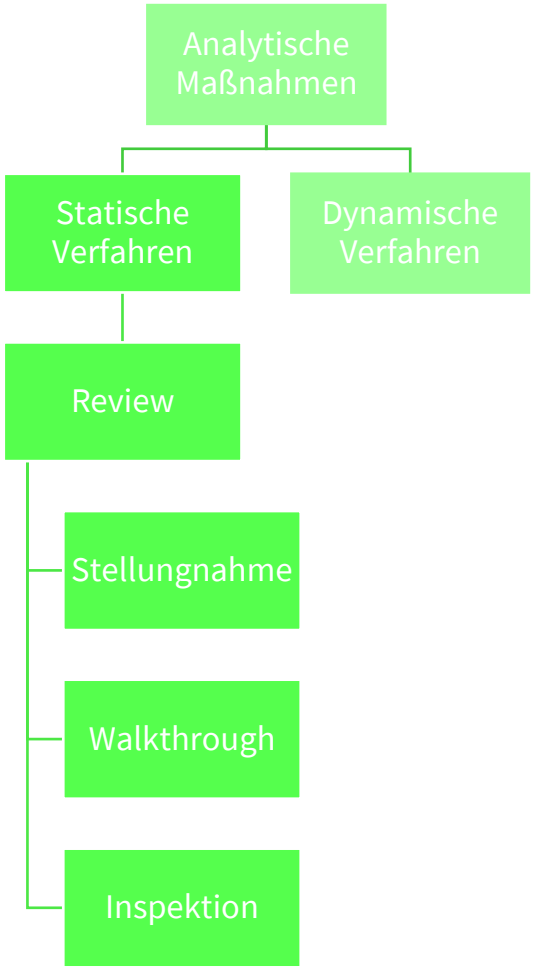
- Es bietet sich an, eine Inspektion bzw. ein Review am Ende jeder frühen Entwicklungsphase als Zwischenabnahme zu verwenden. So wird sichergestellt, dass ein qualitätsgerechtes Ergebnis in die nachfolgende Software-Entwicklungsphase übergeben wird.

- **Das Bestehen der Prüfung kann als Meilenstein genutzt werden.**

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 305-306

EINSATZ UND ÜBERBLICK ÜBER STATISCHE VERFAHREN

Bei statischen Maßnahmen wird das zu testende Artefakt nicht ausgeführt!



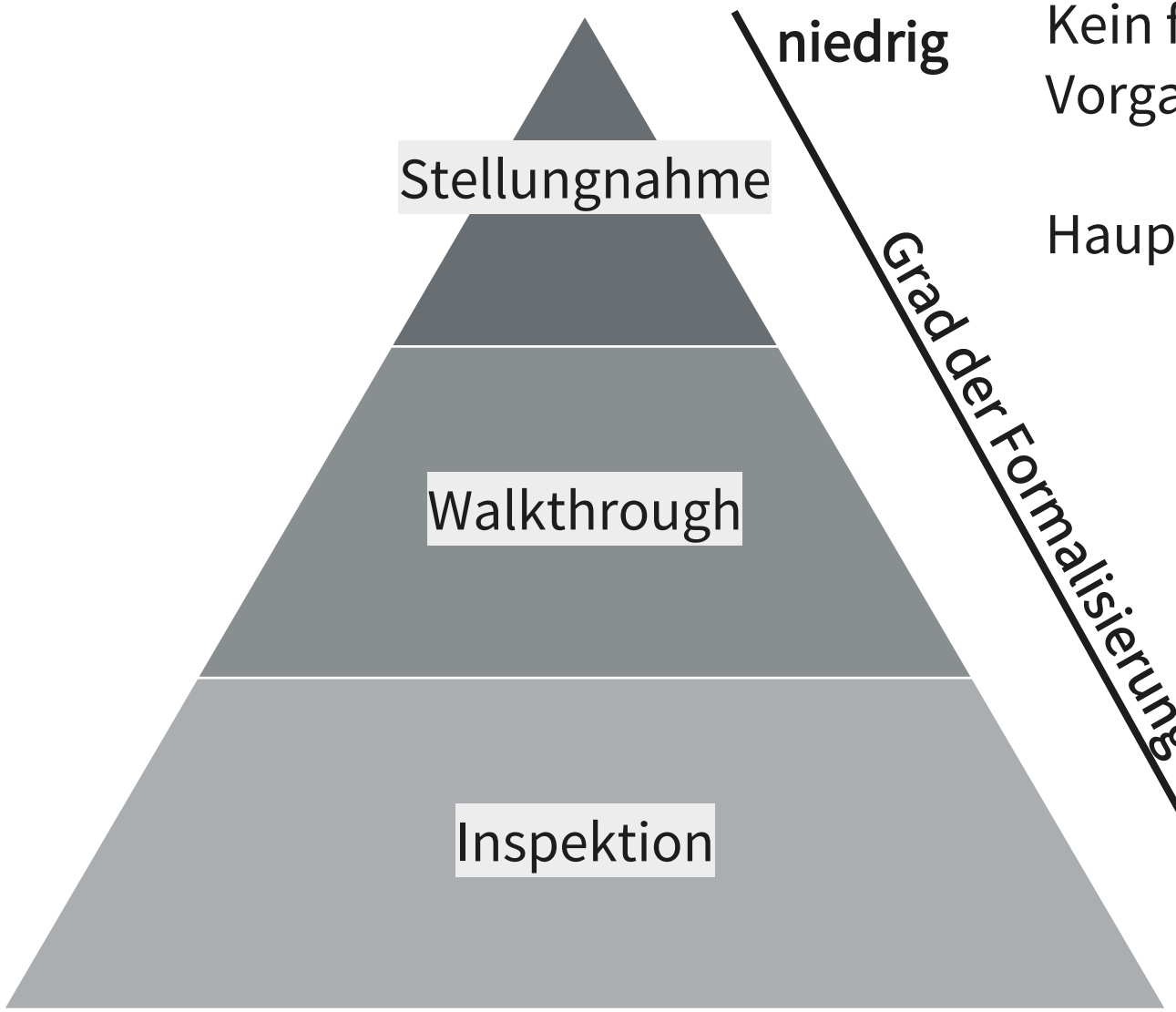
Einsatzgebiete:

- fachlichen Anforderungen
- technischer Spezifikation / Testfälle
- Quality Gates
- Bewertung der geplanten Architektur
- Komponententests

DIE STATISCHE CODEANALYSE FINDET FOLGENDE FEHLER

- Referenzieren einer Variablen mit nicht definiertem Wert
- Typenkonflikte
- Inkonsistente Schnittstellen zwischen Modulen und Komponenten
- Variablen, die nie verwendet werden
- Unerreichbarer (toter) Code (»dead code«)
- Verletzung von Programmierkonventionen
- Sicherheitsschwachstellen
- Syntax-Verletzungen von Code und Softwaremodellen

BEGUTACHTEN MIT REVIEW-TECHNIKEN



Kein formaler Prozess, keine/wenige Vorgaben an Gutachter

Hauptziel: Kostengünstig & Nutzen erzielen

Einbindung eines Review-Teams, dokumentierte Ergebnisse und Abläufe

Hauptziele: Fehlerzustände finden. Lernen, Verständnis erwerben,

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Statische Qualitätssicherung. IU.

EINORDNUNG VON INSPEKTIONEN

- Inspektionen werden in allen Phasen einer Software-Entwicklung eingesetzt.

- Es gibt Inspektionen von Anforderungsdokumenten, von Entwürfen, von Quelltexten und auch von Testfällen.

- Bei den Prüfkriterien handelt es sich typischerweise um einfach prüfbare, syntaktische Regeln.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 309

ÜBUNG ZU INSPEKTIONEN (DIE STELLUNGSNAHME)

1. Entwerfen Sie im Team eine möglichst präzise Anforderungsbeschreibung (z. B. in Bezug auf das Referat oder aus ihrer Fallstudie – wählen Sie eine Funktion aus).

   Alternativ: Sie erhalten von Ihrem Kunden den Auftrag, eine Software zu entwickeln, die die Umrechnung eines Betrags von einer Währung in eine andere Währung ermöglicht. Beschreiben Sie diese Funktion präzise.

2. Tauschen Sie die Anforderungsbeschreibung (übersenden) mit einem anderen Team
3. Nehmen Sie eine Inspektion (ist die Anforderung nachvollziehbar) vor

Je 30 Min., Diskurs über die Learnings

ÜBUNG ZU INSPEKTIONEN (FFM/HAN) – TEIL 2

INTERNATIONALE HOCHSCHULE

Es tauschen bitte:

- Team A mit Team C
- Team B mit Team A
- Team C mit Team B

Jedes Team erstellt bitte eine Zusammenfassung:

- Was war verständlich?
- Wo ist noch etwas unverständlich?
- „Was ging gar nicht“?

Diskurs: Was hat euch komplett gefehlt?

SPEZIFIKATIONEN & FEEDBACK

REVIEWS

- Reviews und Inspektionen verlangen die Durchführung einer Review-, bzw. Inspektionssitzung. Man spricht auch von manuellen Prüfungen in Sitzungstechnik.

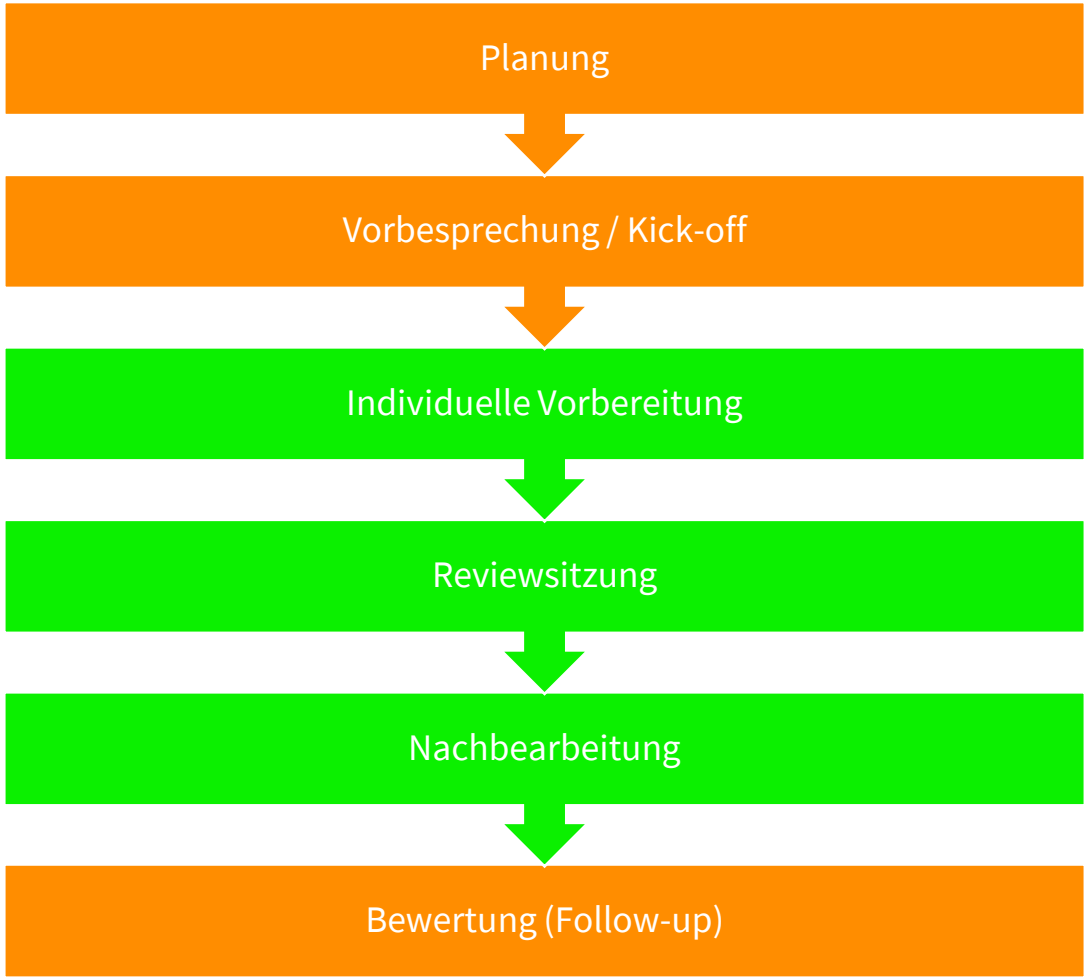
- Reviewtechniken gibt es unter einer Vielzahl von Bezeichnungen mit zum Teil unklaren Abgrenzungen und oft nur geringfügigen Unterschieden. Typische Bezeichnungen sind Peer Review und Structured Walkthrough.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 317

ABLAUF VON INSPEKTIONEN UND WALKTHROUGHS

Planung:

- Festlegung von wichtigen Rahmenbedingungen
- Anfangs-, Endbedingungen
- Prüfbedingungen
- Zusammenstellung des Reviewteams
- Einbindung des Autors
- Organisatorische Planung

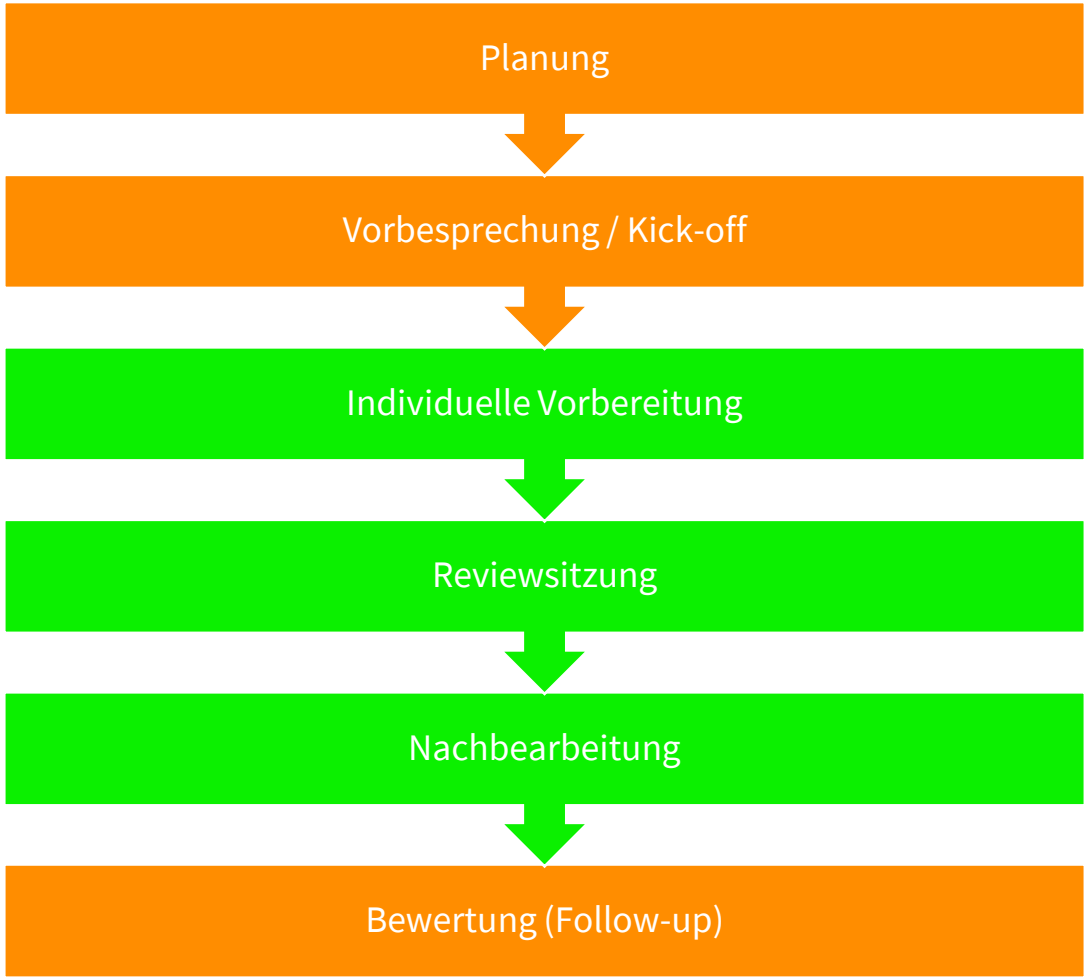


- Bei einem Walkthrough verpflichtend
- Bei einer Inspektion zusätzlich verpflichtend

ABLAUF VON INSPEKTIONEN UND WALKTHROUGHS

Vorbesprechung:

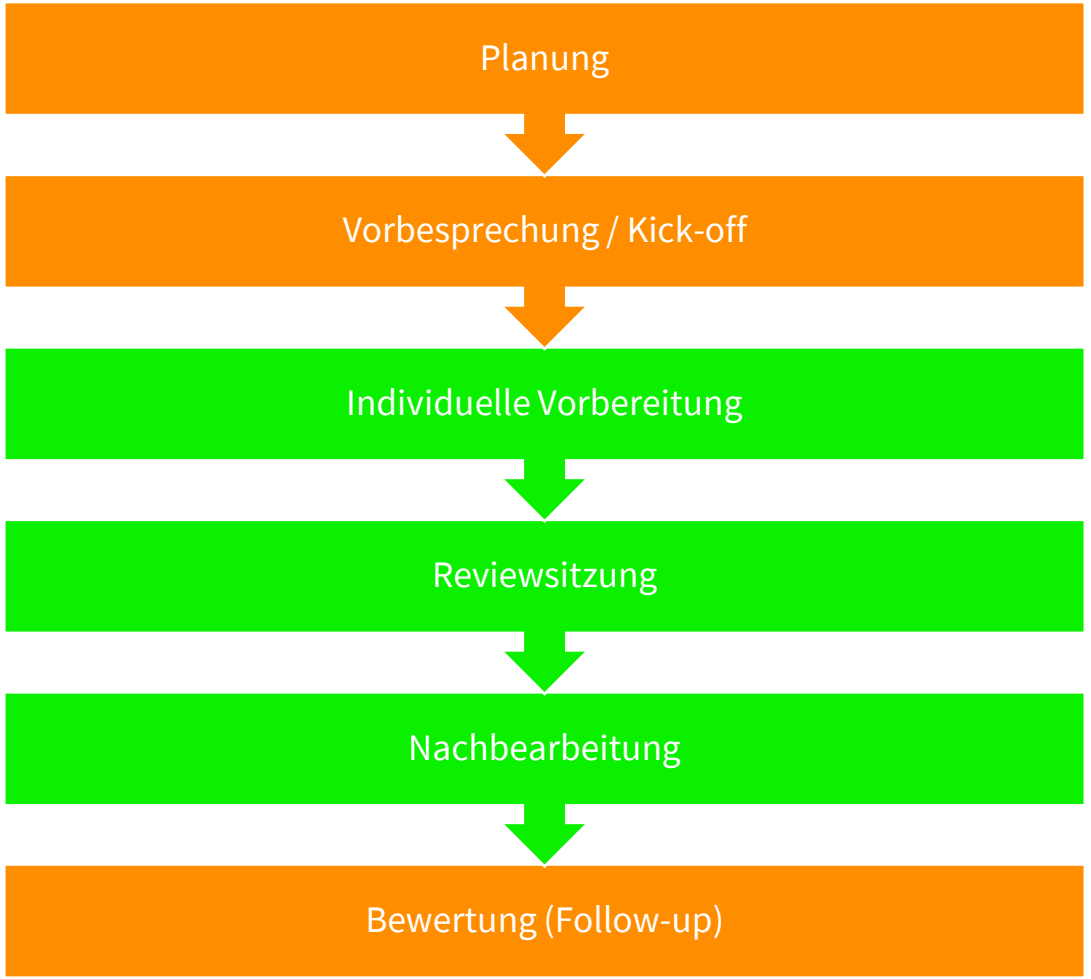
- Nur falls notwendig
  - Wann ist es nicht notwendig?
  - Wann ist alles bekannt?
- **Alle Beteiligten erhalten die notwendigen Informationen!**



- Bei einem Walkthrough verpflichtend
- Bei einer Inspektion zusätzlich verpflichtend

ABLAUF VON INSPEKTIONEN UND WALKTHROUGHS

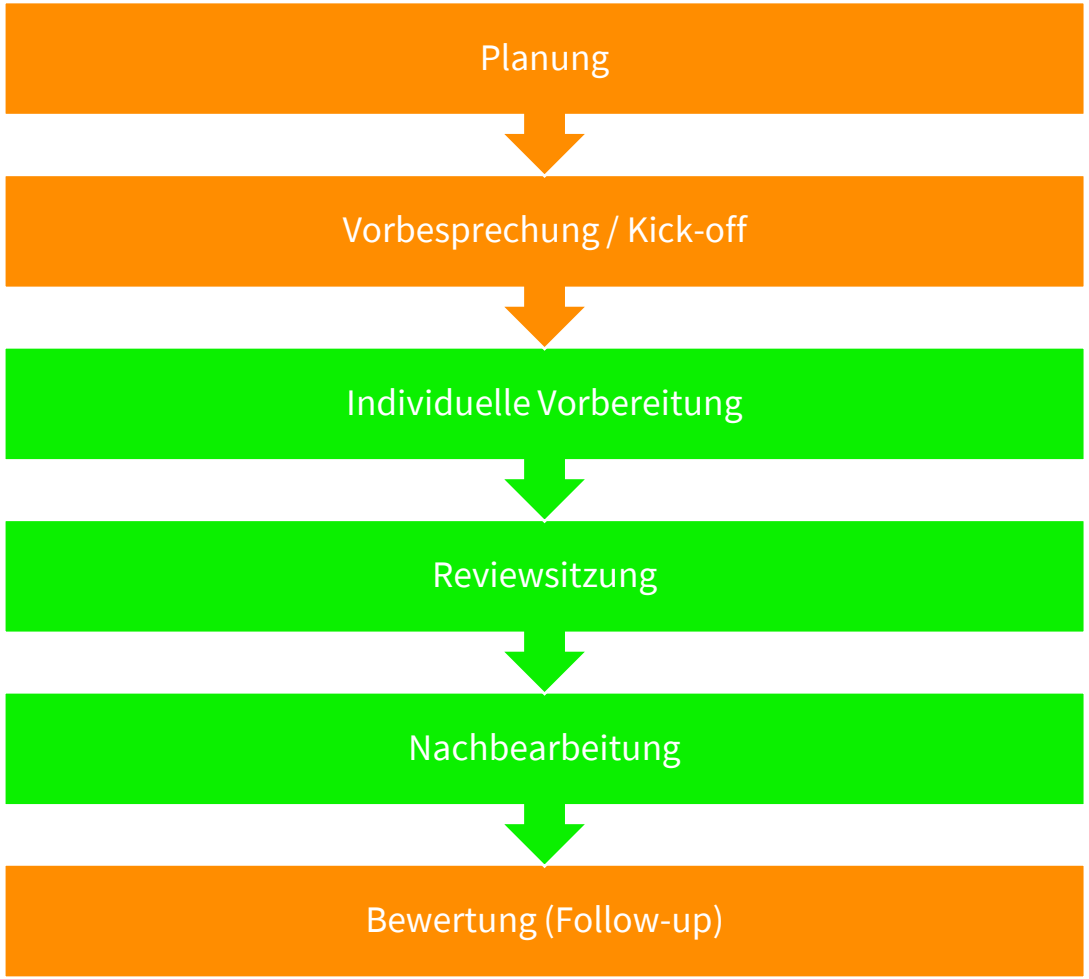
- Neben dem Dokument, das einem Review unterzogen werden soll, müssen den beteiligten Personen weitere Unterlagen zur Verfügung stehen:
  - Dokumente, die herangezogen werden müssen, um zu entscheiden, ob eine Abweichung, ein Fehler oder eine korrekte Beschreibung des Sachverhalts vorliegt, also die Dokumente (z.B. Pflichtenheft, Richtlinien oder Standards), gegen die geprüft wird



- Bei einem Walkthrough verpflichtend
- Bei einer Inspektion zusätzlich verpflichtend

ABLAUF VON INSPEKTIONEN UND WALKTHROUGHS

- Neben dem Dokument, das einem Review unterzogen werden soll, müssen den beteiligten Personen weitere Unterlagen zur Verfügung stehen:
  - Darüber hinaus sind Prüfkriterien (z.B. in Form von Checklisten) sehr sinnvoll, die ein strukturiertes Vorgehen unterstützen

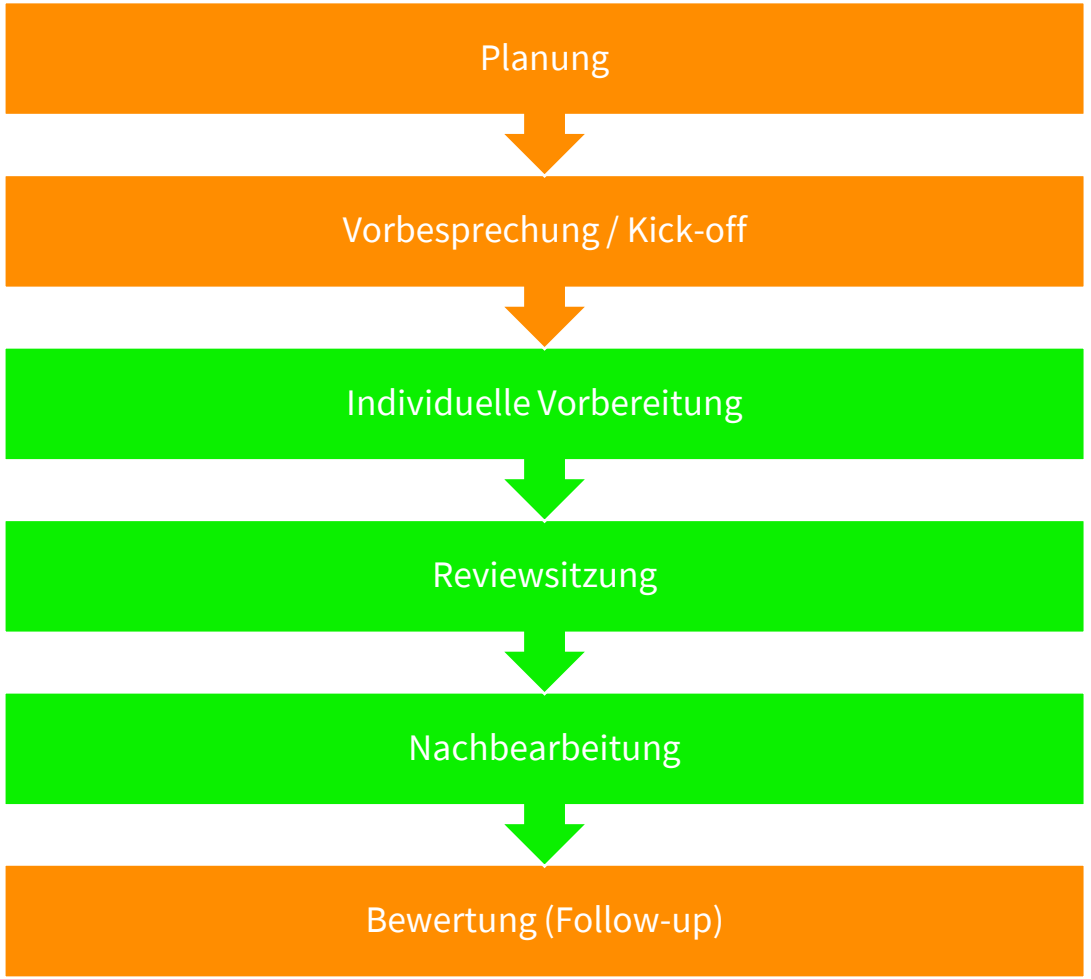


- Bei einem Walkthrough verpflichtend
- Bei einer Inspektion zusätzlich verpflichtend

ABLAUF VON INSPEKTIONEN UND WALKTHROUGHS

Reviewsitzung:

- In der Reviewsitzung wird der Prüfling durchgegangen und die Punkte aus der Vorbereitung genannt
- Reviewteam gibt Empfehlung über die Annahme des Prüflings ab:
  - Am Schluss geben alle Sitzungsteilnehmer das Protokoll frei
  - Der Moderator hat das Recht, eine Sitzung abzusagen oder zu vertragen

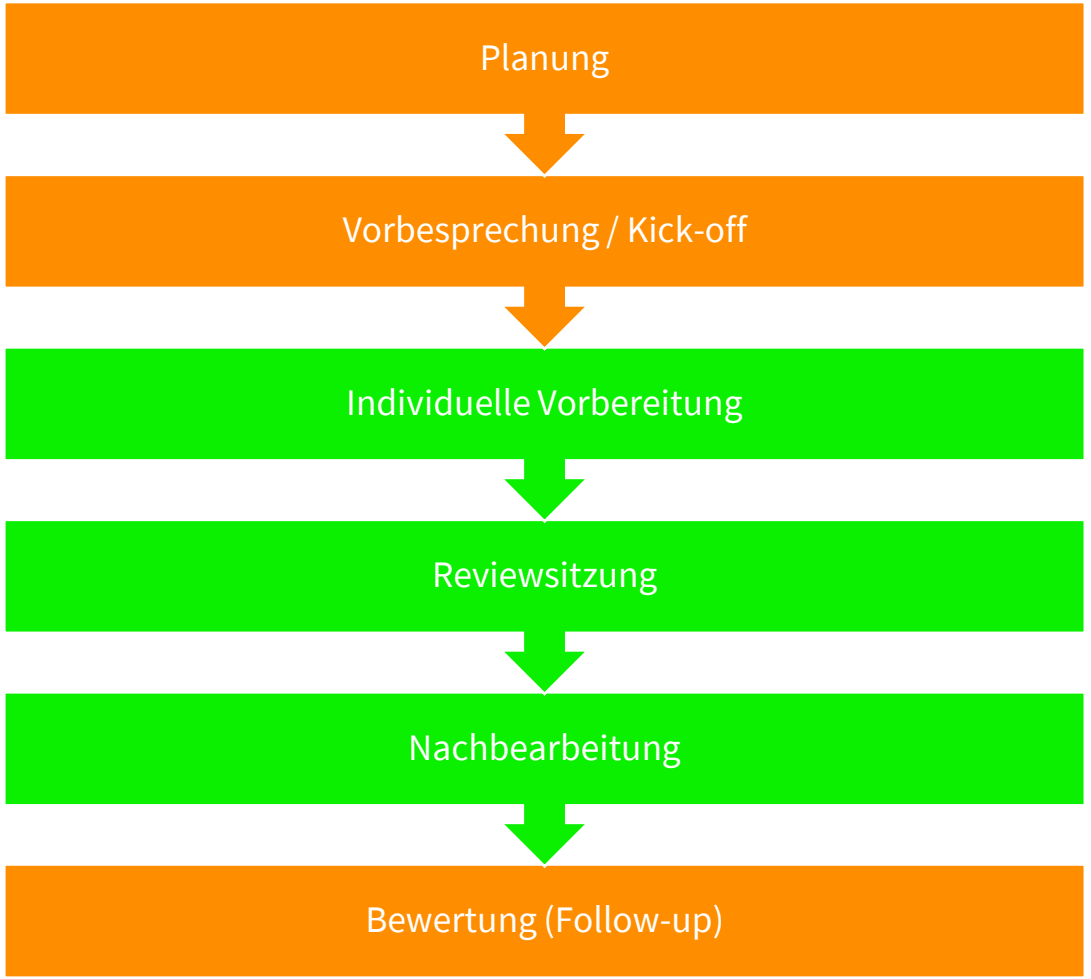


- Bei einem Walkthrough verpflichtend
- Bei einer Inspektion zusätzlich verpflichtend

ABLAUF VON INSPEKTIONEN UND WALKTHROUGHS

Überarbeitung (im Rahmen der Reviewsitzung):

- Der Autor überarbeitet den Prüfling anhand des Protokolls

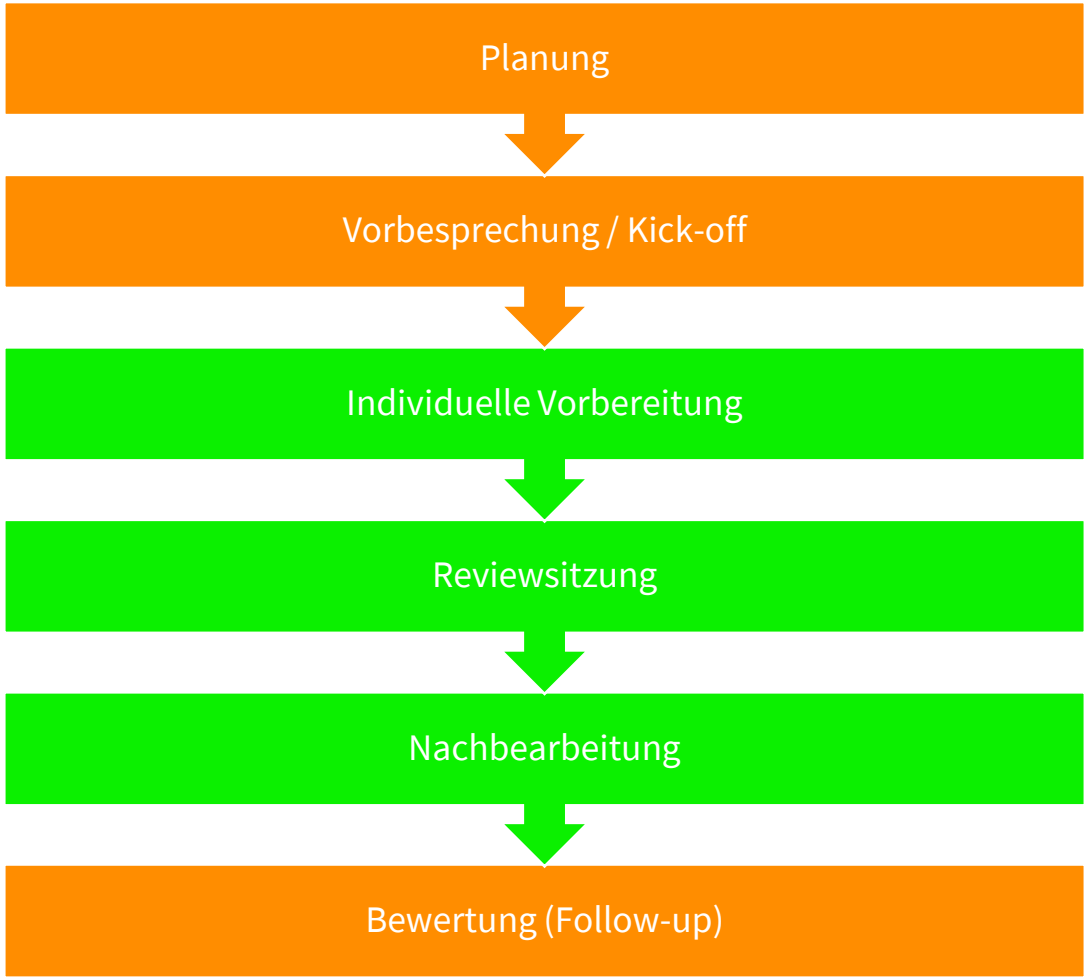


- Bei einem Walkthrough verpflichtend
- Bei einer Inspektion zusätzlich verpflichtend

ABLAUF VON INSPEKTIONEN UND WALKTHROUGHS

Nachbereitung:

- Kontrolle der ordnungsgemäßen Durchführung der Überarbeitung
- Prüfung, dass die aufgedeckten Fehlerzustände durch die Korrekturen auch beseitigt wurden
- Prüfung von Endkriterien (bei mehr formalen Reviewarten)

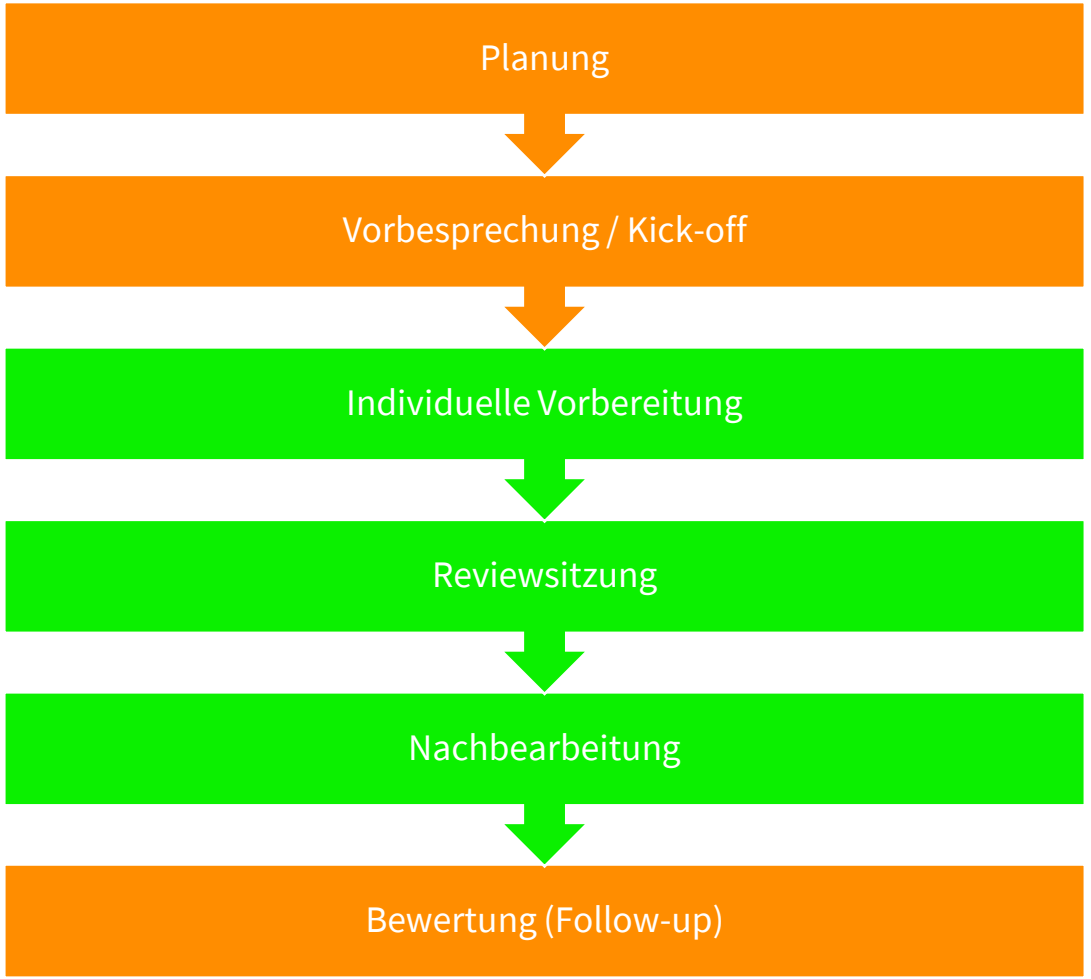


- Bei einem Walkthrough verpflichtend
- Bei einer Inspektion zusätzlich verpflichtend

ABLAUF VON INSPEKTIONEN UND WALKTHROUGHS

Nachbereitung:

- War das Resultat des ersten Reviews nicht akzeptabel und muss ein weiteres Review durchgeführt werden, so ist die hier beschriebene Vorgehensweise für das zweite Review entsprechend durchzuführen, jedoch meist verkürzt



- Bei einem Walkthrough verpflichtend
- Bei einer Inspektion zusätzlich verpflichtend

ZUSAMMENFASSUNG VON ROLLEN UND AKTIVITÄTEN

	Ziele	Ergebnis	Verantwortlich
Planung	▪ Organisation aller benötigten Ressourcen (wie Personen, Termine, Räume) ▪ Bestimmung der Prüfziele ▪ Bestimmung der zu „Bestanden“-Kriterien, ▪ Benennung der Gutachter	▪ Mit allen Beteiligten allen abgestimmte Terminplanung ▪ Dokumentierte Prüfziele und Bestanden-Kriterien	▪ Moderator
Vorbesprech-ung (bei Bedarf)	▪ Alle Beteiligten erhalten einen Überblick über den Prüfling und die Prüfziele. ▪ ggf. Aufteilung der Prüfaspekte / Sichten der Prüfung auf Gutachter ▪ Verteilung aller benötigten Unterlagen ▪ Klären von organisatorischen und fachlichen Fragen	▪ Jeder Gutachter kennt den Prüfling, das Ziel der Prüfung und die Prüfkriterien, für die er verantwortlich ist.	▪ Moderator

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Statische Qualitätssicherung. IU.

ZUSAMMENFASSUNG VON ROLLEN UND AKTIVITÄTEN

	Ziele	Ergebnis	Verantwortlich
Individuelle Vorbereitung der Gutachter	▪ Prüfling wird durch Gutachter individuell durchgearbeitet und anhand der festgelegten Kriterien geprüft	▪ Dokumentierte Fragen des Gutachters, sowie dokumentierte inhaltliche Mängel und formale Mängel (wie Schreibfehler)	▪ Gutachter
Reviewsitzung	▪ Gemeinsames Durchgehen durch den Prüfling durch alle Beteiligten ▪ Diskussion und Bewertung der durch Gutachter identifizierten fachlichen Mängel	▪ Protokoll aller identifizierten fachlicher Mängel ▪ Sammlung aller formalen Fehler (werden nicht in Sitzung besprochen) ▪ Dokumentiertes Ergebnis (keine Mängel, Nacharbeit nötig, Abbruch wegen schwerwiegender Mängel)	▪ Moderator

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Statische Qualitätssicherung. IU.

ZUSAMMENFASSUNG VON ROLLEN UND AKTIVITÄTEN

	Ziele	Ergebnis	Verantwortlich
Nachbereitung	▪ Korrigieren der erkannten Mängel ▪ Bei Inspektion: Mängelbericht und Erledigungsliste der Korrekturen erstellen	▪ Überarbeiteter Prüfling ▪ Mängelbericht, Erledigungsliste	▪ Autor
Bewertung	▪ Prüfen, ob alle Korrekturen umgesetzt wurden ▪ Abschließende Bewertung des Prüflings	▪ Reviewbericht	▪ Moderator

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Statische Qualitätssicherung. IU.

GEGENÜBERSTELLUNG VON REVIEW-TECHNIKEN

	Stellungnahme	Walkthrough	Inspektion
Charakteristik	▪ schnelle, informaler Review ohne explizite Organisation	▪ informaler Review durch mehrere Gutachter	▪ formales Review durch mehrere Gutachter nach definiertem Schema
Anwendungs-szenarien	▪ Schnelles, informales Feedback zum Prüfling erhalten ▪ Internes Feedback unter Kollegen ohne Pflicht zum Nachweis formaler Reviewkriterien	▪ Detaillierte Prüfung wichtiger Artefakte, in denen nicht identifizierte Fehler ein deutliche merkbaren Schaden verursachen können ▪ Schaffung eines einheitlichen Verständnisses von Sachverhalten und Förderung der Teamkommunikation	▪ Prüfung wichtiger Artefakte, in denen nicht identifizierte Fehler ein sehr hohes Schadenspotential haben ▪ Pflicht zum Nachweis von formalen Kriterien der Prüfung

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Statische Qualitätssicherung. IU.

INSPEKTIONSROLLEN

Name der Rolle	Stellungnahme	Walkthrough	Inspektion
Moderator	-	X (bei Bedarf vom Autor zur übernehmen)	X
Gutachter	X	X	X
Autor (des Prüflings)	X	X	X
Protokollführer	-	-	X

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Statische Qualitätssicherung. IU.

AUFGABEN DER ROLLEN

- Moderator:
  - Erfahrungsgemäß haben Inspektionen eine Tendenz, nach der Findung einiger Fehler in eine Diskussion der Lösungskonzepte abzugleiten. Dies ist unerwünscht, da es die Effizienz der Inspektion reduziert.
  - Der Moderator muss dies verhindern.
  - Diskussionen sind nur zu Fehlern und Fehlerarten erlaubt.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 311-312

AUFGABEN DER ROLLEN

- Autor:
  - Der Autor ist der Ersteller des inspizierten Produkts.
  - Er ist für die Korrektur der während der Inspektion gefundenen Fehler zuständig.

- Gutachter:
  - Führt das Inspektionsteam durch die Sitzung.
  - Er trägt die technischen Inhalte erläuternd vor.
  - Muss in der Lage sein, die unterschiedlichen Teile der Arbeit zu beschreiben.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 311-312

AUFGABEN DER ROLLEN

- Protokollant:
  - Der Protokollführer notiert und klassifiziert alle Fehler, die in der Inspektion gefunden werden.
  - Darüber hinaus unterstützt er den Moderator bei der Anfertigung der Inspektionsberichte.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 311-312

FALLSTRICKE VON REVIEWS

1. Fehlende Zeit (Fehleinschätzung bei der Ressourcen-Planung)
2. Fehlende Vorbereitung der Gutachter
3. Fehlende oder unklare Ziele, Reviews nicht als Maßnahme zur Qualitätssicherung, sondern als ‚Anschwärzen‘
4. Rollen und Verantwortlichkeiten werden missverstanden
5. Fehlende oder unzureichende Dokumentation
6. Benötigte Dokumente sind nicht verfügbar, wurden in der Vorbereitung übersehen
7. Benötigtes Personal ist nicht in ausreichendem Maße vorhanden
8. Personal verfügt nicht über die erforderliche Ausbildung
9. Fehlende Unterstützung durch das Management

FEEDBACK GEBEN – DER FEEDBACK „BURGER“ – ZUR RICHTIGEN KOMMUNIKATION VON FEHLERN

1. Positiver Einstieg
- Du hast das gut gemacht, ...
- Mir hat besonders gefallen...

2.1 Konkrete Beobachtung
- Ich habe gesehen/gelesen, dass...
- Mir ist aufgefallen, dass...
- Du hast gesagt/geschrieben, dass...

2.2 Wirkung
- Es hat auf mich gewirkt, als ob...
- Ich hab verstanden, dass...
- Ich fand es merkwürdig...
- Ich habe bemerkt, dass...

2.3 Wunsch bzw. Vorschlag
- Ich schlage vor, dass...
- Ich hätte gerne...
- Vielleicht kannst du...
- Warum versuchst du nicht,...

3. Positiver Abschluss
- Insgesamt hat mir das gut gefallen,...
- Mit kleinen Änderungen bin ich sicher, dass die Sache noch runder wird...

Lob

Kritik

Lob

CODE-REVIEW-ÜBUNG

```java
public static void main(String[] args) {
  Scanner scan = new Scanner(System.in);
  List<Product> products;
  int input = 0; int i = 0;
  do {
     i ++;
     System.out.println("Was möchten Sie tun? 1 = Produkt anlegen,
     2 = kommentieren, 3 = beenden");
     input = scan.nextInt();
     if(input == 1) {
       System.out.println("Name: ");
       String n = scan.next();
       products.add(new Product(i, n, ""));
     }
     if(input == 2) {
       System.out.println("Welches Produkt? (ID)");
       long id = scan.nextLong();
       Product p = getProduct(products, id);
       System.out.println("Kommentar: ");
       String c = scan.next();
       p.comments.add(new Comment(c));
     }
  }while(input != 3);
}
```

Comment.java

```java
public class Comment {
   public Comment(String c) {
   comment = c;
   }
   public long ProduktID;
   public long commentID;
   public String comment;
   public String autor;
}
```

Product.java

```java
public class Product {
  private long id;
  private String name;
  private String description;
  public List<Comment> comments;
  public Product(long pId, String pName, String description) {
    setId(pId);
    name = pName;
    description = description;
  }
  public void setId(long id) {
    this.id = id;
  }
}
```

GRUPPENARBEIT

- Führt eine Inspektion/Walkthrough in eurer Gruppe (4 Personen) durch
  - Übernehmt die Rollen in der Gruppe (Moderator, Gutachter, Autor -des Prüflings-, Protokollführer)
  - Wir diskutieren und vergleichen am Ende das Protokoll (bzw. den Inspektionsbericht)
- Lass auch Chat GPT o. ä. den Quellcode prüfen und vergleicht die Ergebnisse mit euren
- 30 Minuten Zeit (+10 Min Prüfung Chat GPT)

ERGEBNISSE

- Gutachter:
- Protokollant:
- Moderator:
- Autor:

DISKURS

- Was sind eure allgemeinen Learning aus dem Vorgehen?

- Wie würde ein solcher Vorgang im Referat aussehen?
  - Wie viel Zeit wäre notwendig?
  - Wie solltet ihr das planen?
  - Wie kann vorher/nachher sinnvoll dargestellt werden?

REMINDER: DEFINITION DER QUALITÄTSZIELBESTIMMUNG

- „**Qualitätsmerkmal**, Eigenschaft einer Funktionseinheit, anhand derer ihre Qualität beschrieben und beurteilt wird, die jedoch keine aussage über den Grad der Ausprägung enthält“

- „**Qualitätszielbestimmung,** Definition der für ein Produkt zu erreichenden Qualität durch Festlegung der angestrebten Ausprägungen der Qualitätseigenschaften. Die Qualitätszielbestimmung wird zu Beginn der Entwicklung durchgeführt.“

→ Festlegung erfolgt z. B. im Lasten- / Pflichtenheft

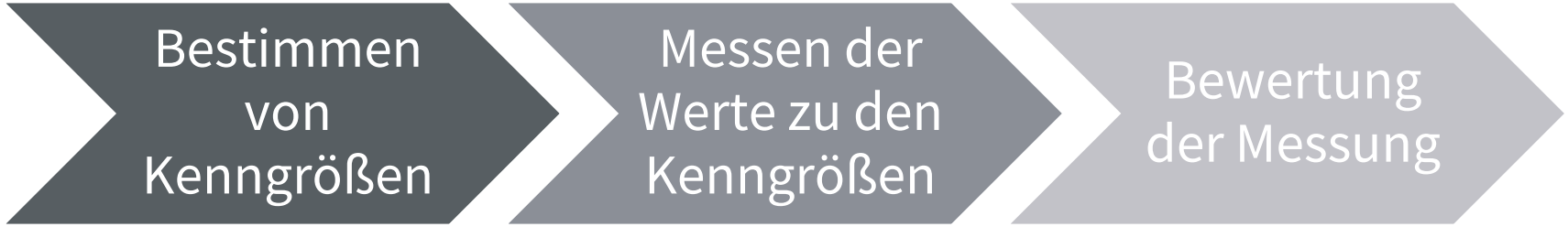
Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 515.

METRIK BEGRIFFSDEFINITION

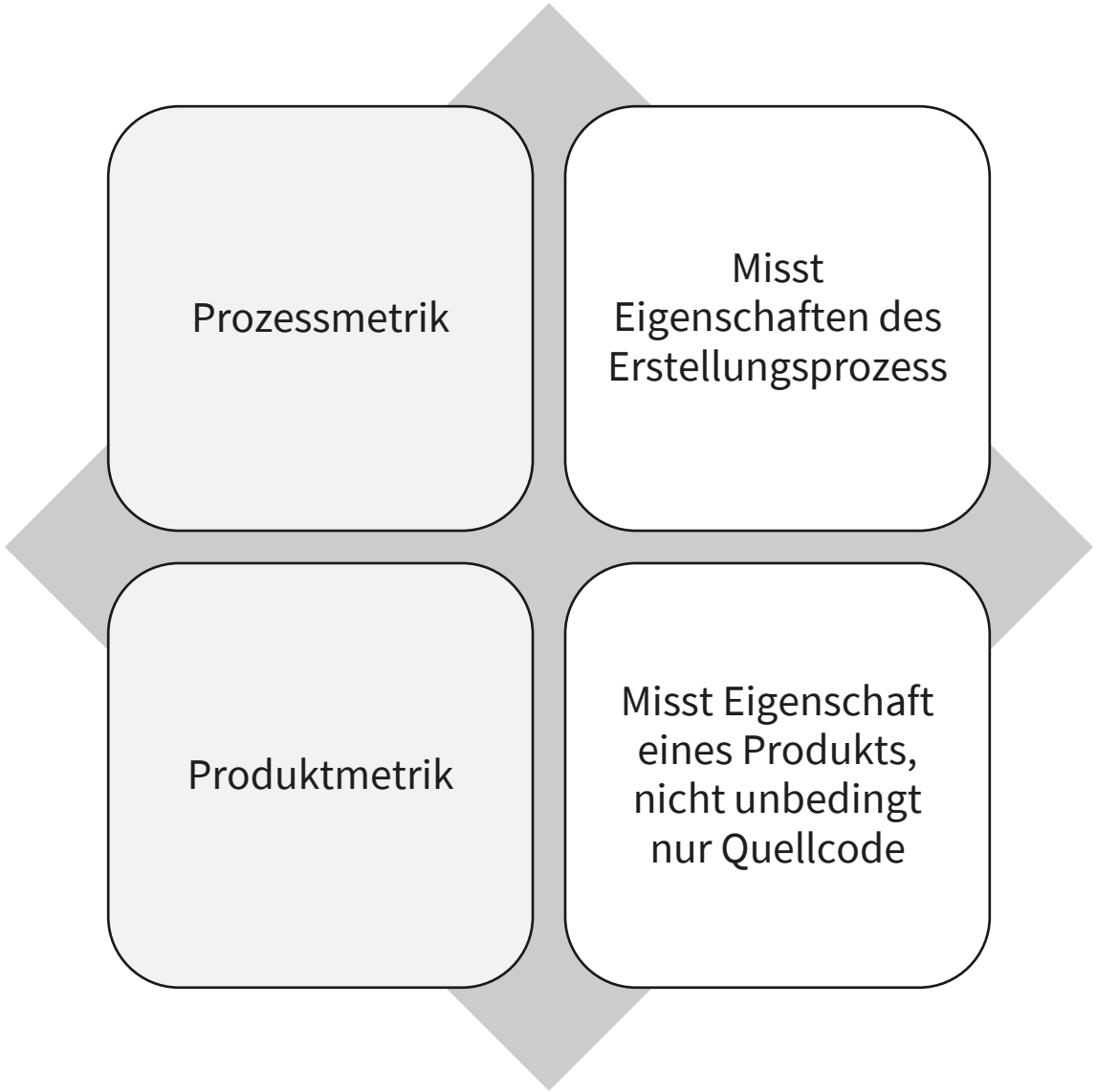
- Grundsätzlich können Eigenschaften, sowohl von im Softwareprozess erzeugten Artefakten als auch von Eigenschaften des Softwareprozesses, selbst gemessen werden.

- Die Funktion, die aus den Ergebnissen der Vermessung eines Prüflings einen Zahlenwert ermittelt, wird Metrik genannt.

METRIK PROZESS



METRIK TYPEN



METRIK BEISPIELE

- Verbrauchte Ressourcen (Zeit, Geld, Personal)
- Anzahl der identifizierten Fehler im Projekt
- Dauer einzelner Aktivitäten
- mittlere Behebungszeit von identifizierten Fehlern
- Anzahl von Änderungen an einer Systemkomponente
- Umfang (Anzahl Seiten, LOC, Anzahl Modellelemente)
- Komplexität,
- Qualität (Anzahl identifizierter fachlicher Fehler, Anzahl identifizierter formaler Fehler)
- Stil (Einhaltung von Konventionen, Lesbarkeit, Redundanzfreiheit)

VOR- UND NACHTEILE VON METRIKEN

Vorteile	Nachteile
Relativ einfache Ermittlung	Nur messbare Ereignisse können ermittelt werden
Programmiersprachenunabhängig	Keine Aussagemöglichkeit für Akzeptanz durch Kunden
Mittel um vielfältige Eigenschaften zu Ergebnisartefakten	Eher Indiz für Qualität

DISKURS

Sie sind für die Qualitätssicherung bei Ihrer Software aus Ihrem Software Engineering Projekt zuständig:

- Welche Metriken können Sie während der Entwicklung zur Qualitätssicherung nutzen?
- Welche Metriken können Sie später zum Reporten zum Management nutzen?

DATENFLUSSANOMALIEANALYSE

- Datenflussanomalieanalyse → Datenflussdarstellungen von Kontrollflussgraphen

- Es werden Datenflussattribute hinzugefügt, die den Knoten bzw. Kanten des Kontrollflussgraphen zugeordnet sind und angeben, welche Variablen in welcher Weise zugegriffen werden.

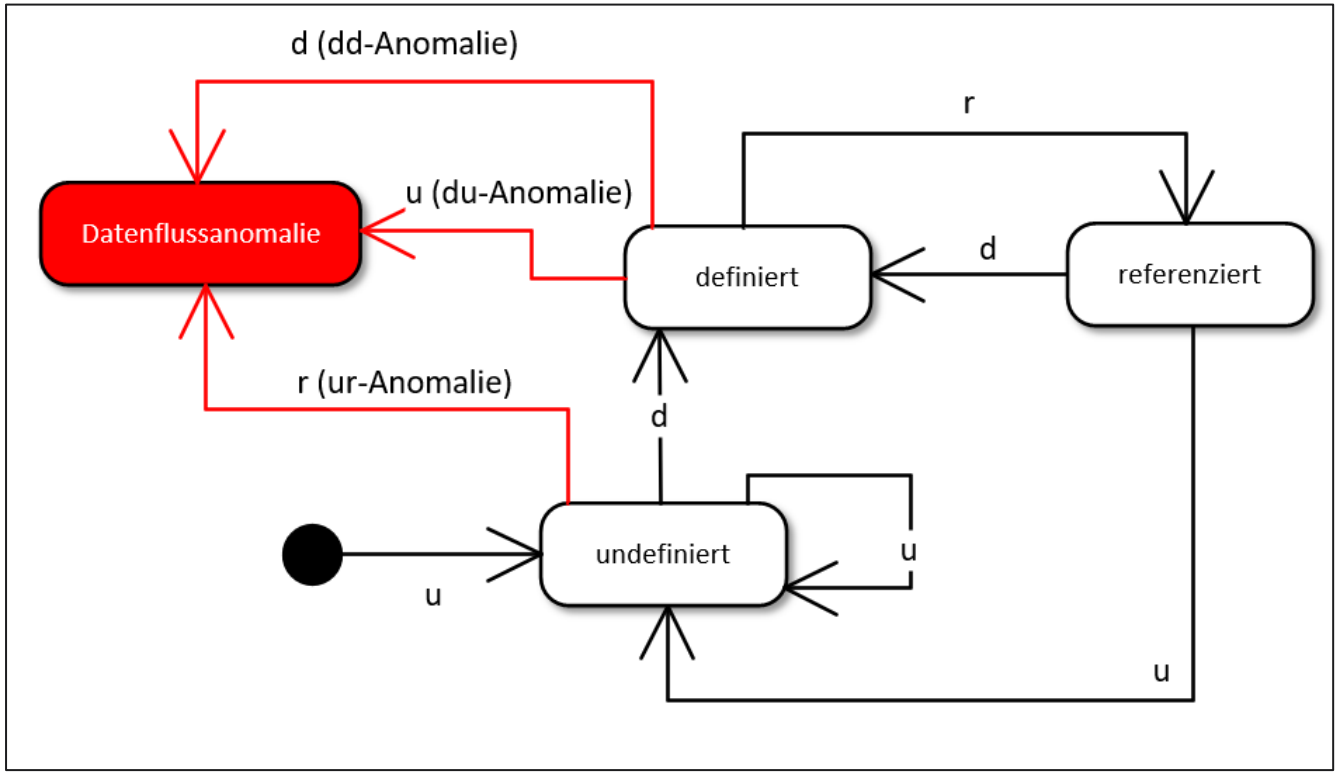
Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 280

DATENFLUSSANOMALIEANALYSE

- Zugriffssequenzen auf eine Variable:

1. u r d r u u
2. u d d r d u

Sequenz 1 beginnt mit u r. Die betrachtete Variable besitzt zum Zeitpunkt der Referenz einen zufälligen Wert, da sie nicht zuvor definiert wurde. […]



(u: Undefinition, d: Definition, r: Referenz)

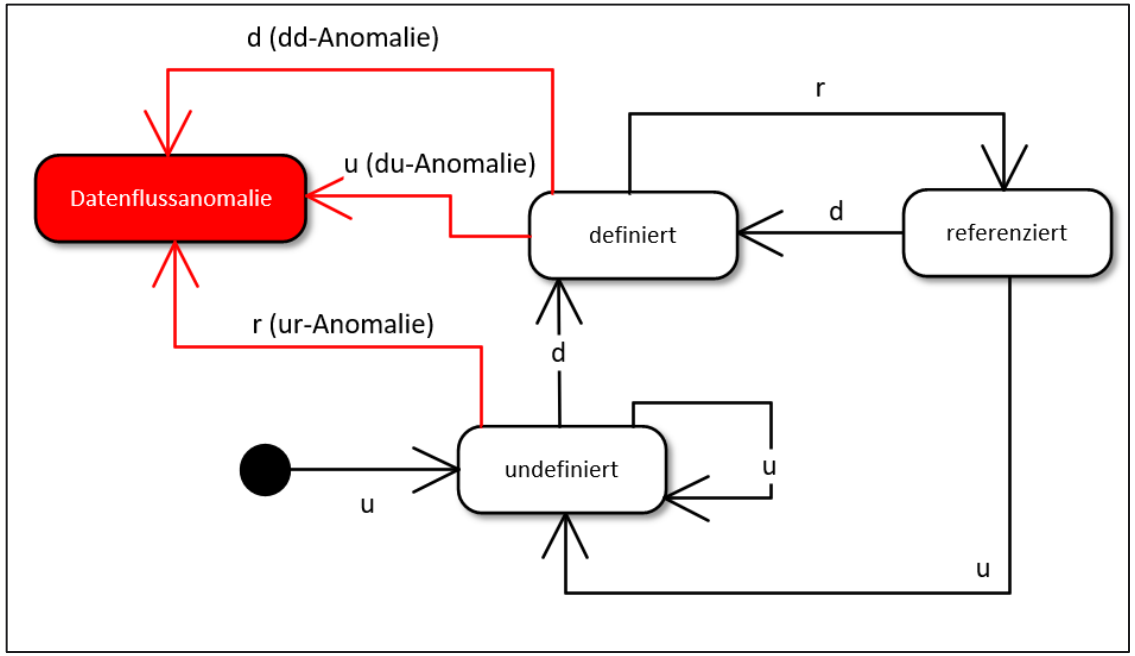
Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 295

DATENFLUSSANOMALIEANALYSE

- Zugriffssequenzen auf eine Variable:

1. u r d r u u
2. u d d r d u

Sequenz 2 enthält zwei aufeinanderfolgende Variablendefinitionen. Die erste Definitionen besitzt keine Wirkung, da der Wert stets durch die zweite Definition überschrieben wird. Die Datenflussanomalie ist vom Typ dd.



(u: Undefinition, d: Definition, r: Referenz)

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 295

CODEBEISPIEL EINER DATENFLUSSANOMALIEANALYSE

- Bei der Datenflussanomalienanalyse reicht es aus, die Pfade bis zum zweiten Schleifendurchlauf zu analysieren.
- Sind bis dort keine Datenflussanomalien aufgetreten, so ist sichergestellt, dass auch auf den Pfaden mit einer höheren Anzahl von Schleifeniterationen keine Anomalien auftreten werden.

```
public static void main(String[] args) {
// varA --> Undefiniert (u)
        {
        // Definition (d)
        double varA = 3.0;
        // Referenz (r) und Definition (d)
        varA = varA * 2.0;
        // Referenz (r) und Definition (d)
        varA = varA - 3.0;
        System.out.println(varA);
        }
// varA --> Undefiniert (u)
System.out.println(varA);
}
```

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 298

ÜBUNG DATENFLUSSANOMALIEANALYSE - KLEINGRUPPENDISKURS

Rechenprogramm.java

```java
public static void main(String[] args) {
    double a = Double.parseDouble(args[0]);
    double b = Double.parseDouble(args[1]);
    double ergebnis = a + b;
    ergebnis = a * b;
    System.out.println(ergebnis);
    c = a * b;
    ergebnis = b / c;
    System.out.println(ergebnis);
    ergebnis = a + b;
}
```

- Untersuchen Sie den Datenfluss des Rechenprogramms auf Anomalien
  - Welche Anomalien stellen Sie fest?
  - Wie bewerten Sie diese?
  - 10 Min. in Referatsgruppen

ERGEBNISSE

05

INTERNATIONALE HOCHSCHULE

DYNAMISCHE QUALITÄTSSICHERUNG: TESTEN

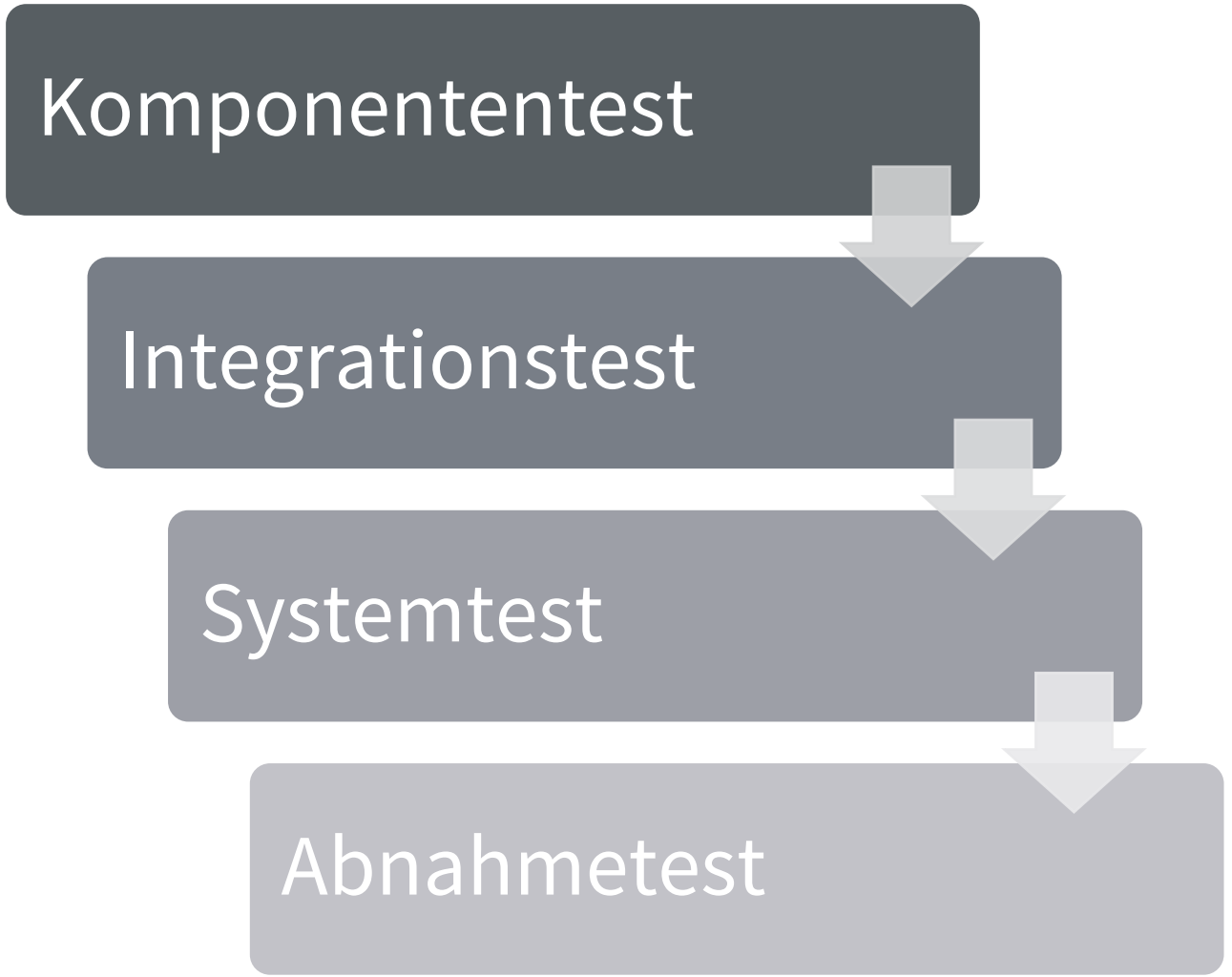
LERNZIELE

- Grundlagen zu Testfall und Teststufen
- typischen Techniken zur Testfallerstellung
- anwendungsfallbasierte Testfallerstellung
- Äquivalenzklassenbildung
- zustandsbasierten Testfallerstellung
- Zufallsdaten

TESTSTUFEN

- Testen erfolgt (soll erfolgen) entwicklungsbegleitend
- Teststufen orientieren sich an Entwicklungsstufen

- Qualitätssicherung von Software geschieht in mehreren Teststufen
- Welche Eigenschaften getestet werden hängt auch von den Teststufen ab



Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

TESTFALLSPEZIFIKATION: GRUNDGRÖßE TESTFALL

Handlungsanweisung zur Durchführung von Softwaretests

besteht (wenigstens) aus folgenden Elementen

Beschreiben Daten zum Testfall (z.B. Name, ID, …)	Vorbedin-gungen	Testaktionen	Testdaten	Nachbedin-gungen

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

BEISPIEL TESTFALLDEFINITION 1

Template für konkrete Testfälle	
ID des Testfalls	Eindeutige ID
Testgegenstand	Offizieller Name der zu testenden Systemkomponente oder der Schnittstelle oder der Funktion des Systems/der Komponente
Getestetes Detail	Name der Klasse(n) oder Komponente(n), die bei diesem Test getestet werden
Methode(n), die getestet werden	Detaillierter Name der Methode/der Funktionen/der Schnittstelle und deren Parameterliste
Kurzbeschreibung des Tests	Kurze Beschreibung, was mit dem Test geprüft bzw. sichergestellt werden soll
Testdaten/Eingabe	Welche Eingabedaten werden benötigt, um den Testfall durchzuführen? Mit welchen Parametern sollen die zu testenden Funktionen aufgerufen werden? Falls es sich um Objekte handelt: Welche Attribute sollen sie enthalten? Falls es sich um Dateien handelt: Welchen Inhalt haben sie?
Erwartetes Ergebnis/ Erwartete Systemreaktion	Welche Ausgaben oder Systemreaktionen werden zu den Testdaten erwartet?
Vorbereitung/Vorbedingung	Konkrete Schritte, um das System/die Komponente für den Test vorzubereiten: • Erzeugen oder Einspielen von Testdaten in die Datenbank • Aufrufen bestimmter Methoden, damit das System in den Zustand gelangt, in dem getestet werden kann (Bsp.: Anmelden von Nutzern am System) • Sicherstellen, dass bestimmte Daten in der Datenbank vorliegen

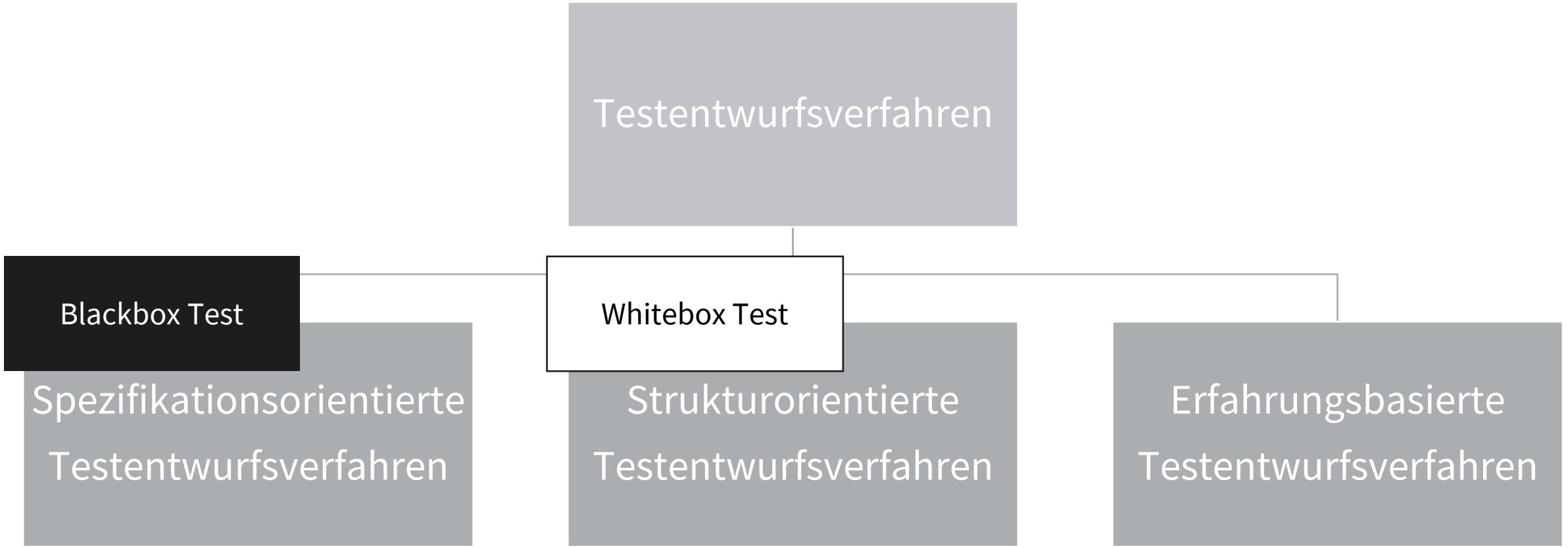
Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

BEISPIEL TESTFALLDEFINITION 2

Template für konkrete Testfälle	
ID des Testfalls	Eindeutige ID
Durchführung	Welche Systemfunktion(en)/Methode(n) muss/müssen beim Test ausgeführt werden? Bei komplexeren Testfällen mit mehreren Aufrufen: In welcher Reihenfolge müssen die Methoden/System-funktionen ausgeführt werden?
Überprüfung/Nachbe-dingung	Geben Sie hier die konkreten Bedingungen eines bestandenen Tests an. Welches Systemverhalten/welche Ausgaben werden vom System erwartet? Welche Kriterien müssen die im System gespeicherten Daten erfüllen? Bei der Erfüllung welcher Kriterien ist der Test erfolg-reich? Wie werden die Kriterien gemessen?

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

TESTENTWURFSVERFAHREN: WHITE- UND BLACKBOXTESTS



TESTENTWURFSVERFAHREN: WHITEBOXTESTS

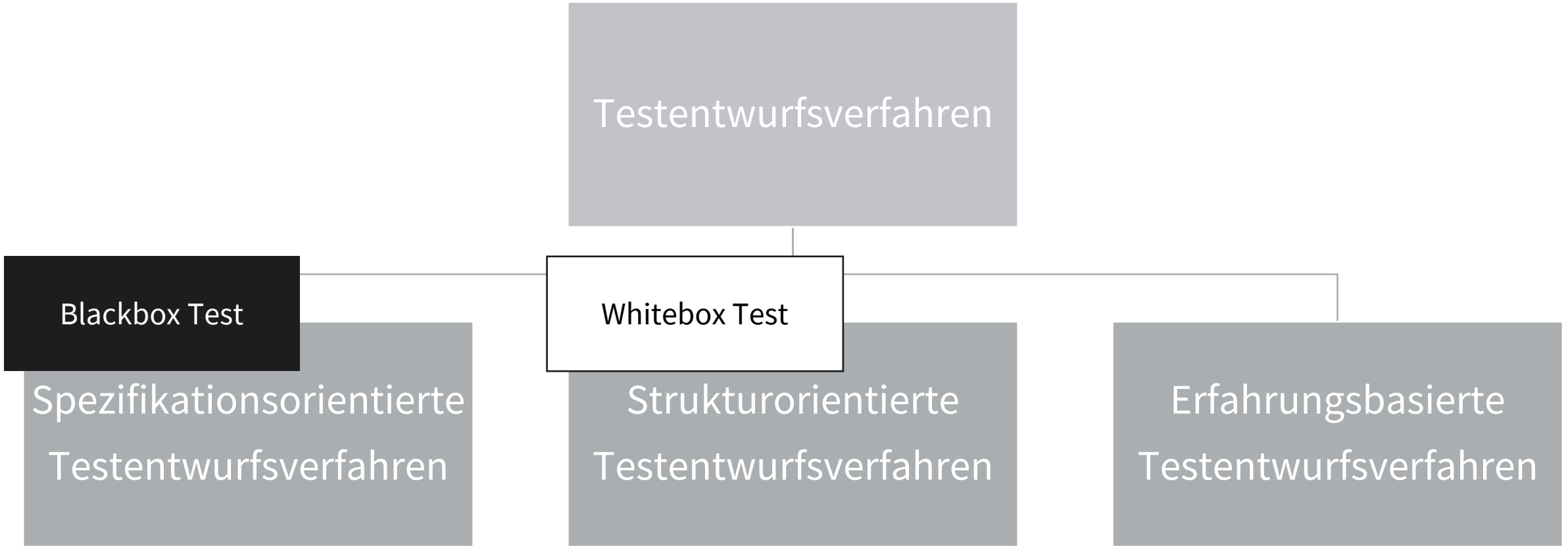
Ziel: Testabdeckung des (bekannten) Quellcodes

- meist implementierungsbegleitend / durch Entwickler
- auf Komponentenebene

Schritte:

1. Programmcode implementieren
2. Struktur des erzeugten Codes ermitteln
3. Testfälle auf Basis der bekannte Codestruktur schreiben & ausführen
4. Testergebnisse beurteilen

TESTENTWURFSVERFAHREN: WHITE- UND BLACKBOXTESTS

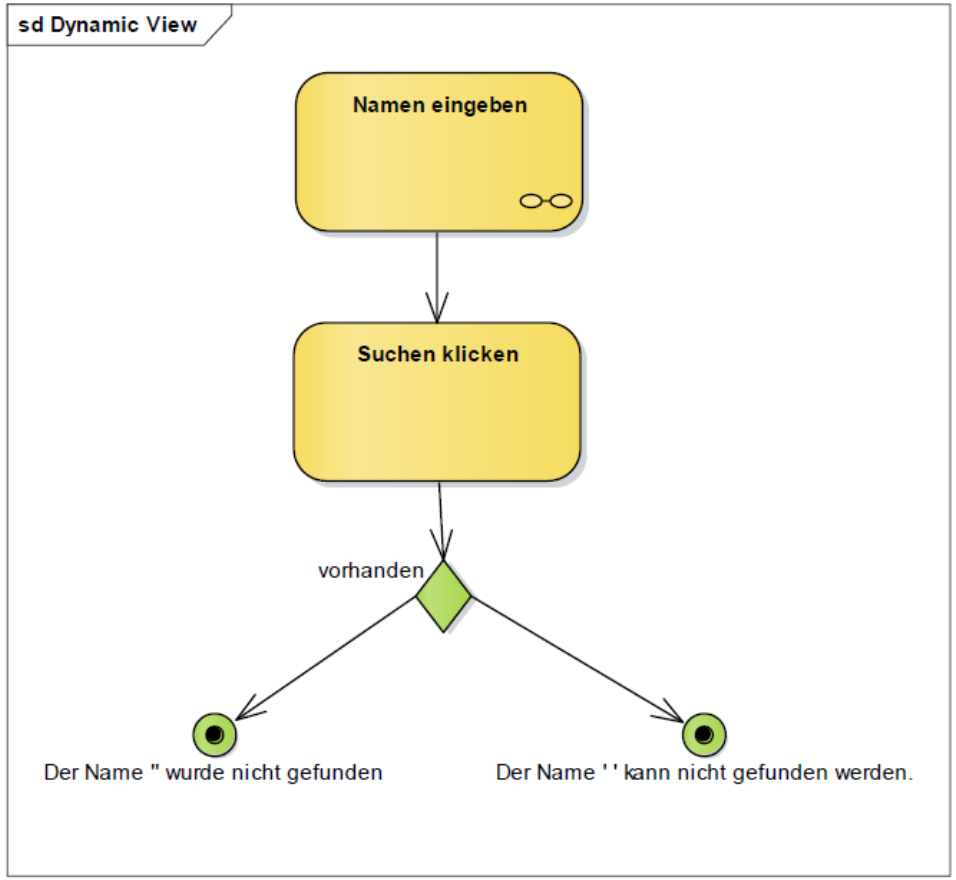


TESTENTWURFSVERFAHREN: BLACKBOXTESTS

- Anwendernah
- Prüfpunkte aus der Spezifikation

1. Spezifikation erhalten
2. Spezifikation interpretieren (was kann getestet werden)
3. Testfälle schreiben
4. Testfälle in der Software ausführen
5. Testergebnisse beurteilen (ggf. Wiedereinstieg bei 3.)

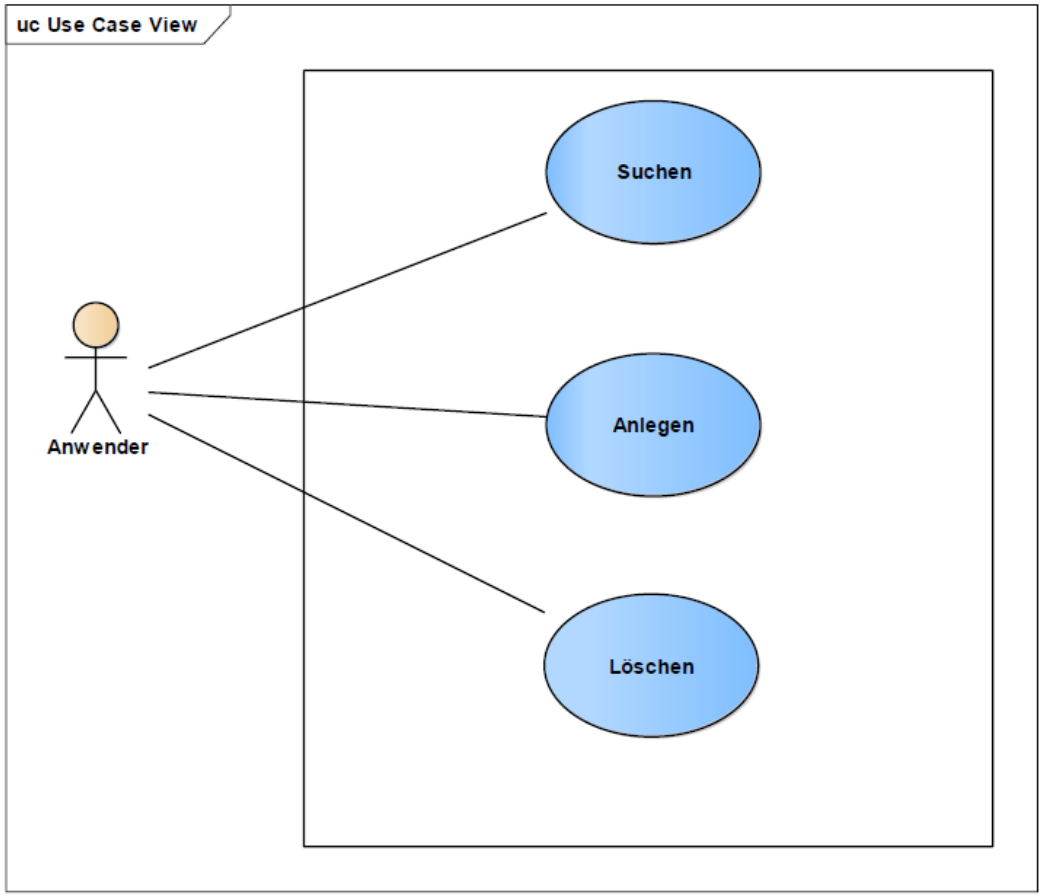
- Anwendungsfallbasierte Testfallerstellung



Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

TESTENTWURFSVERFAHREN: BLACKBOXTESTS

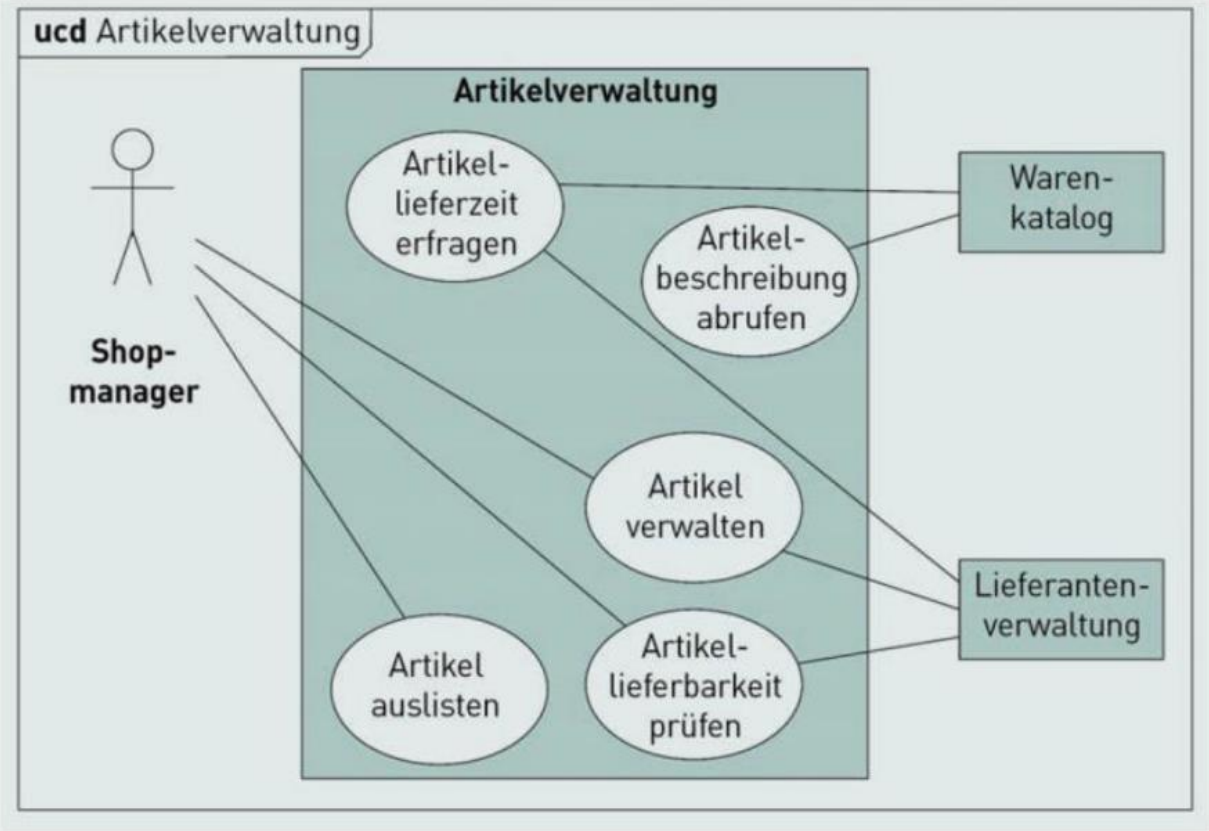
- Anwendungsfallbasierte Testfallerstellung



Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

TESTENTWURFSVERFAHREN: BLACKBOXTESTS

- Anwendungsfallbasierte Testfallerstellung



Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

TESTABDECKUNG

- Testabdeckung ist das Maß für die Vollständigkeit von Testfällen.
- Der konkrete Messwert ist abhängig von der Art des Tests und der gewählten Technik der Testfallerstellung.

ANWENDUNGSFALLBASIERTE TESTFALLERSTELLUNG

- Ansatzpunkt zur Testfallentwicklung: fachliche Anwendungsfälle
- Dokumentation z.B. über Use Case Diagramme (ein Use Case Diagramm kann dabei mehrere Anwendungsfälle beinhalten) oder Sequenzdiagramme
- Mindestanforderung: pro Anwendungsfall ein Testfall

TESTFALLAUSWAHL

Prüfen der Anwendungsfälle nach folgenden Kriterien:

- Wertbeitrag
- Nutzungshäufigkeit
- Schadenspotential
- Typische Fehler
- Geforderte Testabdeckung

FUNKTIONSORIENTIERTE TESTTECHNIKEN

- Die funktionsorientierten Testtechniken stellen die Software-Spezifikation in den Mittelpunkt des Testgeschehens → Es ist daher insbesondere darauf zu achten, dass eine zum jeweiligen Test passende Spezifikation zur Verfügung steht und auch verwendet wird.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 50

FUNKTIONSORIENTIERTE TESTTECHNIKEN

- Die Spezifikation dient zur Beurteilung der Vollständigkeit des Tests sowie zur Herleitung der Testfälle und der Beurteilung der Reaktionen der Software.

- Aufgrund ihrer Herleitung aus der Spezifikation sind die Testfälle eines funktionsorientierten Tests systematisch an der Überprüfung der Soll-Funktionalität ausgerichtet. Daher kann insbesondere geprüft werden, ob die Spezifikation vollständig in Software umgesetzt wurde.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 50

FUNKTIONALE ÄQUIVALENZKLASSENBILDUNG (BLACK BOX)

- Die funktionale Äquivalenzklassenbildung versucht die Komplexität eines Testproblems durch fortgesetztes Zerlegen so weit zu reduzieren, dass schließlich eine sehr einfache Wahl von Testfällen möglich wird.

- Am Ende der Zerlegung steht eine Menge von so genannten Äquivalenzklassen.

- Alle Werte einer Äquivalenzklasse sollen durch die zu testende Software gleichartig bearbeitet werden.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 58

FUNKTIONALE ÄQUIVALENZKLASSENBILDUNG (BLACK BOX)

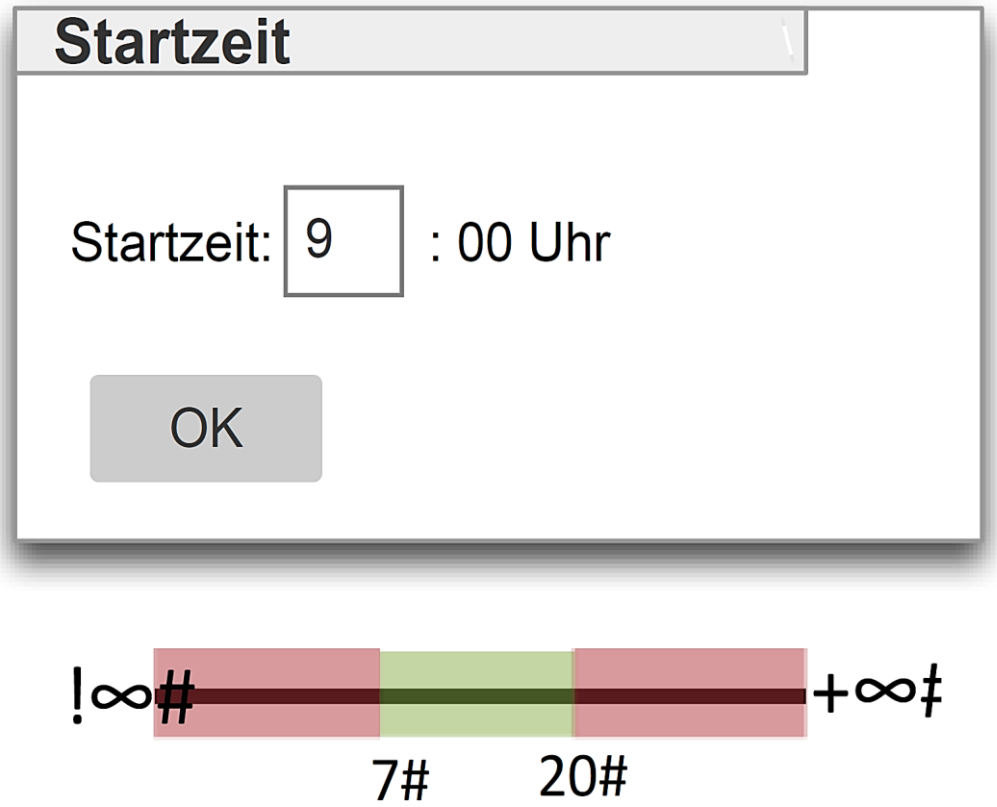
- Eingabewerte der Funktionen werden in Bereiche, die gleiches, d.h. fachlich identisches Systemverhalten, Verhalten hervorrufen sollen, eingeteilt.

- Die funktionale Äquivalenzklassenbildung ist gut geeignet für den Test von Software, die nicht zustandsbasiert ist und bei der die Reaktionen nicht abhängig sind von komplizierten Eingabeverknüpfungen.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 58

ÄQUIVALENZKLASSEN - BEISPIEL

- Gültige Äquivalenzklasse B1: Alle Zahlen zwischen 7 und 20: {x |7 <= x <=20 und x ist ganze Zahl}
- Ungültige Äquivalenzklasse B2: Alle Zahlen kleiner als 7: {x |x < 7}
- Ungültige Äquivalenzklasse B3: Alle Zahlen größer als 20: {x | 20 > x}



Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

GRENZWERTANALYSE

- Die der Grenzwertanalyse zu Grunde liegend Annahme ist die Beobachtung, dass Softwarefehler häufig in dem Grenzbereich liegen, der den Übergang von gültigen zu ungültigen Eingabewerten darstellt.

- **Die Fehler liegen an den Grenzen von Äquivalenzklassen.**

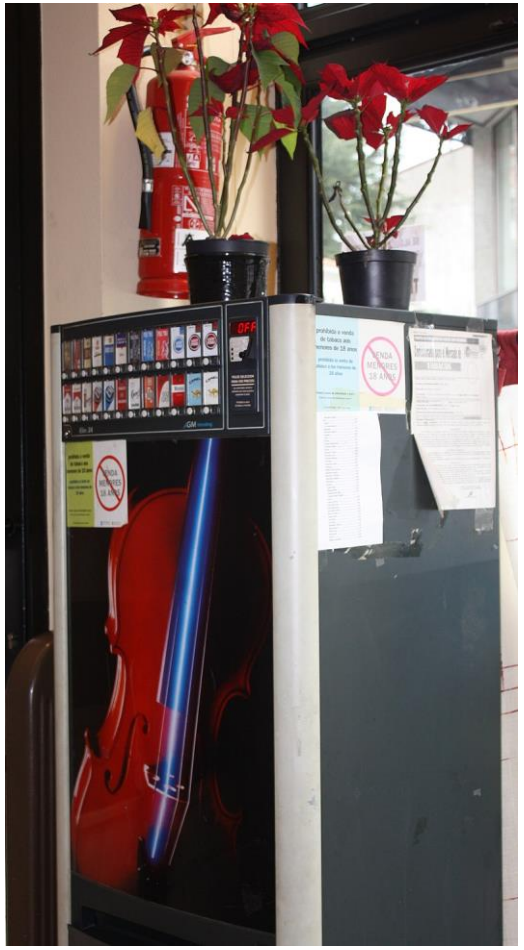
GRENZWERTANALYSE – BEISPIEL ZIGARETTENAUTOMAT

Gruppenarbeit:

Wie kann die Äkquivalenzklasse für einen Zigarettenautomaten aussehen?

15 Minuten, Gruppenarbeit

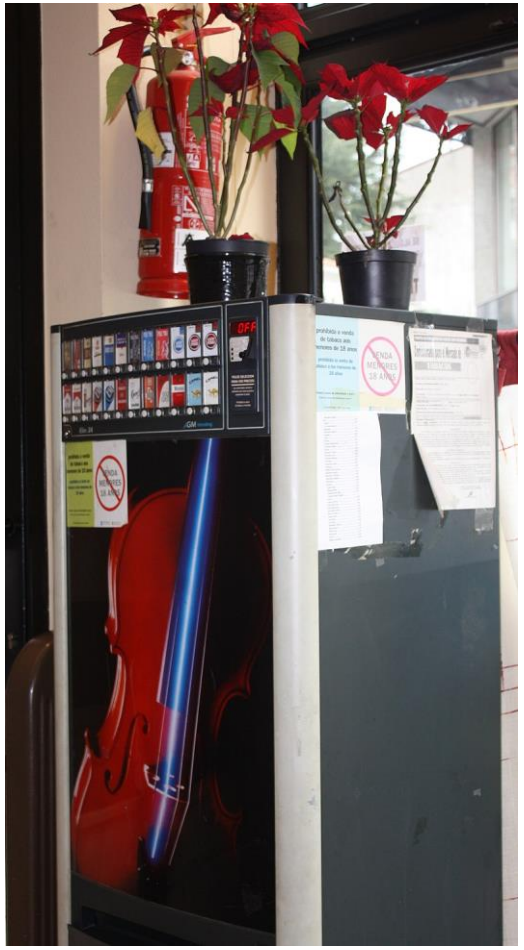
Diskurs-Bonus-Frage: Habt ihr in euren Software Engineering Projekten auch Bereich, wo Äkquivalenzklassen möglich sind?



ERGEBNISSE

GRENZWERTANALYSE – LÖSUNG ZIGARETTENAUTOMAT

1. Ausgabe von Zigaretten erst ab 18 Jahren
2. Gültige Äquivalenzklasse: >= 18
3. Ungültige Äquivalenzklasse: < 18 (enthält somit auch Werte wie 0 und -56)
4. Grenzwert: 18, (aber auch der Übergang von positivem auf negativen Wert, 0, -1, 1)



VORGEHEN BEI DER TESTFALLERSTELLUNG

1. Äquivalenzklassen für jeden Eingabeparameter der Funktion bzw. jedes GUI-Element der Dialogmaske bestimmen und eindeutig kennzeichnen.

2. Testfälle erstellen, mit denen alle gültigen Äquivalenzklassen abgedeckt werden. Die Anzahl der erforderlichen Testfälle ist dabei so gering wie möglich zu halten.

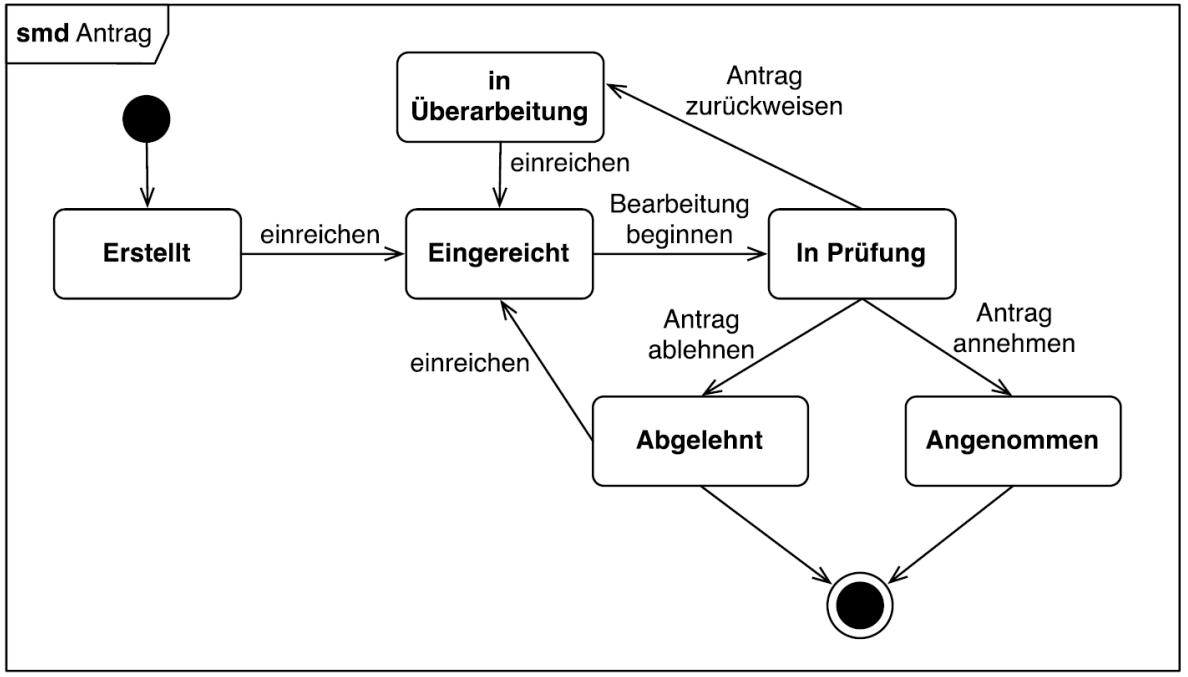
VORGEHEN BEI DER TESTFALLERSTELLUNG

3. Testfälle erstellen, mit denen alle ungültigen Äquivalenzklassen abgedeckt werden. Im Unterschied zu den Testfällen für gültige Äquivalenzklassen werden bei den Testfällen für ungültige Äquivalenzklassen pro Testfall nur eine ungültige Äquivalenzklasse getestet.

4. Erstellung von konkreten Eingabedaten auf Basis der ausgewählten Äquivalenzklassen für jeden Testfall.

ZUSTANDSBASIERTES TESTEN

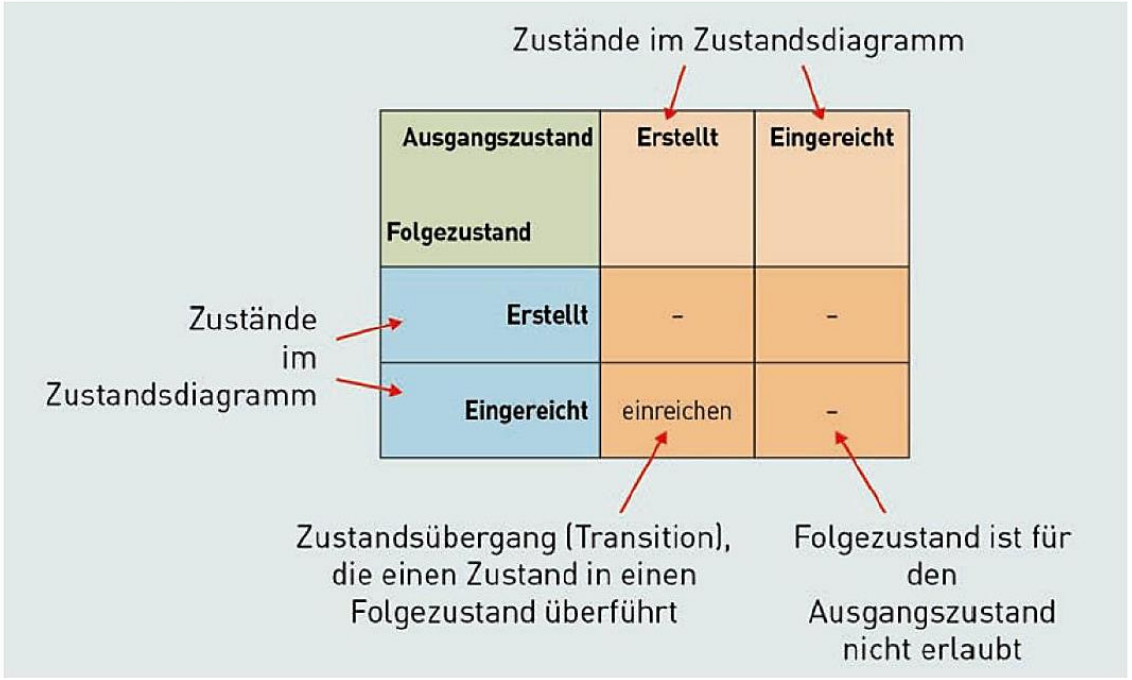
- Zustand = innerer oder äußerer Zustand eines Softwareobjekts
- Eingesetzt bei Softwaresystemen, bei denen die Zustände wichtig sind, oder deren Spezifikation zustandsbasiert (z.B. durch Zustandsdiagrammen) ist.
- WhiteBox oder BlackBox möglich



Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

ZUSTANDSBASIERTES TESTEN

1. Ableiten der Zustandstabelle
2. Ausfüllen der Zustandstabelle mit möglichen Übergängen
3. Ausfüllen der Zustandstabelle mit ungültigen Übergängen



Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

ZUSTANDSBASIERTES TESTEN

1. Ableiten der Zustandstabelle
2. Ausfüllen der Zustandstabelle mit möglichen Übergängen
3. Ausfüllen der Zustandstabelle mit ungültigen Übergängen

Ausgangszustand / Folgezustand	Erstellt	Eingereicht	In Prüfung	In Überarbeitung	Angenommen	Abgelehnt
Erstellt	–	–	–	–	–	–
Eingereicht	einreichen	–	–	einreichen	–	einreichen
In Prüfung	–	Bearbeitung beginnen	–	–	–	–
In Überarbeitung	–	–	Antrag zurückweisen	–	–	–
Angenommen	–	–	Antrag annehmen		–	
Abgelehnt	–	–	Antrag ablehnen	–	–	–

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

ZUSTANDSBASIERTES TESTEN VOR- UND NACHTEILE

Vorteile	Nachteile
Leichte Anwendbarkeit bei vorhandenen Zustandsdiagrammen	Bei vielen Zuständen unübersichtlich
Gute Visualisierbarkeit der Testabdeckung	Leichtes Vergessen von Funktionen, die Zustände nicht ändern
Auch für GUI Systeme einsetzbar	

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

ZUFALLSTESTDATEN

- Im Rahmen eines gegebenen Wertebereichs von Eingabeparametern werden gültige Werte zufällig erzeugt.

- Achtung: zwar viele verschiedene Daten werden erzeugt, jedoch nicht in jedem Fall wirklich realistische Daten.

- Achtung, das Testzielbetrachten, sollen z.B. die Datenfelder in ihrer Funktionalität geprüft werden müssen Äquivalenzklassen eventuell mitbetrachtet werden.

ZUFALLSTEST

- Der Zufallstest (random test) ist eine Testtechnik, die aus den Wertebereichen der Eingabedaten zufällig Testfälle erzeugt.

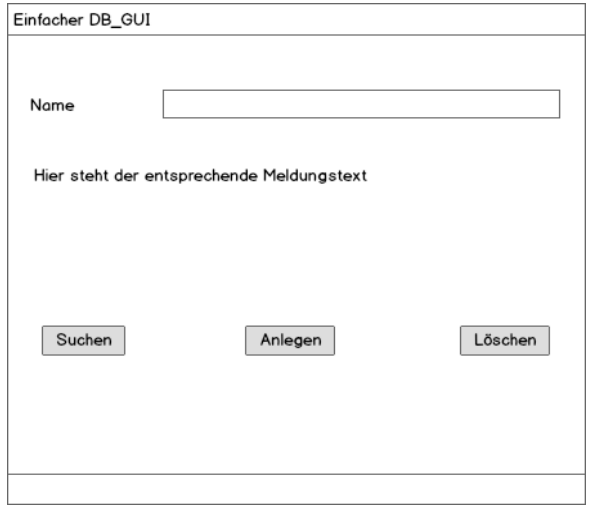
- Im Unterschied zu den anderen dargestellten Testverfahren, liegt dem Zufallstest keine deterministische Strategie zugrunde. Der Zufallstest ist eine stochastische Testtechnik.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 208.

FALLSTUDIE TESTFÄLLE (GRUPPENARBEIT – 40 MIN.)

Folgende **Funktionen** und Eigenschaften hat eine GUI:

- Der Meldungstext und das Namensfeld ist zunächst leer (bei jedem Neustart der Software ist dies der Fall)
- Der Button ‚löschen' löscht eventuelle Eingaben aus dem Feld und setzt die Buttons wieder auf eingabebereit
- Ist kein Text in der Eingabebox sind die Buttons alle disabled
- Der Meldungstext verschwindet sobald Text in das Textfeld eingegeben wird
- Der Meldungstext kann folgende Wert annehmen:
  - Der Name <Inhalt des Textfeldes> kann nicht gefunden werden
  - Der Name <Inhalt des Textfeldes> wurde gefunden
  - Der Name <Inhalt des Textfeldes> war bereits vorhanden, konnte nicht angelegt werden
  - Der Name <Inhalt des Textfeldes> wurde angelegt
  - Der Name <Inhalt des Textfeldes> wurde gelöscht
  - Der Name <Inhalt des Textfeldes> konnte nicht gelöscht werden, er wurde nicht gefunden
- Die Meldung erscheint nach dem Drücken des entsprechenden Buttons



Wie können passende Testfälle aussehen?

Welche Verfahren sind geeignet?

ERGEBNISSE

ZUSTANDSBASIERTES TESTEN – KONTEXT ABHÄNGIG

Anfangszustand / Folgezustand	Leer	Name eingege-ben	Name nicht gefunden	Name gefunden	Name bereits vorhanden	Name wurde angelegt	Name wurde gelöscht	Name wurde nicht gelöscht
Leer	-	-	-	-	-	-	E: Löschen	E: Löschen
Name eingegeben	Eingabe	-	-	-	E: Anlegen / Löschen	E: Anlegen	-	-
Name nicht gefunden	-	Eingabe	-	-	-	-	-	-
Name gefunden	-	Eingabe	-	-	-	-	-	-
Name bereits vorhanden	-	-	-	E: Anlegen	-	-	-	-
Name wurde angelegt	-	-	E: Anlegen	-	-	-	-	-
Name wurde gelöscht	-	-	-	E: Löschen	-	-	-	-
Name wurde nicht gelöscht	-	-	E: Löschen	-	-	-	-	-

ZUSTANDSBASIERTES TESTEN – KONTEXT ABHÄNGIG

Anfangszustand / Folgezustand	Leer	Name eingege-ben	Name nicht gefunden	Name gefunden	Name bereits vorhanden	Name wurde angelegt	Name wurde gelöscht	Name wurde nicht gelöscht
Leer	-	-	-	-	-	-	E: Löschen	E: Löschen
Name eingegeben	Eingabe	-	-	-	E: Anlegen / Löschen	E: Anlegen	-	-
Name nicht gefunden	-	Eingabe	-	-	-	-	-	-
Name gefunden	-	Eingabe	-	-	-	-	-	-
Name bereits vorhanden	-	-	-	E: Anlegen	-	-	-	-
Name wurde angelegt	-	-	E: Anlegen	-	-	-	-	-
Name wurde gelöscht	-	-	-	E: Löschen	-	-	-	-
Name wurde nicht gelöscht	-	-	E: Löschen	-				

Welche Fehlverhalten können wir erkennen?

FEEDBACK REFERATE

- Feedback aus den Gruppen (10 Min. Vorbereitung):

- Reflexionsfragen:
  - Welche Fragen zum Leistungsnachweis sind offen?
  - Welche Methoden wollt ihr einsetzen?
  - Wie ist der Stand zum Qualitätsplan?

06

INTERNATIONALE HOCHSCHULE

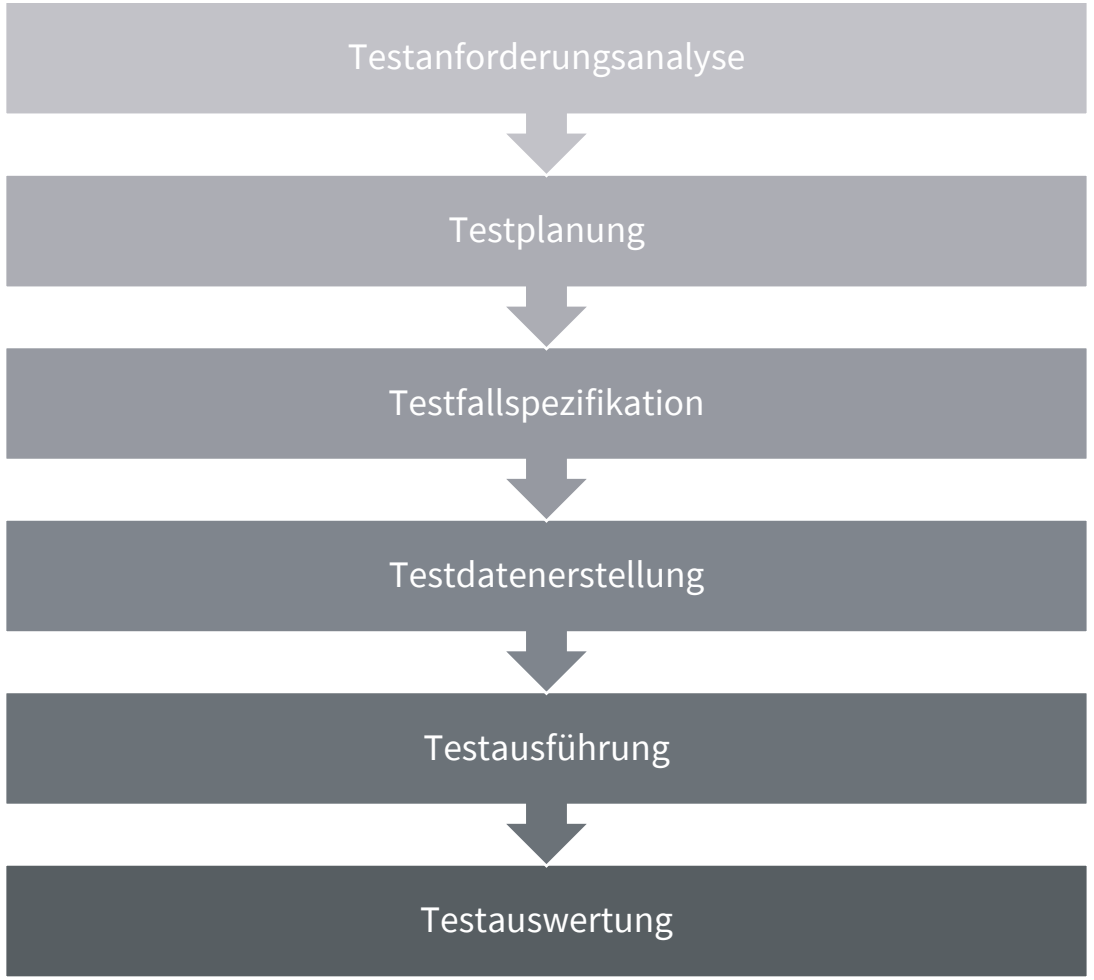
SYSTEMATISCHES TESTEN VON SOFTWARE

LERNZIELE

1. Aktivitäten zum methodischen Testen
2. Komponententests, inkl. automatische Unit Tests und Test Driven Development
3. Integrationstests, Integrationsstrategien
4. System- und Abnahmetests

AKTIVITÄTEN ZUM METHODISCHEN TESTEN

- Aktivitäten sind in allen Teststufen vorhanden
- Anpassungen je Teststufe bei Inhalt und Ausgestaltung



Quelle

DIE TESTPLANUNG: WELCHE FRAGEN MÜSSEN GEKLÄRT WERDEN

Der Test	**Was?**	Welche Objekte und Funktionen sind zu testen?
	Warum?	Welche Ziele verfolgt der Test?
	Wann?	Welche Termine sind einzuhalten?
	Wo?	Wo soll getestet werden?
	Wie?	Unter welchen Bedingungen soll getestet werden?
	Womit?	Mit welche Mitteln und Werkzeugen soll getestet werden?
	Wer?	Welche Mitarbeiter führen den Test durch?

REMINDER: TESTFALLSPEZIFIKATION: GRUNDGRÖßE TESTFALL

Handlungsanweisung zur Durchführung von Softwaretests

besteht (wenigstens) aus folgenden Elementen

Beschreiben Daten zum Testfall (z.B. Name, ID, …)	Vorbedingun-gen	Testaktionen	Testdaten	Nachbedin-gungen

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

REMINDER: BEISPIEL TESTFALLDEFINITION 1

Template für konkrete Testfälle	
ID des Testfalls	Eindeutige ID
Testgegenstand	Offizieller Name der zu testenden Systemkomponente oder der Schnittstelle oder der Funktion des Systems/der Komponente
Getestetes Detail	Name der Klasse(n) oder Komponente(n), die bei diesem Test getestet werden
Methode(n), die getestet werden	Detaillierter Name der Methode/der Funktionen/der Schnittstelle und deren Parameterliste
Kurzbeschreibung des Tests	Kurze Beschreibung, was mit dem Test geprüft bzw. sichergestellt werden soll
Testdaten/Eingabe	Welche Eingabedaten werden benötigt, um den Testfall durchzuführen? Mit welchen Parametern sollen die zu testenden Funktionen aufgerufen werden? Falls es sich um Objekte handelt: Welche Attribute sollen sie enthalten? Falls es sich um Dateien handelt: Welchen Inhalt haben sie?
Erwartetes Ergebnis/ Erwartete Systemreaktion	Welche Ausgaben oder Systemreaktionen werden zu den Testdaten erwartet?
Vorbereitung/Vorbedingung	Konkrete Schritte, um das System/die Komponente für den Test vorzubereiten: • Erzeugen oder Einspielen von Testdaten in die Datenbank • Aufrufen bestimmter Methoden, damit das System in den Zustand gelangt, in dem getestet werden kann (Bsp.: Anmelden von Nutzern am System) • Sicherstellen, dass bestimmte Daten in der Datenbank vorliegen

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

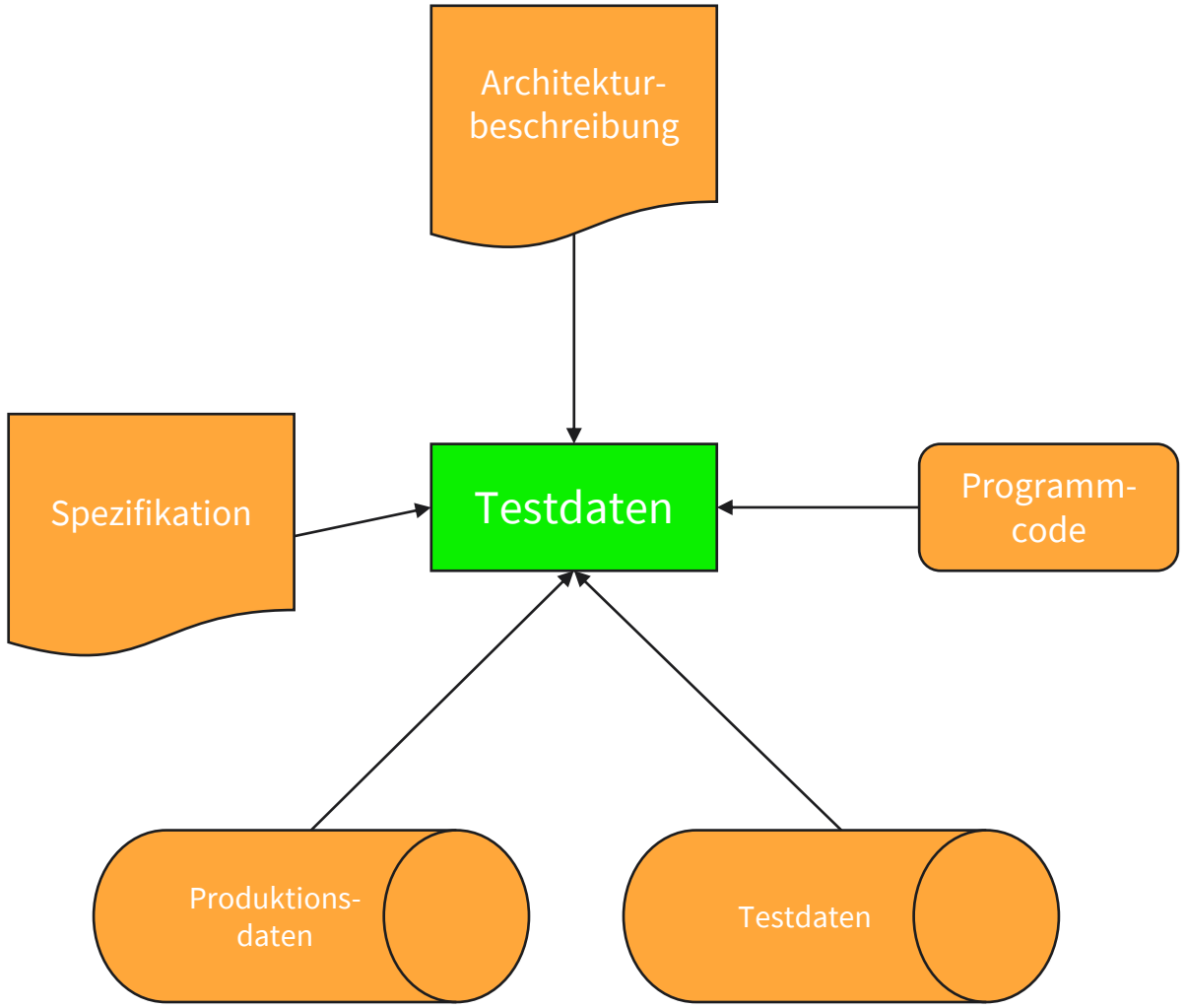
REMINDER: BEISPIEL TESTFALLDEFINITION 2

Template für konkrete Testfälle	
ID des Testfalls	Eindeutige ID
Durchführung	Welche Systemfunktion(en)/Methode(n) muss/müssen beim Test ausgeführt werden? Bei komplexeren Testfällen mit mehreren Aufrufen: In welcher Reihenfolge müssen die Methoden/System-funktionen ausgeführt werden?
Überprüfung/Nachbe-dingung	Geben Sie hier die konkreten Bedingungen eines bestandenen Tests an. Welches Systemverhalten/welche Ausgaben werden vom System erwartet? Welche Kriterien müssen die im System gespeicherten Daten erfüllen? Bei der Erfüllung welcher Kriterien ist der Test erfolg-reich? Wie werden die Kriterien gemessen?

Schiffmann, J. (2022). Qualitätssicherung im Softwareprozess – Dynamische Qualitätssicherung. IU.

TESTDATENERSTELLUNG

- Synthetische oder reale Testdaten
- Erstellung
- Management des Testdatenbestands



ERSTELLUNG VON (ZUFALLS-)TESTDATEN

Gruppenarbeit (40 Min.):

Erklärt an einem Beispiel eurer Wahl:

- Was sind Charakteristika beim arbeiten mit Testfällen bei eurem Projekt → Welche Ausprägungen (Funktionen und Situationen) kann bei eurer Softwarelösung getestet werden?

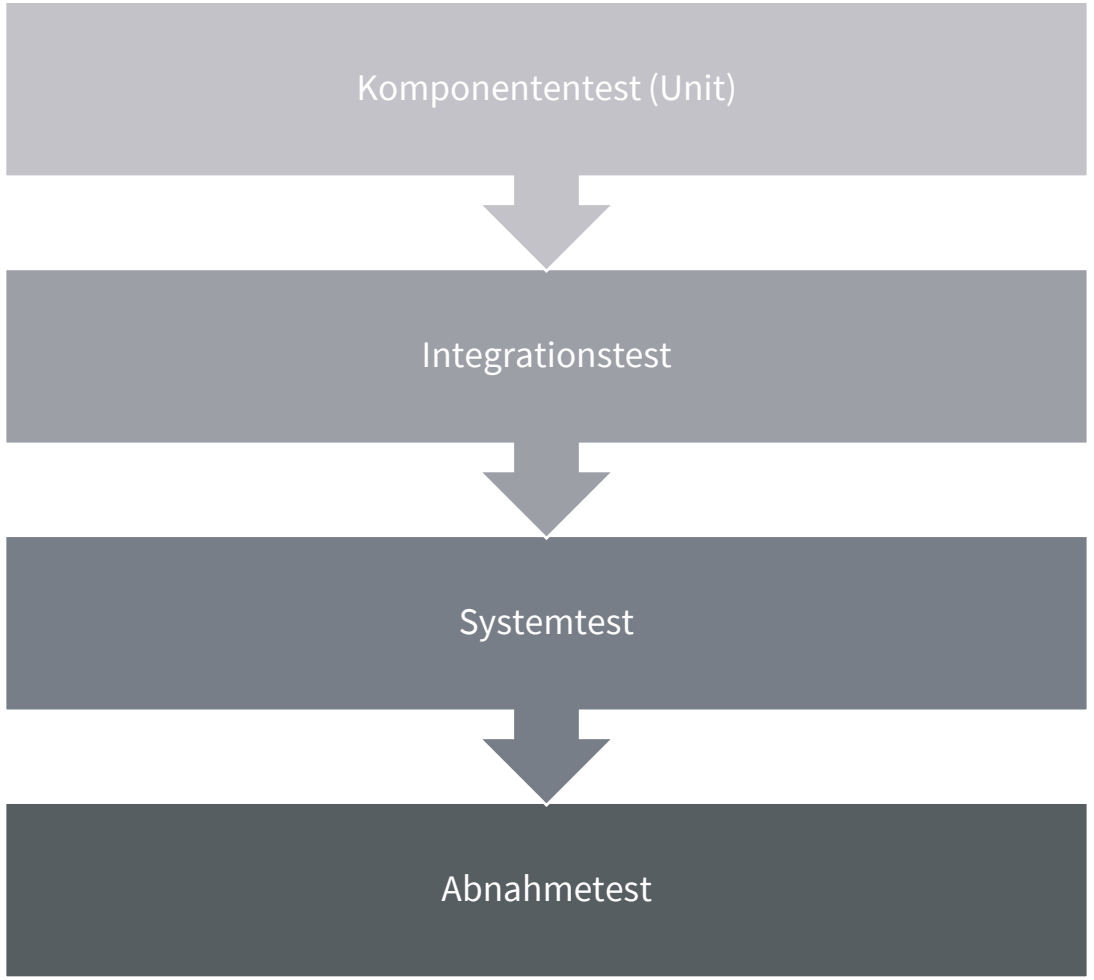
- Bei Systemen mit Datenverwaltung: Aus welchen Quellen können Testdaten für den Testfall stammen? Wie können diese für euer Beispiel erstellt werden? (grobe Codebeispiele → ggf. Spezifikation für ChatGPT)

ERGEBNISSE

TESTSTUFEN | QUALITÄTSSICHERUNG VON SOFTWARE IN MEHREREN TESTSTUFEN

- Testen erfolgt (soll erfolgen) entwicklungsbegleitend
- Teststufen orientieren sich an Entwicklungsstufen

- Welche Eigenschaften getestet werden hängt **auch** von den Teststufen ab



KOMPONENTENTEST

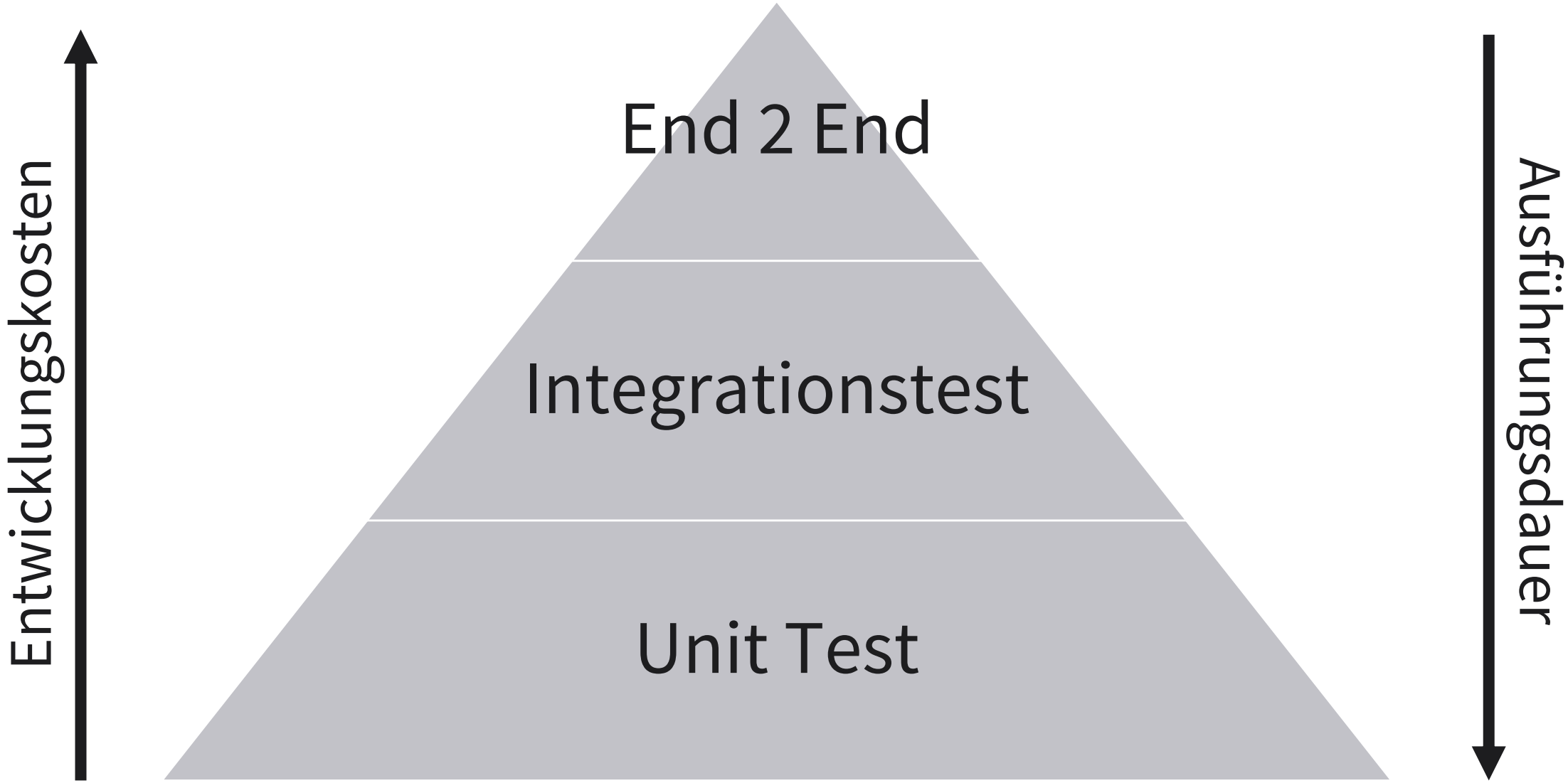
Eigenschaften:

- Reproduzierbarkeit
- Unabhängigkeit
- Regressionsfähigkeit

Scope: logische Komponente

- Komponenten werden einzeln getestet (vor Integration)
- Hoher Automatisierungsgrad (Frameworks für gängige Programmiersprachen)
- Whitebox-Test

BERÜCKSICHTIGUNG VON TESTKOSTEN



Lenger, M. (o. D.). Was ist ein guter Unit-Test? – Und wie entwickelt man ihn? Informatik Aktuell. https://www.informatik-aktuell.de/entwicklung/methoden/was-ist-ein-guter-unit-test-und-wie-entwickelt-man-ihn.html

WAS IST EIN UNITTEST?

Ein einem Unit-Test wird eine Einheit im Programm durch einen Test abgedeckt: Wie groß diese Einheit ist, ist nicht genau definiert:

- Im klassischen Sinne ist dieses eine Methode, ein Modul oder eine Klasse (für jede Klasse wird eine Testklasse geschrieben).

- Es können auch mehrere Methoden oder ein größeres Modul sein → es sollte ein abgrenzbarer Teil eines Programms sein, welcher eigenständig Daten verarbeitet.

WAS BEINHALTET EIN UNIT TEST?

- Es werden Testfälle mit Akzeptanzkriterien beschrieben.
  - Es kann aber auch auf Erfüllung oder auf Fehler (Exception) getestet werden: Eine Exception wird erwartet, aber nicht ausgeliefert --> Der Test ist fehlgeschlagen

- Bei Testfällen sollte man die verschiedenen Abläufe und Zustände in einem Programm (oder auch in einer Unit) berücksichtigen und diese Testen.

WELCHE ABLÄUFE KÖNNEN HIER EINTRETEN?

Beispielprogramm:

```
while(B) {
  A
  }
C
```

WELCHE ABLÄUFE KÖNNEN HIER EINTRETEN?

INTERNATIONALE HOCHSCHULE

Beispielprogramm:

```
while(B) {
  A
  }
C
```

C = Abbruch

AC = Die Schleife wird einmal durchlaufen

AAC = Die Schleife wird mehrfach durchlaufen

DER UNTERSCHIED ZWISCHEN UNIT- UND INTEGRATIONSTESTS

- Ein Unit-Test überprüft einzelne Komponente oder Einheit einer Anwendung isoliert von anderen Komponenten.
- Ein Integrationstest überprüft, wie verschiedene Komponenten oder Einheiten einer Softwareanwendung zusammenarbeiten.
- Integrationstests werden nach den Unit-Tests durchgeführt
- Unit-Tests werden von Entwicklern geschrieben und sind in der Regel schnell auszuführen.

DISKUSSION ZUM START VON TEST DRIVEN DEVELOPMENT

- Warum kann Testautomatisierung bei der Softwareentwicklung wichtig sein?
  - Überlegt auch in diesem Kontext:
    - Welche Systeme zur Testautomatisierung kennt und nutzt ihr?
    - Programmiert ihr defensiv (Arbeiten mit Tests – Tests first) oder offensiv (Tests second oder gar nicht?)?

TEST DRIVEN DEVELOPMENT (TDD)

- Einsatz in der agilen Softwareentwicklung:
  - „With incremental development, there is no system specification that can be used by an external testing team to develop system tests.“

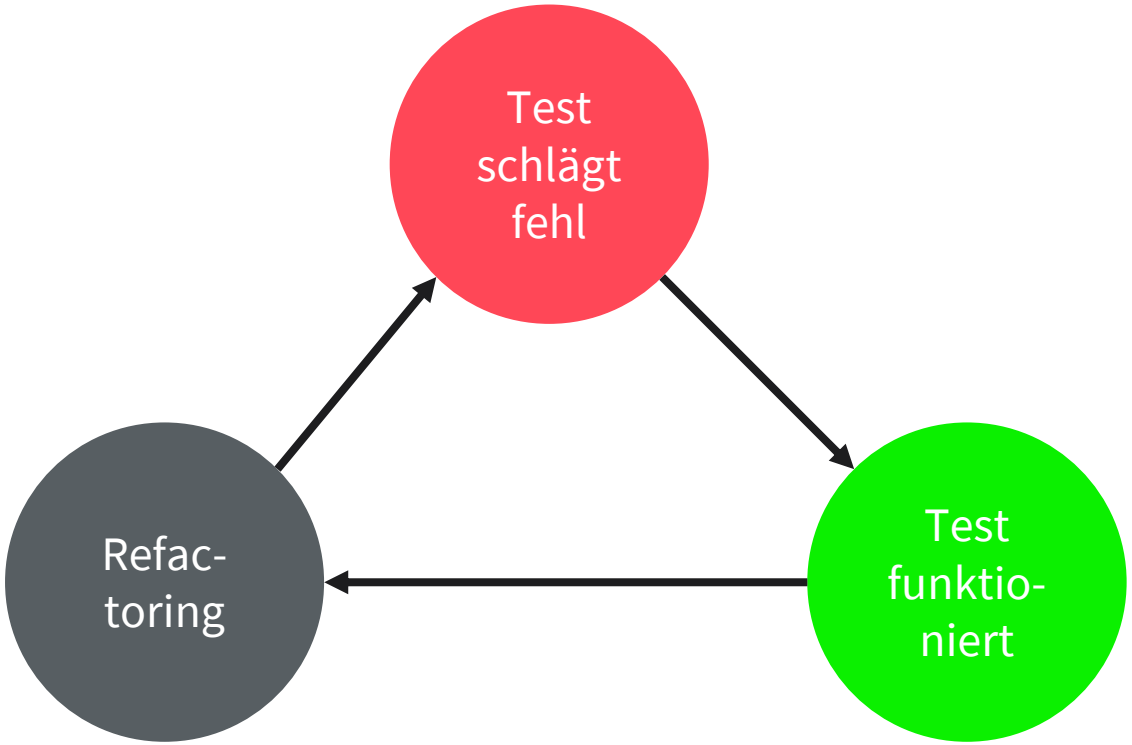
- TDD lenkt den Fokus früh auf ggf. notwendige Interfaces (Schnittstellen), was Missverständnisse reduziert (Wann arbeite ich Wo mit Daten? Wann werden diese Wo übergeben?).

- Der fachliche Input des Kunden fließt in die Entwicklung des Tests ein, sodass dieser zur Spezifikation der Anforderung wird.

Sommerville, Ian (2016): Software Engineering. Tenth Edition. Boston, Columbus, Indianapolis u. a.: PEARSON. Harlow: Pearson Education Limited. 2016. S. 81-82.

TEST DRIVEN DEVELOPMENT (TDD)

- Zuerst werden anhand der Anforderung (mehrere und passende) Testfälle entwickelt: Die Implementation erfolgt so lang, bis die Testfälle durch den Code erfüllt werden



TEST DRIVEN DEVELOPMENT (TDD)

- TDD erfordert den Einsatz automatisierter Testframeworks, z. B. JUnit (Java) oder NUnit (.net). Diese Frameworks bieten Funktionen wie Assertions und Testläufer.

- Assertions sind Anweisungen, die in einem Unit-Test verwendet werden, um zu überprüfen, ob das Ergebnis einer Funktion dem erwarteten Ergebnis entspricht.

- Ein Beispiel für eine Assertion in Java ist:

assertEquals(erwartetesErgebnis, tatsächlichesErgebnis);

Sommerville, Ian (2016): Software Engineering. Tenth Edition. Boston, Columbus, Indianapolis u. a.: PEARSON. Harlow: Pearson Education Limited. 2016. S. 81-82.

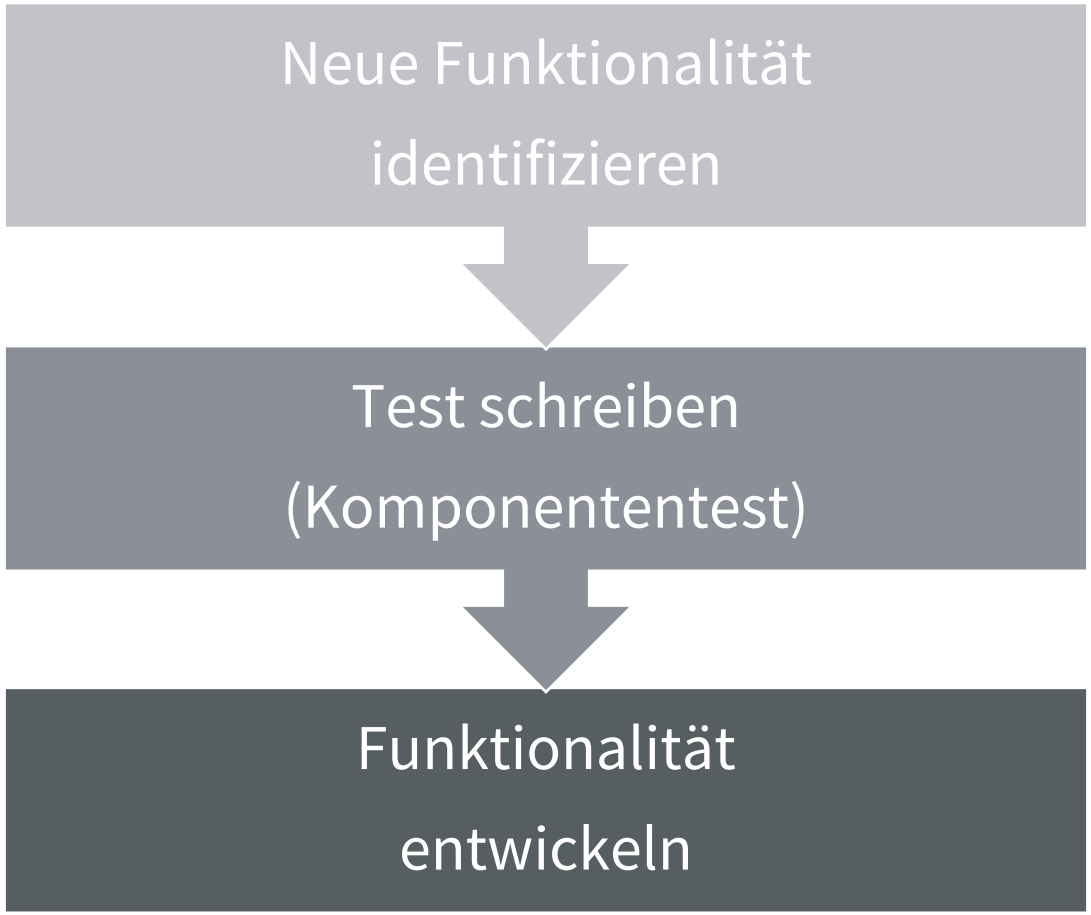
TEST DRIVEN DEVELOPMENT (TDD)

- **Testläufer** sind Programme, die die Tests ausführen, indem sie die Assertions ausführen und das Ergebnis auswerten.
  - Sie können mehrere Tests gleichzeitig ausführen und die Ergebnisse analysieren, um zu sehen, ob die Tests erfolgreich waren oder nicht.

- Tests vor jedem Commit beugt Problemen vor dem Release vor

Sommerville, Ian (2016): Software Engineering. Tenth Edition. Boston, Columbus, Indianapolis u. a.: PEARSON. Harlow: Pearson Education Limited. 2016. S. 81-82.

TEST DRIVEN DEVELOPMENT (TDD)



BEISPIEL TEST JUNIT

```
import org.junit.jupiter.api.Test;
import static
org.junit.jupiter.api.Assertions.assertEquals;

public class CalculatorTest {
  @Test
  public void testAddition() {
    Calculator calculator = new Calculator();
    int result = calculator.add(2, 3);
    assertEquals(5, result);
  }
}
```

@Test ist ein Annotationstyp, der in der JUnit-Bibliothek verwendet wird, um eine Methode als Testmethode zu markieren.

In unserem Fall wird die Methode testAddition() als Testmethode markiert, um zu überprüfen, ob die Methode add() des Calculators korrekt funktioniert.

HERAUSFORDERUNGEN DES TEST DRIVEN DEVELOPMENT (TDD)

- Teilweise wird von der Regel abgewichen, zuerst den Testfall zu entwickeln
- Mitunter sind Tests ineffizient, da sie nicht die volle Funktionalität testen
- Das User-Interface lässt sich schwer testen (Sommerville, S. 83)

Sommerville, Ian (2016): Software Engineering. Tenth Edition. Boston, Columbus, Indianapolis u. a.: PEARSON. Harlow: Pearson Education Limited. 2016. S. 81-82.

ÜBUNG ZU TEST-DRIVEN-DEVELOPMENT

Das Projekt IT-Kaffeebohne steht kurz vor dem Abschluss. Ihr müsst kurzfristig als Entwickler einspringen. Die Anforderung, die umzusetzen ist, lautet wie folgt:

> Entwickelt eine Funktion um für eine gegebene Temperatur zu entscheiden, ob der Kaffeevollautomat auf den Becher einen Warnhinweis „Vorsicht heiß“ druckt. Der Warnhinweis soll ausgegeben werden, wenn die Temperatur 50°C überschreitet.

Die Aufgabe:

- Wie kann ein Softwaretest für dieses Beispiel aussehen? (ChatGPT ist zugelassen)
- 15 Minuten in den Gruppen (versucht verschiedene Programmiersprachen)

BEISPIELLÖSUNGEN

BEISPIELLÖSUNG TEST-DRIVEN-DEVELOPMENT

```
public static void hotCoffeeWarning(double temperature) {
   if (temperature > 50) {
      System.out.println("Vorsicht heiß!");
   }
}
import org.junit.Test;
public class HotCoffeeWarningTest {
   @Test
   public void testHotCoffeeWarning() {
      double temperature = 60;
      HotCoffeeWarning.hotCoffeeWarning(temperature);
      assertEquals("Vorsicht heiß!", outContent.toString());
   }
}
```

INTEGRATIONSTEST

Scope: Zusammenspiel der Komponenten

- Sobald zwei Komponenten fertig gestellt sind, können diese in einen Integrationstest treten
- Treiber und Dummies als Testmöglichkeiten
- Integrationsstrategien

Integrationsstrategien

- Bottom-Up
- Top-Down
- By Value

INTEGRATIONSTEST: TREIBER UND DUMMIES

- Als Treiber werden dabei die Softwarefragmente bezeichnet, die das Aufrufen anderer Komponenten simulieren.
- Dummies hingegen simulieren Komponenten, die durch andere Komponenten aufgerufen werden.

Ziel des Systemtests ist die Überprüfung, ob das System als ganzes die spezifizierten Anforderungen erfüllt

Herausforderungen:

- möglichst originalgetreue Nachbildung der Produktivumgebung des Kunden
- Bereitstellung von originalgetreuen Datensätzen
- Realistisch Auslastung und Nutzerverhalten

TESTARTEN SIND MANNIGFALTIG

Düsterbeck, F. 2014, Testen mit, durch und in Scrum, abgerufen am 25.03.2023 von https://de.slideshare.net/fduesterbeck/frank-dsterbeck-testen-mit-durch-und-in-scrum-asqf

TESTARTEN IM SYSTEMTEST

Testarten unterscheiden sich nach

- zu prüfendem Qualitätsmerkmal
- Testzielen
- Testobjekt (z.B. System oder Dokumentation)

TESTARTEN IM SYSTEMTEST

Mögliche Testarten im Systemtest:

- funktionaler Systemtest
- Performancetest
- Lasttest
- Usability-Test
- Wiederinbetriebnahmetest

SYSTEMTEST: LAST- UND PERFORMANCETEST

Lasttest (auch Stresstest): Das Ziel ist explizit das Testen des Systemverhaltens unter besonders hohen Anforderungen an die zur Verfügung stehenden Ressourcen.

Performancetest:

Kontext	Definition	Beispiel
Latenz	Zeitspanne, die von der Anfrage an das System bis zur Beantwortung vergeht.	Die Zeit, die das System bei der Bestellung in einem Onlineshop vom Klick auf den OK-Button bis zur Anzeige des Bestätigungsdialogs benötigt.
Durchsatz	Verarbeitungsgeschwindigkeit von Funktionen oder Daten.	Die Anzahl der verarbeiteten Bestellvorgänge pro Minute.
Transaktionsrate	Maß zur Verarbeitungsgeschwindigkeit von definierten, komplexen Änderungen am Datenbestand.	Die Anzahl der gebuchten Überweisungen im Kernsystem einer Direktbank.

SHORTDISCUSSION

INTERNATIONALE HOCHSCHULE

Blitzlicht:

Warum ist es nicht sinnvoll Testes direkt miteinander zu kombinieren?

Wieso kann man keinen Lasttest während des Integrationstest durchführen?

REGRESSIONSTESTS

Regressionstests = Testfälle, die bereits erfolgreich durchgeführt worden sind und nach einer Anpassung des Systems erneut durchgeführt werden.

Die Eigenschaft Regressionsfähigkeit von Testfällen bezeichnet daher die Fähigkeit zur wiederholten Durchführung des Tests.

ZIEL DES REGRESSIONSTEST

- „Ziel des Regressionstests ist nachzuweisen, dass Modifikationen von Software keine unerwünschten Auswirkungen auf die Funktionalität besitzen.

- Durch Software-Modifikationen, z. B. durch Funktionalitätserweiterungen oder Fehlerkorrekturen können neue Fehler in zuvor fehlerfreie Teile eingefügt werden.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 192.

ZIEL DES REGRESSIONSTEST

- „Der Regressionstest zielt auf die Erkennung derartiger Seiteneffekte.
- Weil nicht davon ausgegangen werden kann, dass korrekt abgearbeitete Testfälle auch nach Änderungen der Software weiterhin korrekt abgearbeitet werden, reicht die einmalige Durchführung von Testfällen nicht aus.
- Streng genommen, müssten alle bisherigen Testfälle nach Modifikationen wiederholt werden.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 192.

ZIEL DES REGRESSIONSTEST

- Ein Regressionstest besteht aus der Wiederholung von bereits durchgeführten Testläufen:
  - Die zu testende Software ist pro durchzuführendem Testfall in den gleichen Zustand zu versetzen wie bei der vorhergehenden Durchführung dieses Testfalls.
  - Es sind identische Eingaben vorzunehmen.
  - Die neu erzeugten Ausgaben sind mit den Ausgaben der Vorläuferversion zu vergleichen.
  - Falls keine Unterschiede auftreten, ist der Regressionstestfall erfolgreich absolviert.
  - Falls Unterschiede erkannt werden, so ist zu prüfen, ob diese gewünscht sind, oder ob sie fehlerhafterweise aufgetreten sind.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 192f.

REGRESSIONSTESTS UND TESTAUTOMATISIERUNG

- „Der so genannte Regressionstest prüft das Verhalten einer aktuellen Software Version gegen das Verhalten der Vorläuferversion. Der Regressionstest besitzt eine hohe praktische Bedeutung. “

- „Regressionstests sind möglichst zu automatisieren und zwar einschließlich des Soll-/Ist-Ergebnisvergleichs.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 179, 436.

GROUPWORK

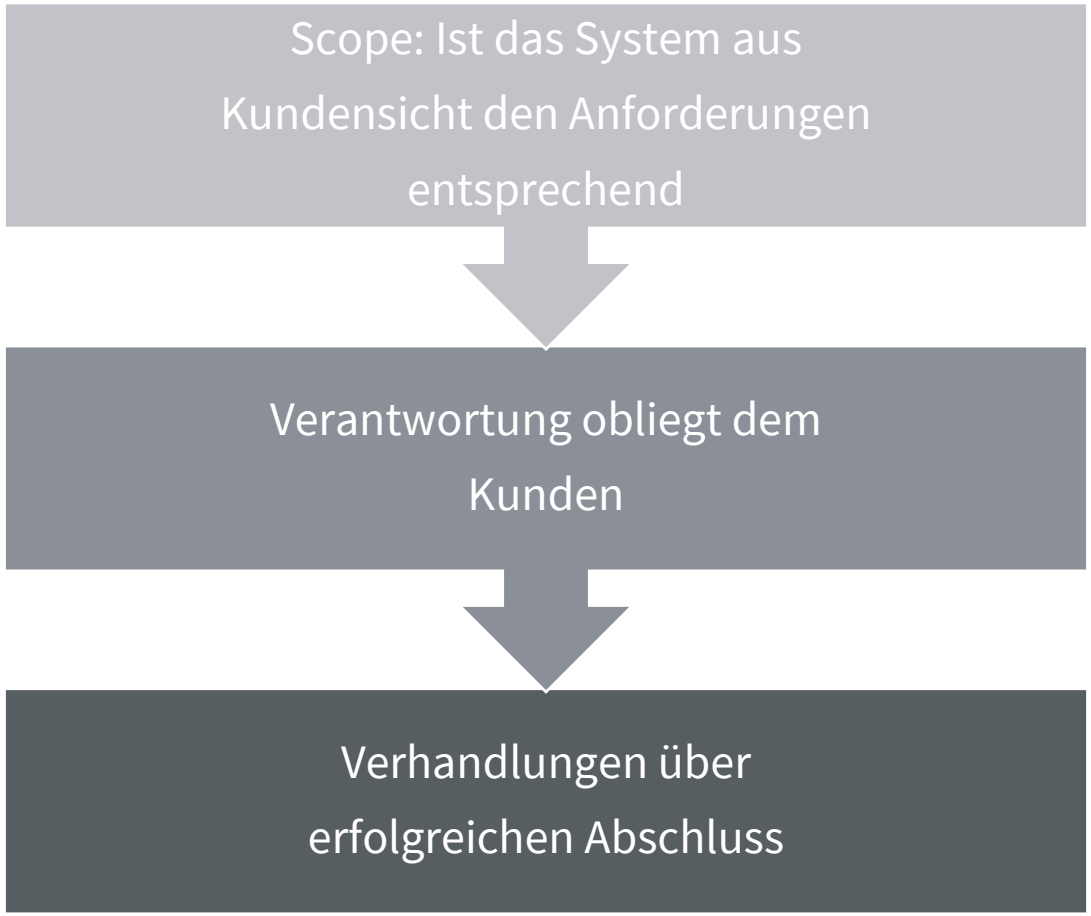
Gruppenarbeit, 20 Minuten:

Wählt ein praxisnahes Beispiel aus:

- Zeigt an diesem Beispiel den Unterschied zwischen einem Last- und Performancetest
- Zeigt an eurem Beispiel Möglichkeiten für standardisierte (Unit) Tests und einen Regressionstest

ERGEBNISSE

ABNAHMETESTS



Kurzes Reminder-Blitzlicht:

Warum sollte es eine Trennung von Fehlersuche und Fehlerbehebung geben?

Warum kann dieses Vorgehen wichtig und sinnvoll sein?

07

INTERNATIONALE HOCHSCHULE

SYSTEMATISCHE QUALITÄTSSICHERUNG VON ANFORDERUNGEN, ARCHITEKTUREN UND PROZESSEN

LERNZIELE

- Aktivitäten zur Qualitätssicherung von Anforderungen
- Qualität von Architekturen
- Reifegrade für Softwareprozesse

QUALITÄTSMANAGEMENT

- „Im Mittelpunkt des umfassenden Qualitätsmanagements stehen die:
  - Kunden mit ihren Aufgaben, Prozessen, Erfordernissen und Erwartungen und
  - das Bemühen der Lieferanten, mit ihren Leistungen stets zu entsprechen.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 41, 43.

QUALITÄTSMANAGEMENT

- „Qualitätsmanagement wird nach ISO 9000:2005 als ‚aufeinander abgestimmte Tätigkeiten zum leiten und lenken einer Organisation bezüglich Qualität' definiert."

- „Qualitätsmanagement dient dabei als Oberbegriff für eine Vielzahl von Managementaufgaben und –aktivitäten zur Sicherstellung von Qualität in Prozessen, Projekten und Produkten."

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 41, 43.

DIN ISO 9001

- „Norm zum Thema Qualitätsmanagement“

- „DIN ISO 9000-Reihe. Diese Normen sind auf eine Fülle von stark unterschiedlichen Unternehmen anwendbar, und können daher keine spezifischen Techniken fordern.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 385

DIN ISO 9001

- „ISO 9001 ist im Wesentlichen frei von konkreten technischen Inhalten. Sie fordert die Durchführung bestimmter Tätigkeiten, z. B. die Beurteilung von Unterlieferanten oder die Existenz von Eingangsprüfungen. Sie legt nicht fest, wie diese Tätigkeiten zu realisieren sind. Im Wesentlichen definiert die DIN EN ISO 9001 einen organisatorischen Rahmen für das Qualitätsmanagement.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 385

ANSATZ DES QUALITÄTSMANAGEMENTS

- „Traditionelles, veraltetes (Qualitätsmanagement für Software wie auch für materielle Produkte) basiert auf dem Ansatz, das fertige Produkt mehr oder weniger gründlich auf Erfüllung der Anforderungen zu prüfen (klassisch durch Testen) und gefundene Abweichungen zu korrigieren. Dieser Ansatz hat sich aber als nicht ausreichend erwiesen.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 27

ANSATZ DES QUALITÄTSMANAGEMENTS

- „[Heute] geht man zunehmend dazu über, die Qualität des Software Produktes indirekt über die Qualität des Entwicklungsprozesses zu verbessern. Prüfungen und Tests sind dann Teilaufgaben dieses Prozesses und als solche nur Teil des gesamten Qualitätsmanagements. Dieser Ansatz basiert auf der Hypothese, dass man durch einen qualitativ hochwertigen, reifen Prozess auch qualitativ hochwertige Produkte herstellen kann.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 27

ROLLEN IM QUALITÄTSMANAGEMENT

In der Praxis kann eine Aufgabenverteilung wie folgt organisiert sein:

Softwareentwickler/innen: Zuständig für den Aufbau der Softwarelösung; Sicherstellung der Qualität der Komponente, für die sie verantwortlich sind

Software-Tester: Bewertet die Qualität der Software und verringert Risiko

Projektleiter/in / Manager/in: Verwaltet Softwareprojekte & leitet Softwareabteilungen

Reflexion: Wer übernimmt bei euch das Qualitätsmanagement?

KOSTEN VON QUALITÄT

Beachten: Es besteht eine Verbindung zwischen Qualitätsgrad (Wie hoch ist meine Qualität) und den Kosten, die für die Erstellung oder Erbringung notwendig sind.

- Zur Erbringung von Qualität werden Prüf- und Vorbeugekosten in Kauf genommen.

- Für Abweichungen in der Qualität werden Fehlerkosten in Kauf genommen.

BEDEUTUNG VON QUALITÄTSSICHERUNG

- „Im Schadensfall ergibt sich z. B. aus dem Produkthaftungsgesetz die Verpflichtung Ersatz eines Schadens, der durch ein fehlerhaftes Produkt entstanden ist.“

- „Ein Haftungsausschluss setzt voraus, dass der Fehler zum Zeitpunkt des in Verkehr bringen noch nicht vorlag, oder dass er nach dem Stand von Wissenschaft und Technik nicht erkennbar war, Die Beweislast für diesen Sachverhalt liegt im wesentlichen beim Hersteller.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 382

STANDARDS

- Softwarefehler können Schadensersatzforderungen nach sich ziehen
- „Besonders Firmen, die sicherheitskritische Systeme entwickeln, müssen im Schadensfall erhebliche juristische Konsequenzen fürchten. “
- Der Softwarehersteller trägt die Beweislast, nachzuweisen, dass sein Produkt fehlerfrei geliefert wurde oder ein Fehler nicht hätte erkannt werden können.

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 382

STANDARDS

- „Durch Einhaltung der jeweils relevanten Normen kann ein Hersteller sicherstellen, dass der Stand der Technik erreicht ist, und er damit seine Sorgfaltspflicht erfüllt hat.“

Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 382

Kernaktivität:

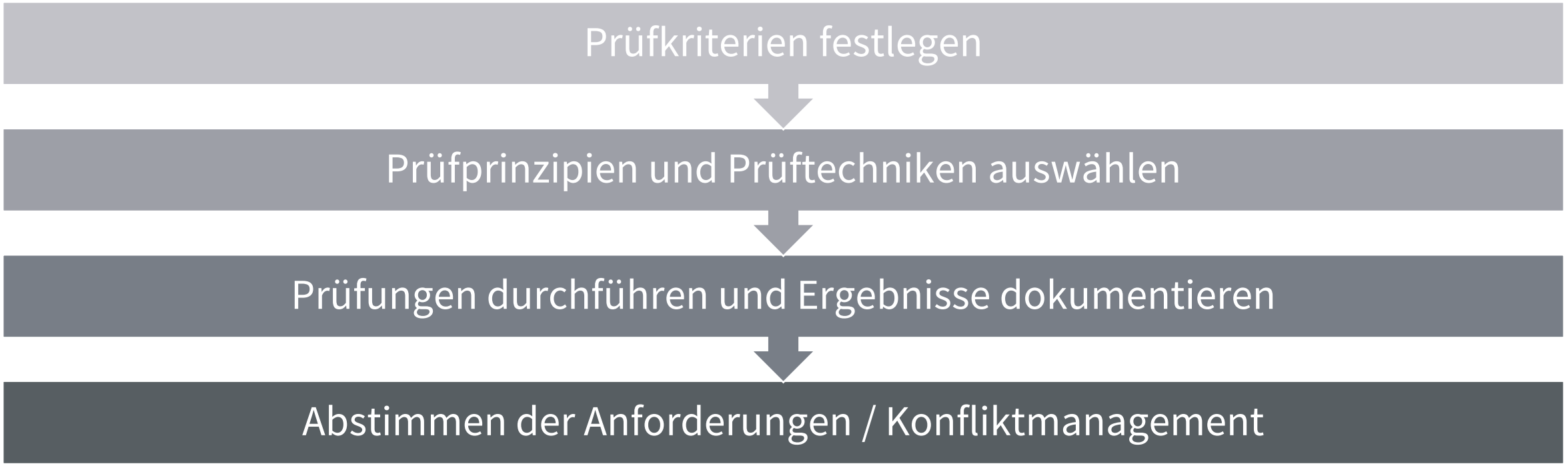
- Requirements Engineering (Anforderungsmanagement), auch entwicklungsbegleitend
- fachliche und technische Korrektheit
- Abstimmung mit allen relevanten Stakeholdern
- Achtung: Für Anforderungen gilt → vollständiges Testen ist nicht möglich

SHORTDISCUSSION

Reflektion:

- Was hat Produktqualität mit Prozessqualität zu tun (Insbesondere bei Software)?

REMINDER: QUALITÄTSSICHERUNG VON ANFORDERUNGEN



REMINDER: QUALITÄTSSICHERUNG VON ANFORDERUNGEN

Prüfkriterien für Anforderungen
Prüfen des Inhalts
Prüfen der Dokumentation
Prüfkriterien zur Abgestimmtheit

Prüfprinzipien
Beteiligung der richtigen Stakeholder
Trennung von Fehlersuche und Fehlerkorrektur
Prüfung aus unterschiedlicher Sicht
Geeigneter Wechsel von Entwicklungsartefakten
Konstruktion von Entwicklungsartefakten
Wiederholte Prüfung

Was versteht man noch einmal unter Softwarearchitektur?

„Eine Softwarearchitektur ist einer der Architekturtypen in der Informatik und beschreibt die grundlegenden Komponenten und deren Zusammenspiel innerhalb eines Softwaresystems.“

Wikipedia-Autoren. (2003, 6. Dezember). Softwarearchitektur. https://de.wikipedia.org/wiki/Softwarearchitektur

Prüfung, der Eignung der Architektur zur Erfüllung der Anforderungen

Ex ante oder ex Post?

- **Ex Ante (vor der Implementierung):** Szenariobasiertes Vorgehen: Bewertung anhand von definierten Anwendungsszenarien => Architecture Trade off Analysis Methode (ATAM)

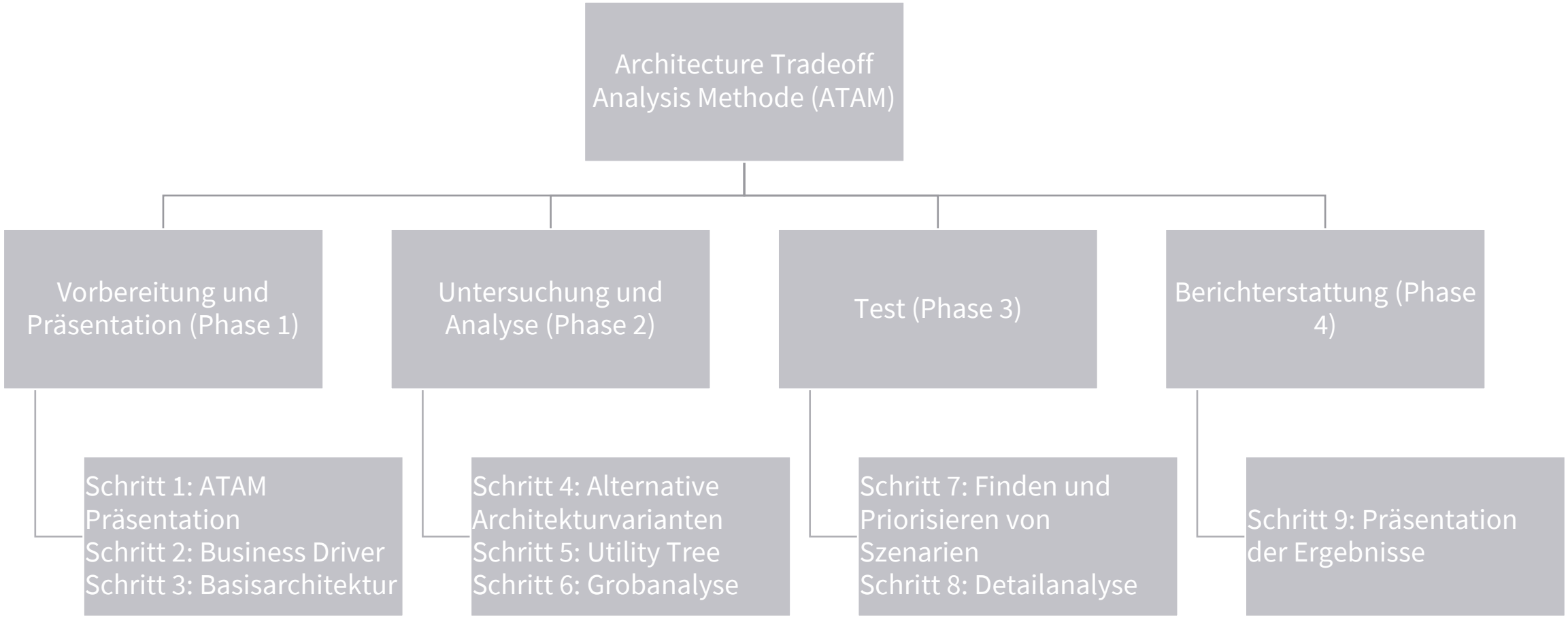
- **Ex Post (nach der Implementierung):** Prüfen der Implementierung gegen Architekturbeschreibung (Architecture Compliance Checking)

QUALITÄTSSICHERUNG VON ARCHITEKTUREN: ATAM

- Szenariobasiertes Vorgehen, d.h. Bewertung anhand von definierten Anwendungsszenarien (Ex ante)
  - Erkennt Trends
  - Früh anwendbar

- “When evaluating an architecture using the ATAM, the goal is to understand the consequences of architectural decisions with respect to the quality attribute requirements of the system.”
  (inkl. des Trade-Offs der Ziele Architekturentscheidungen)

QUALITÄTSSICHERUNG VON ARCHITEKTUREN: ATAM



Kazman, R., Klein, M., Clementes, P. (2000). ATAM: Method for Architecture Evaluation, CMU/SEI.

QUALITÄTSSICHERUNG: PROZESSANALYSE

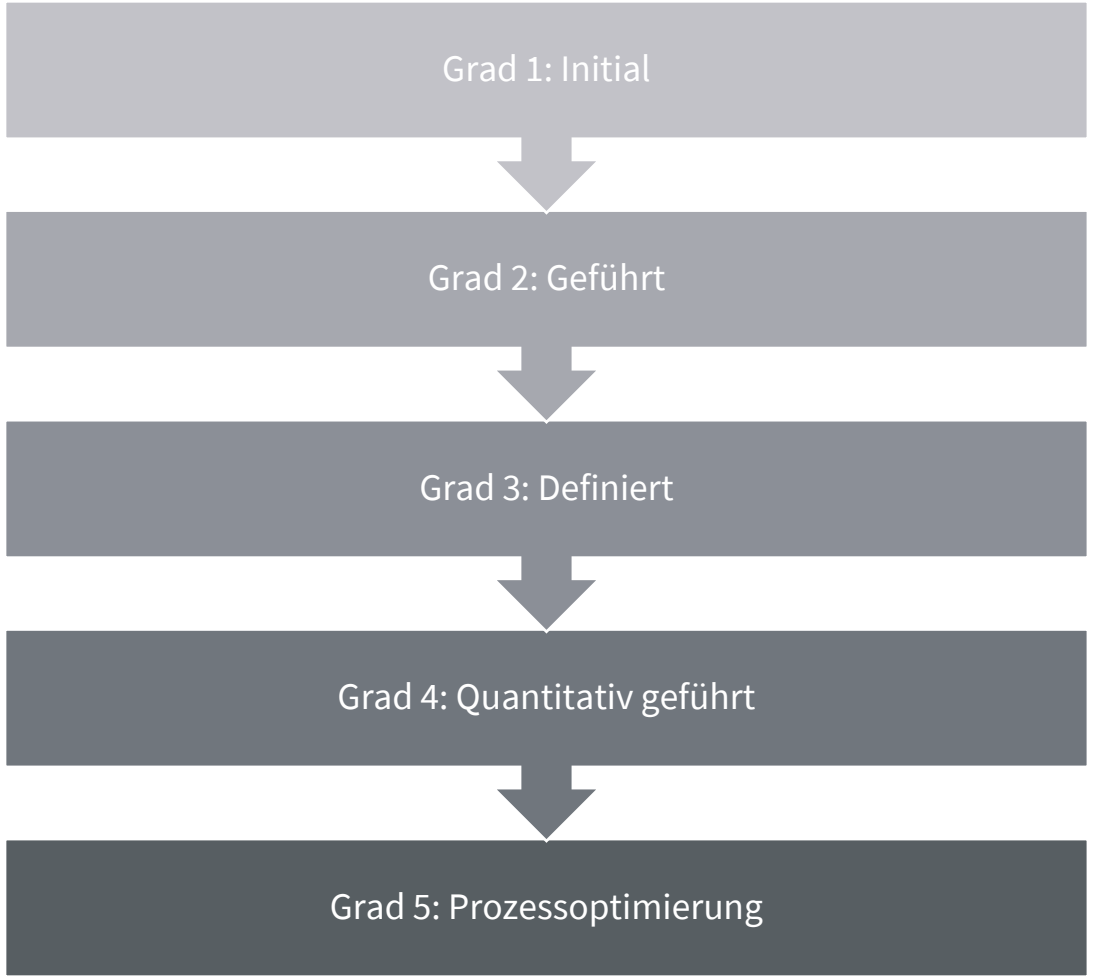
- Durch Stakeholderbefragung mit Hilfe von Prozessmerkmalen
- Reifegradmodelle z.B. Capability Maturity Model Integration

QUALITÄTSATTRIBUTE FÜR SOFTWAREPROZESSE

- Verständlichkeit
- Standardisierung
- Sichtbarkeit
- Messbarkeit
- Unterstützbarkeit
- Akzeptanz
- Zuverlässigkeit
- Stabilität
- Wartungsfreundlichkeit
- Schnelligkeit

QUALITÄTSSICHERUNG: CAPABILITY MATURITY MODEL INTEGRATION

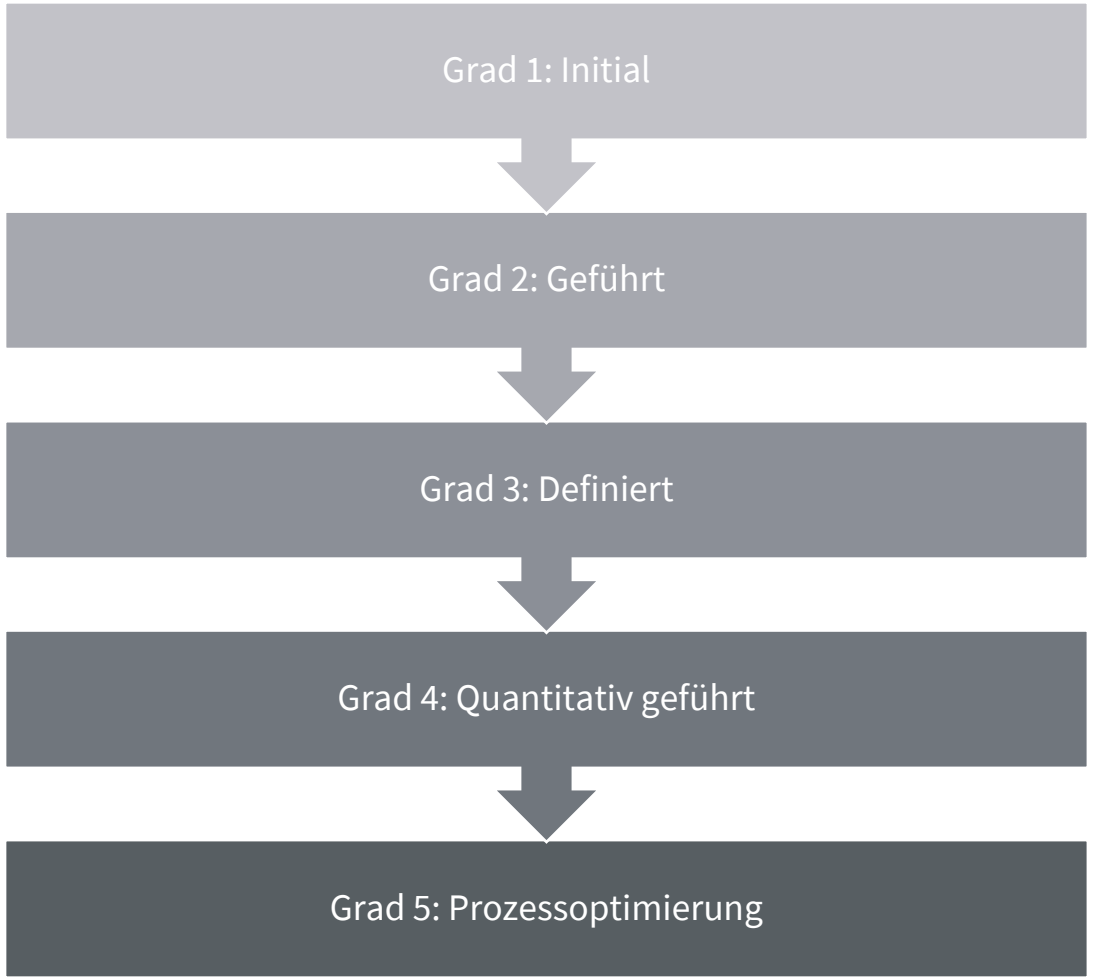
- Rahmenwerk für Prozessverbesserungen
- Stufenmodell
- 22 Prozessgebiete aus dem Software Engineering
- Ziele, die spezifisch für jedes Prozessgebiet formuliert werden; sowie
- Vorgehensweisen, mit denen die Ziele erreicht werden können.



QUALITÄTSSICHERUNG: CAPABILITY MATURITY MODEL INTEGRATION

Stufe 1 –Initial:

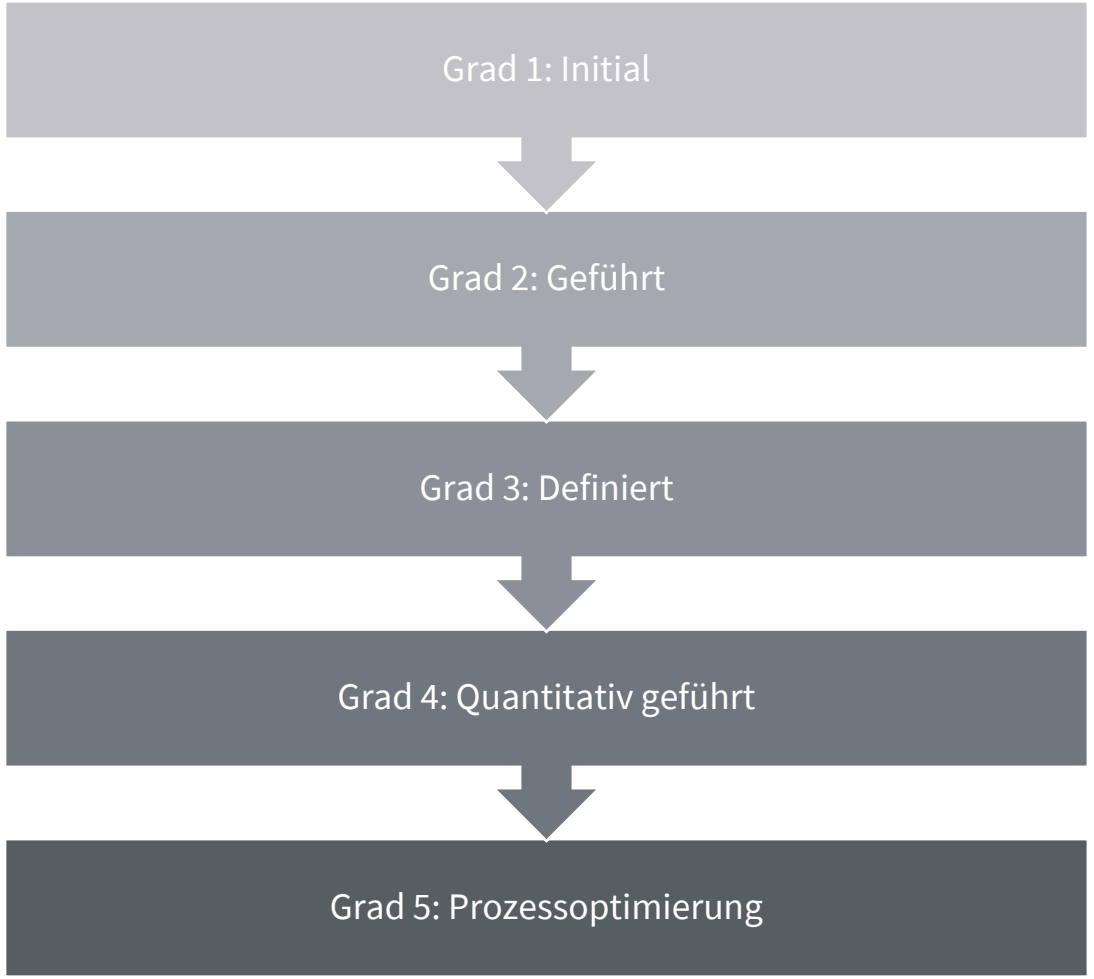
- Es gibt keine Prozessdefinition, die Aktivitäten finden ad hoc statt bzw. werden chaotisch durchgeführt. Der Prozess ist in der Regel nicht wiederholbar.



QUALITÄTSSICHERUNG: CAPABILITY MATURITY MODEL INTEGRATION

Stufe 2 –Geführt:

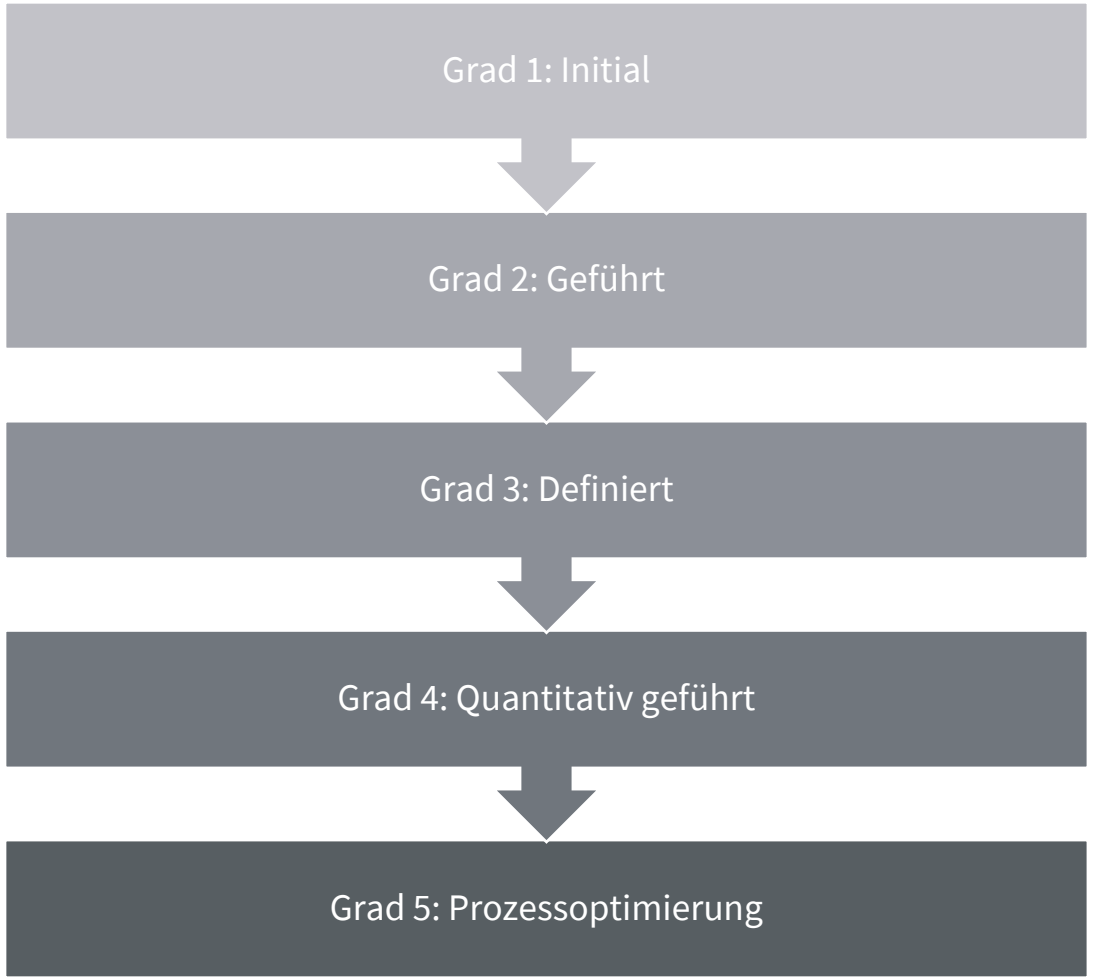
- Der Prozess ist wiederholbar. Umfang und Ergebnis der zu erledigenden Aktivitäten ist den Teammitgliedern bekannt. Es gibt aber keinen detaillierten Prozess, der Aktivitäten, Rollen, Ergebnisse und deren Abhängigkeiten explizit definiert.



QUALITÄTSSICHERUNG: CAPABILITY MATURITY MODEL INTEGRATION

Stufe 3 –Definiert:

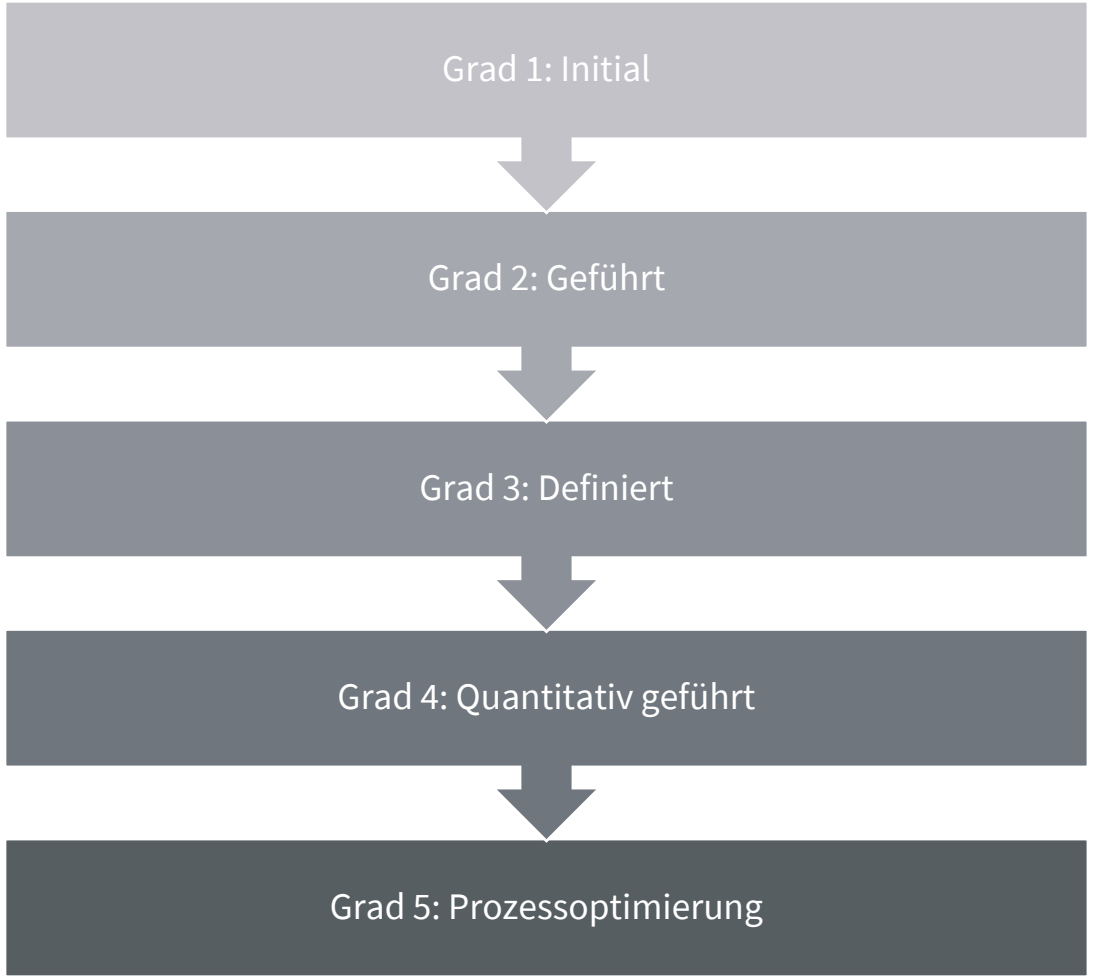
- Es gibt einen definierten Prozess, in dem Aktivitäten, Rollen und Ergebnisse verbindlich dokumentiert sind. Der Prozess enthält konkrete Methoden und Vorgehensweisen zu den einzelnen Software Engineering Aktivitäten.



QUALITÄTSSICHERUNG: CAPABILITY MATURITY MODEL INTEGRATION

Stufe 4 –Quantitativ geführt:

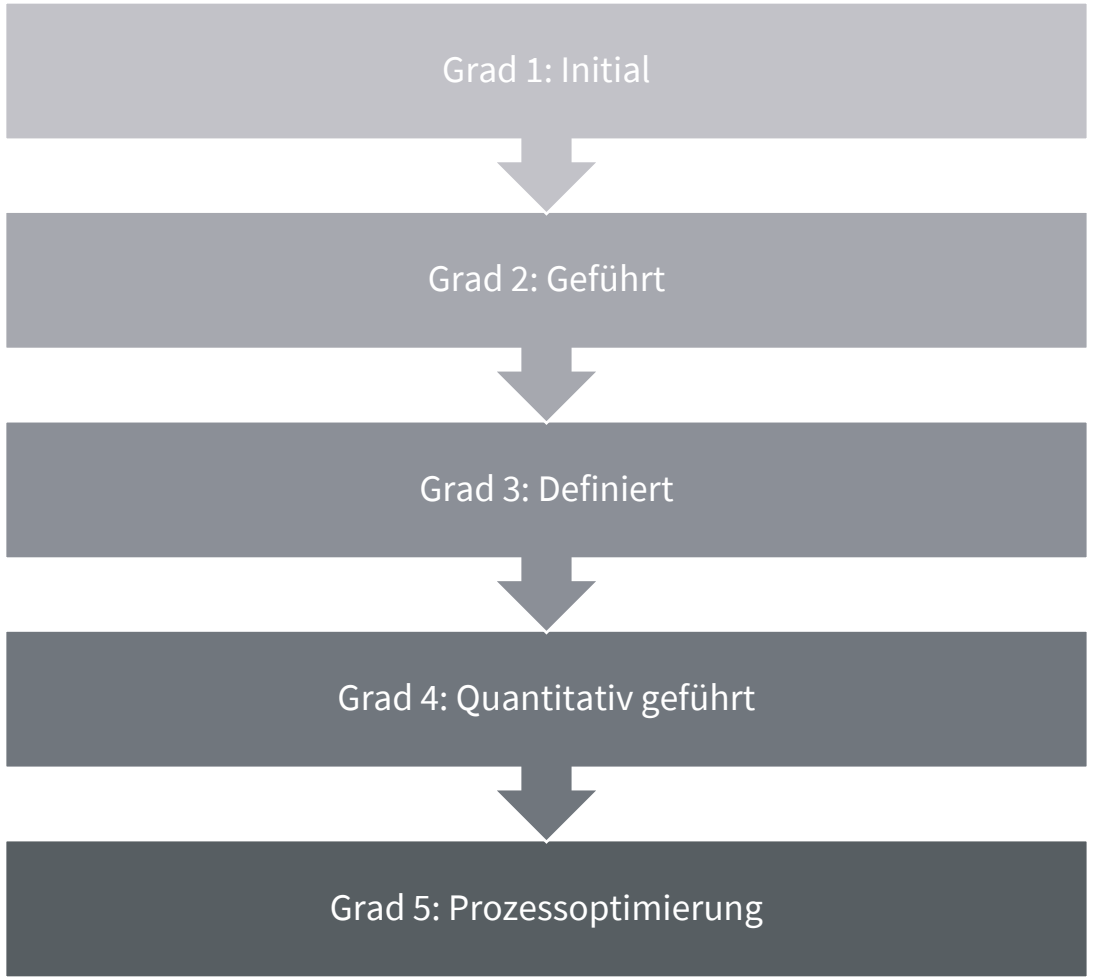
- Die Prozessqualität kann über Messung quantitativer Prozesseigenschaften festgestellt und gesteuert werden. Über erhobene Kennzahlen zu Aktivitäten der Prozesse sowie zu den erzeugten Ergebnissen können Havarien identifiziert und entsprechend gegengesteuert werden.



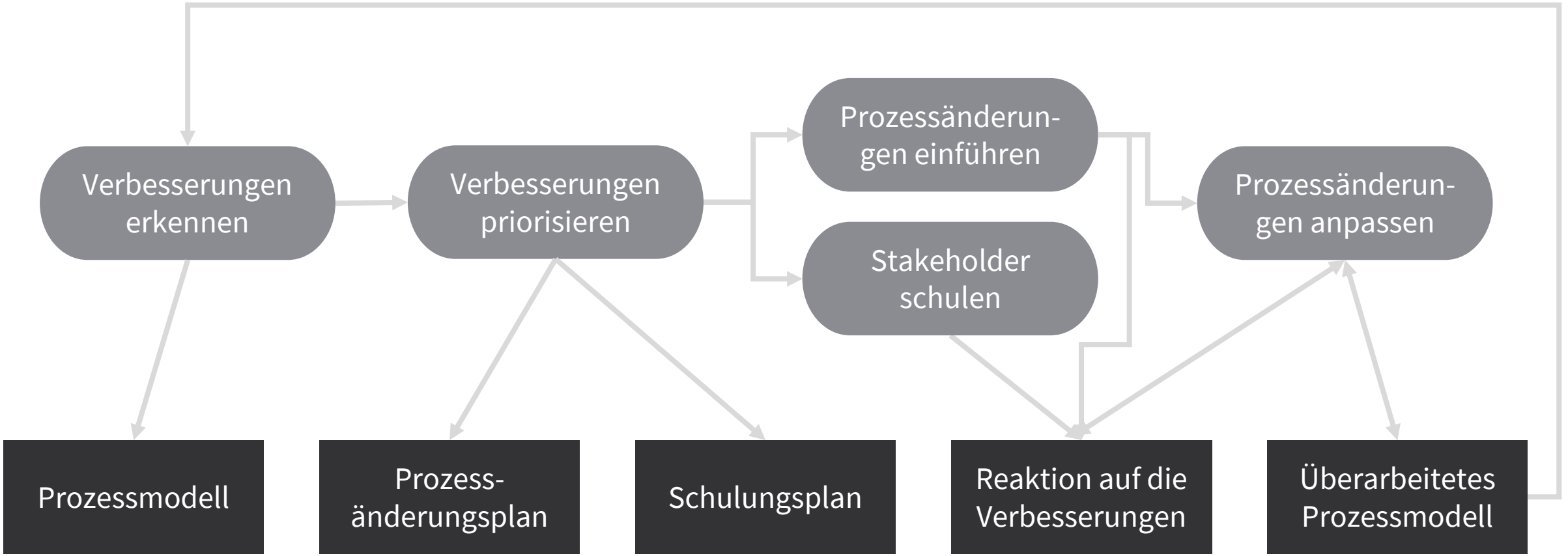
QUALITÄTSSICHERUNG: CAPABILITY MATURITY MODEL INTEGRATION

Stufe 5 –Prozessoptimierung:

- Prozess- und Produktmessungen werden kontinuierlich erhoben und zur Prozessverbesserung eingesetzt. Je nach Messwert kann der Prozess auf den aktuellen Bedarf hin angepasst werden.



PROZESSVERBESSERUNGEN



SHORTDISCUSSION

Gruppendiskurs:

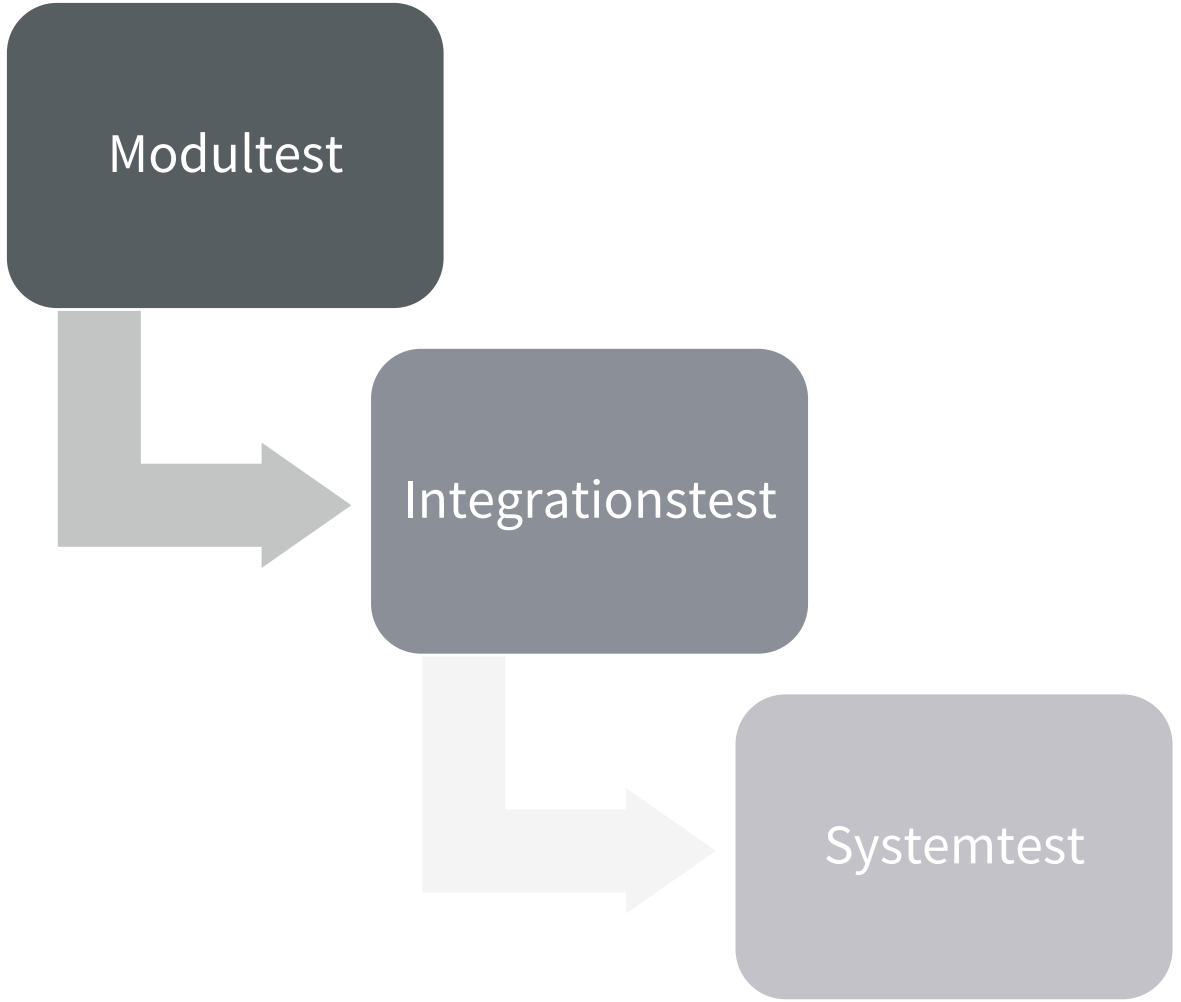
Warum ist es eigentlich nicht möglich Anforderungen komplett zu testen?

Welches sind vernünftige Größen zum testen?

REMINDER: ARTEN DER QUALITÄTSSICHERUNG IN DER SOFTWAREENTWICKLUNG

Modultest:

„Das Ziel des Modultests ist die Überprüfung einzelner Komponenten (Module) gegen ihre Spezifikation.“

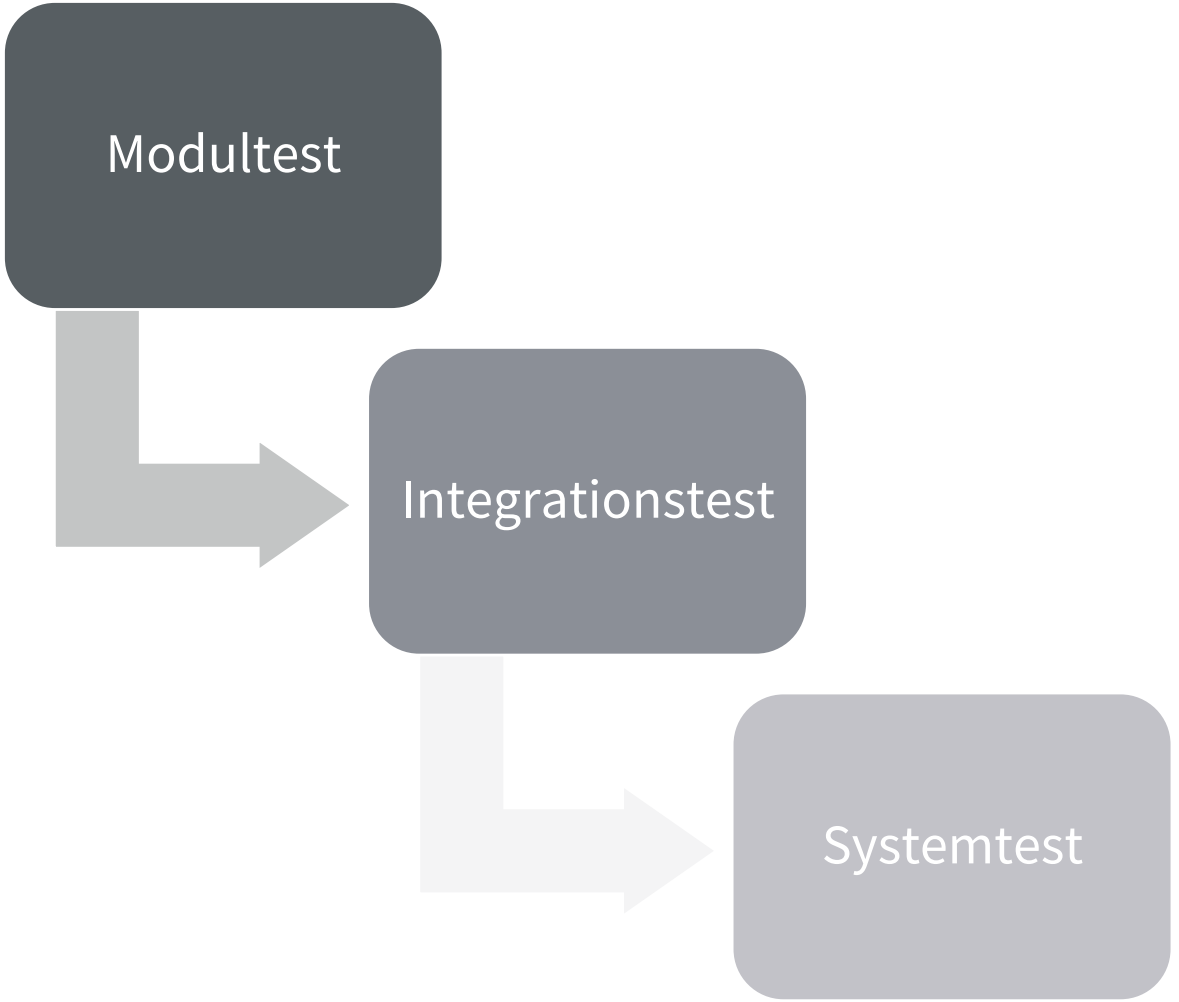


Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 370-371

REMINDER: ARTEN DER QUALITÄTSSICHERUNG IN DER SOFTWAREENTWICKLUNG

Integrationstest:

„Das Ziel des Integrationstests ist die Überprüfung des Interagierens verschiedener Module über Schnittstellen. In dieser Phase sind der innere Aufbau von Modulen und die von der derzeit betrachteten Schnittstelle weit entfernten Schnittstellen nicht relevant.“

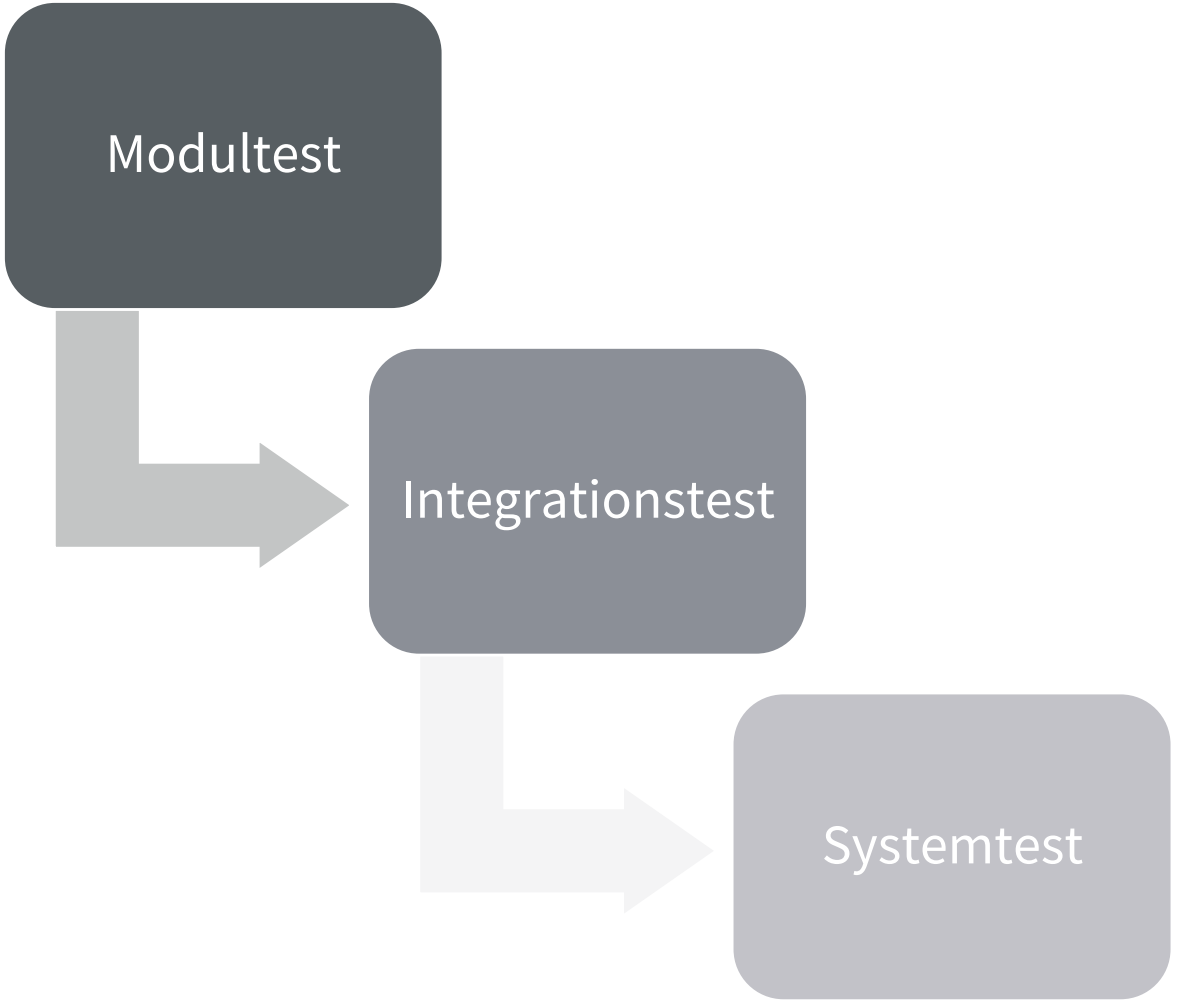


Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 370-371

REMINDER: ARTEN DER QUALITÄTSSICHERUNG IN DER SOFTWAREENTWICKLUNG

Systemtest:

„Die Ziele des Systemtests sind die Prüfung der Funktion, Leistung und Qualität des fertig integrierten Systems gegen die Anforderungsdokumente. Im Systemtest werden die inneren Strukturen von Systemen nicht betrachtet.“



Liggesmeyer, Peter (2009): Software-Qualität. Testen, Analysieren und Verifizieren von Software. 2. Auflage. Heidelberg: Spektrum Akademischer Verlag, Springer. S. 370-371

08

INTERNATIONALE HOCHSCHULE

REFERATE

TERMINE:

Matrikelnr.	Thema	Datum	Start	Ende
102100879	Verwaltung von Arzneimitteln (Lagersystemen)	11.07.2023	9:30	10:30
102106605		11.07.2023	9:30	10:30
102102243		11.07.2023	9:30	10:30
102100503		11.07.2023	9:30	10:30
102101117	Lernkarten (Die einzig wahre APP)	11.07.2023	10:45	11:15
102107273		11.07.2023	10:45	11:15
102107675	Generierung von Passwörtern (Die Passwort Gruppe)	11.07.2023	11:30	12:15
102107019		11.07.2023	11:30	12:15
102106238		11.07.2023	11:30	12:15
102100408	Lernkarten (Luca Lernkarten APP)	11.07.2023	13:00	13:30
102106081		11.07.2023	13:00	13:30
102101455	Generierung von Zufallszahlen - Ami Paris	11.07.2023	13:45	14:30
102100511		11.07.2023	13:45	14:30
102106917		11.07.2023	13:45	14:30

DANKE

IU International Hochschule
Duales Studium
Siemensstraße 10
30173 Hannover

Prof. Dr. Knut Linke
0511 310109 35
knut.linke@iu.org