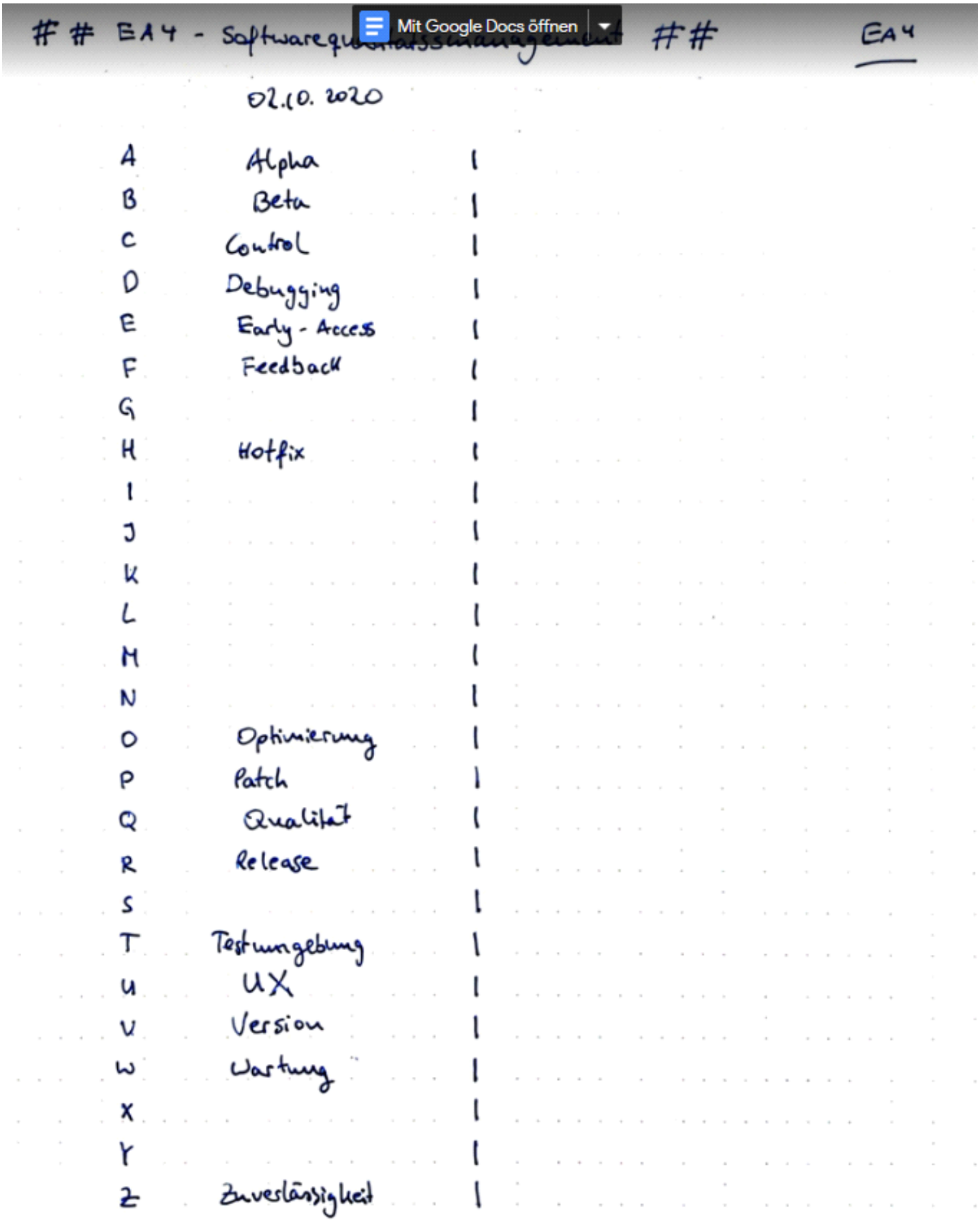
EA 4 - 02.10.2020 Softwarequalitätsmanagement

Freitag, 9. Oktober 2020 12:30



Erfasster Bildschirmausschnitt: 09.10.2020 12:31

EA 4 - 02.10.2020 Vorgehensmodelle

Freitag, 9. Oktober 2020 12:01

1. Vorgehensmodelle

Vorgehensmodelle dienen zur Strukturierung eines mit dem Ziel, Komplexität beherrschbar zu machen.

(" divide et impera ")

Die Phasen dienen zur Zuordnung zu zeitlichen und/oder fachlichen und/oder organisatorischen Aspekte

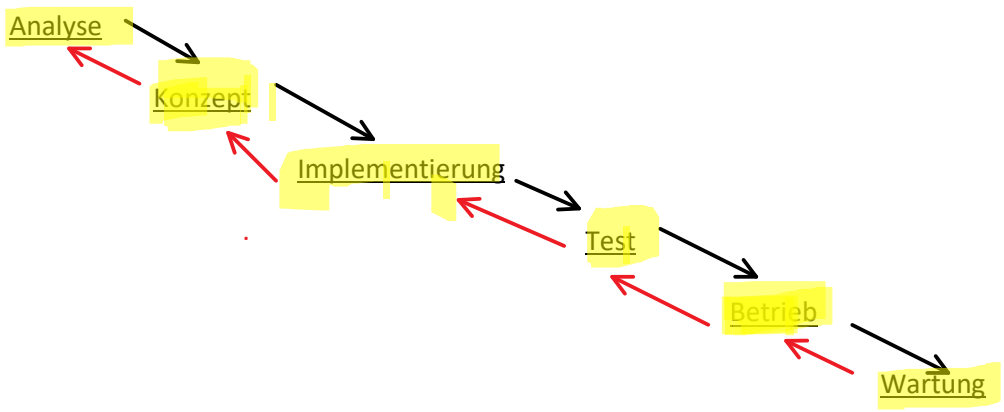
2. Arten von Vorgehensmodellen

- Sequenzielle V.
- Nebenläufige V.
- Inkrementelle V.
- Spezielle / Mischmodelle V.

3. Sequenzielle Vorgehensmodelle

- Eine Phase startet nach Abschluss der vorherigen Phase
- Geringer Managementaufwand
  - Projektdauer = Summe der Phasendauern
- Setzt "endgültige" Definitionen voraus

1.2.1 Wasserfallmodell



Jede Phase erzeugt ein Dokument

Sequenzieller Ablauf mit Rückkopplung auf Nachbarphase (Bei Bedarf)

Lastenheft:	**Laie**
Pflichtenheft:	**Profi**

EA 4 - 29.01.2021 - Versionskontrolle

Freitag, 29. Januar 2021 12:49

Versionsverwaltung

(Synonyme: Versionskontrolle, VCS → Version Control System)

- Werkzeug für Speicherung und Abruf von Änderung an Dateien
- verwaltet "Stände"(Versionen, Zeitstempel,...) für jede Datei
- koordiniert gemeinsamen Zugriff
- primär für Quelltextdateien erstellt (speicher alles ab)
- Verwendung von VCS u.a. in/als separates Produkt oder integriert in Business Software (z.B. Content Management System (CMS), Dokumentationssystem, Synchro.-sw.)

Grundkonzepte

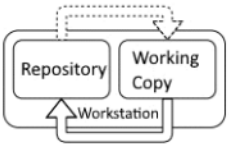
➔ **Repository:** Verzeichnis zur Lagerung von Projektinhalten (Dateien), das neben den eigentl. Inhalten auch Änderungsinfos enthält / Inhalte werden nicht direkt bearbeitet

➔ **Working copy:** Kopie eines bestimmten Standes (Version) eines Repos, an dem Änderungen vorgenommen werden

Klassifizierung nach Verteilungsgrad

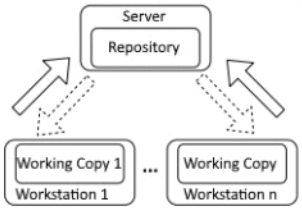
1. Lokale Versionsverwaltung

- (einzelne) Dateien auf lokalem Rechner versioniert
- z.B. GNU RCS ("Revision Control System", 1982)



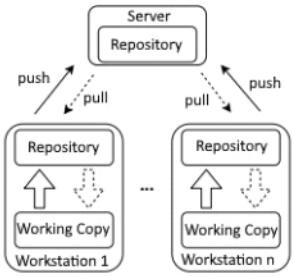
2. Zentrale Versionsverwaltung

- Zentraler Server zuständig für alle Clients
- CVS (1990)
  Subversion (2000)



3. Verteilte Versionsverwaltung (Distributed VCS → DVCS)

- Zentraler Server ist optional, Dateien liegen lokal imd werden synchronisiert
- Bitkeeper (2000), git (2005)



EA 4 - 12.02.21

Freitag, 12. Februar 2021 13:23

Test-Begriff

Hefter!

> Ein Test ist eine stichprobenartige Prüfung um Fehlerwirkungen gezielt und systematisch aufzudecken

QUELLE: [1], KAPITEL 2.1.2

7 / 15

Test-Begriff

Hefter!

Ergänzungen:

- **stichprobenartige Prüfung**: für nicht-einfache Programme ist ein vollständiger Test i. A. nicht möglich
- **Fehlerwirkung**: Auswirkungen von Fehlerursachen zu finden (nicht die Ursachen selbst)
- **gezielt und systematisch**: Einführung eines Testmanagements

Test-Gründe

Hefter!

Warum testen?

- Stabilität herstellen: Fehlerwirkungen (z. B. Programmab-

Warum testen?

- Stabilität herstellen: Fehlerwirkungen (z. B. Programmabsturz) erkennen
- Qualitätsbestimmung: z. B. bei Abnahmetests zur Projektabnahme
- Wirtschaftlichkeit erhöhen: „in time“, „in budget“, „in quality“
- Fehlerwirkungen vorbeugen: Auseinandersetzung mit Programm und/oder Dokumenten
- Kompetenz transportieren: Marketingaspekte liefern (Testabdeckung, Lines of Code (LOC))

Fehlermeldung - Inhalte

1) Identifikation

- Nummer
- Testobjekt (Produkt)
- Version (des Testobjektes)
- Umgebung (HW, SW, inkl. Version, Einsatzort)
- Melder
- Entwickler
- Erfassung (Datum und Uhrzeit)

12 / 15

EA 4 - 19.02.21

Freitag, 19. Februar 2021 12:00

Fehlermeldung - Inhalte

Hefter!

2) Klassifikation

- Bearbeitungsstatus („Neu“, „In Arbeit“, ...)
- Problemklasse („Betriebsverhindernd“, „Leichter Fehler“, ...)
- Priorität (Dringlichkeit der Korrektur)
- Anforderung (Verweis auf Anforderung in Spezifikation)
- Fehlerquelle (Vermutung, soweit möglich)

13 / 15

Fehlermeldung - Inhalte

Hefter!

3) Problembeschreibung

- Testfall (ggf. Verweis auf Testfallnummer; Schritte zur Reproduktion)
- Problem (erwartetes und tatsächliches Verhalten)
- Kommentar (z. B. Hinweise vom Entwickler)
- Verweis (auf andere Meldungen, die relevant sind)

14 / 15

Hefter!

Komponententest

- Prüfung einzelner Softwarebausteine (Module, Units, Klassen) gegen Spezifikationen
- Sicherstellung der Gesamtfunktionalität durch mehrere Testfälle für Teilfunktionalitäten
- Synonym: Modultest, Unittest
- Quelltextnah

Hefter!

Komponententest - Beispiel

Zu testende Komponente:

```
public int berechneProvision(int projektwert, int
projektlaufzeit){
 return (int)(projektwert *0.08 / projektlaufzeit);
}
```

dazu gehöriger Komponententest:

```
void testBerechneProvision(){
 assertEquals(400,berechneProvision(25000, 5));
}
```

Hefter!

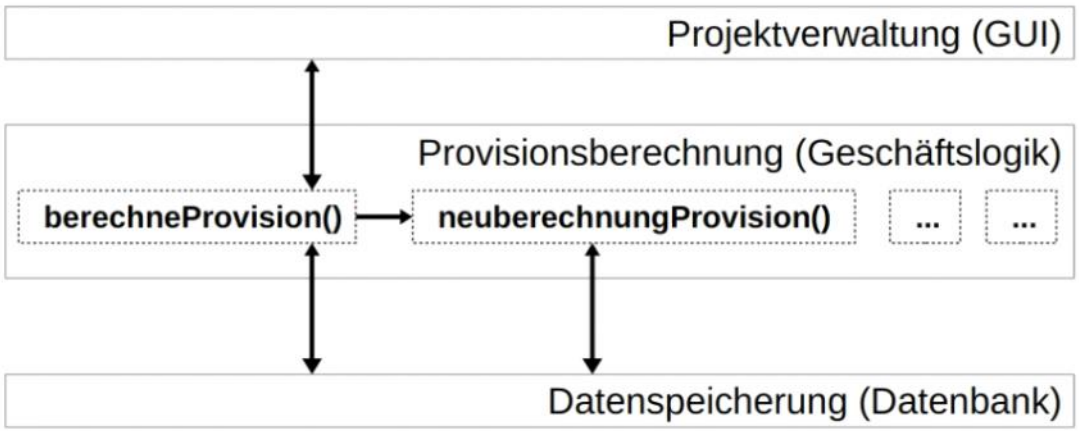
Integrationstest

- Prüfung mehrerer Komponenten gegen den technischen Entwurf (Architektur)
- Voraussetzung: verbundene Komponenten arbeiten einzeln möglichst fehlerfrei
- Ziel: Fehlerzustände / Wechselwirkungen in Schnittstellen/im Zusammenspiel erkennen
- Typische Fehlerwirkungen: Timing-, Durchsatz-, Lastprobleme

5 / 9

Hefter!

Integrationstest - Beispiel



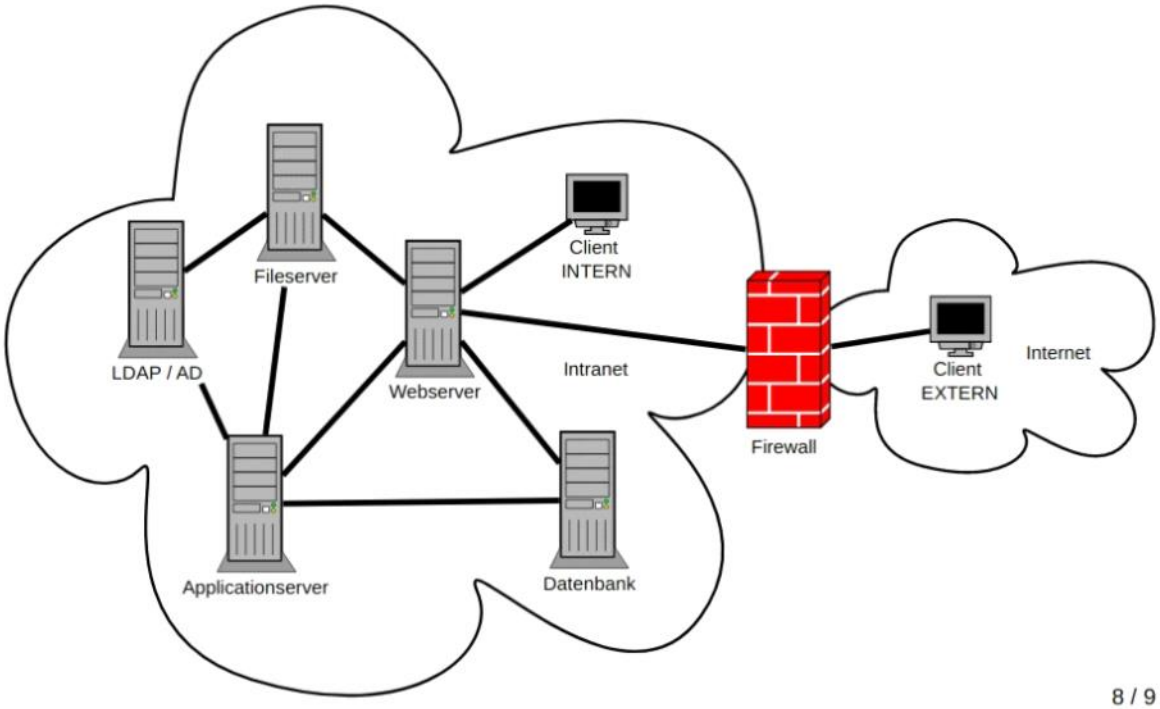
6 / 9

Systemtest

- Prüfung des Gesamtsystems gegen den Anforderungskatalog
- Durchführung in Testumgebung, die der Produktivumgebung möglichst nahe kommt
- Berücksichtigung von Verteilungen, Netzwerk, Treiber, Drittsoftware, etc.

7 / 9

Systemtest - Beispiel



8 / 9

Abnahmetest

Hefter!

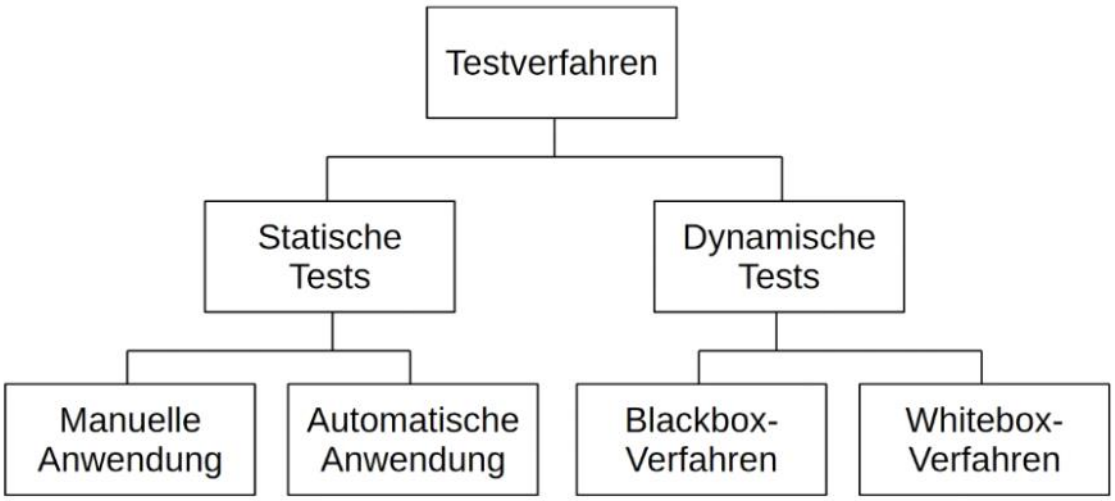
- Prüfung des Systems aus Kundensicht, ob die vereinbarten Leistungen erfüllt wurden
- übliche Formen:
  - Test auf vertragliche Akzeptanz
  - Test auf Benutzerakzeptanz
  - Feldtests (Alpha- und Beta-Tests)
- findet teilweise auch in anderen Teststufen statt

EA 4 - 23.03.21

Dienstag, 23. März 2021 08:50

Überblick Testverfahren

Hefter!



2 / 13

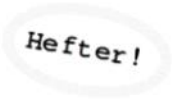
Statische Tests: Manuelle Ausführung

Hefter!

- Menschen führen eine Analyse durch
- Analyse von Spezifikationen, Quelltexten, …
- Prüfung mittels Ist-/Soll-Vergleich, z. B. „Konzept gegen Quelltext“, „Quelltext gegen Konvention“
- Beispiele:
  - Review
  - Audit
  - Peer Programming

- Audit
- Peer-Programming

Statische Tests: Automatische Ausführung



- Programme führen eine Analyse durch
- Analyse von Spezifikationen, Quelltexten, …
- Voraussetzung: Informationen liegen in einem maschinenlesbaren Format vor (z. B. UML, Quelltext, XML)
- Prüfung gegen Spezifikationen, Best Practice, Wörterbücher
- Beispiele:
  - Compiler
  - Metriken aus statischer Codeanalyse

EA 4 - 28.05.21

Freitag, 28. Mai 2021 10:13

Dynamische Tests

Hefter!

- Untersuchung von Quelltext
- Test mit Testdaten und Ausführung
- Vorgehen:
  - Ausführung des Testobjektes auf einem Prozessor
  - Bereitstellung der Eingabedaten
  - Auswerten der Ausgabedaten

10 / 13

Blackboxtests

Hefter!

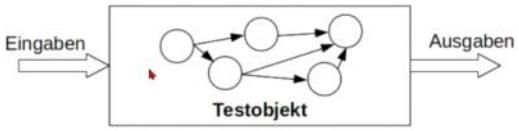
- Arbeiten ohne Kenntnis der Implementierung des zu testenden Objektes
- Verhalten des Testobjekts wird von außen beobachtet (Point-of-Observation (PoO) außen)
- Ablauf des Tests wird nur durch die Wahl der Eingaben gesteuert (Point-of-Control (PoC) außen)
- Testfälle basieren auf den Anforderungen (Datentypen, zulässige Wertebereiche, usw.)
- Beispiele
  - Smoketest
  - Entscheidungstabellentest

11 / 13

Hefter!

Whiteboxtests

- Arbeiten unter Kenntnis der Implementierung des zu testenden Objektes
- Bei Ausführung der Testfälle wird der innere Ablauf im Testobjekt analysiert (PoO innen)
- Eingriff in den Ablauf im Testobjekt ist möglich (PoC *kann* innen liegen)
- Ergänzende Testfälle können auf Grund der Programmstruktur des Testobjektes gewonnen werden
- Beispiele
  - Anweisungsüberdeckungstest
  - Zweigüberdeckungstest



12 / 13

Hefter!

Dynamische Tests

- Untersuchung von Quelltext
- Test mit Testdaten und Ausführung
- Vorgehen:
  - Ausführung des Testobjektes auf einem Prozessor
  - Bereitstellung der Eingabedaten
  - Auswerten der Ausgabedaten

10 / 13

Übung Testerstellung
Szenario 1 - TK-Anlage

S1 - Übersicht Testszenarien und -fälle

Testszenario A - „Konferenzen“

Testfall A1	Konferenzen mit internen Teilnehmern
Testfall A2	Konferenzen mit gemischten Teilnehmern

Testszenario B – „“Weiterleitungen“

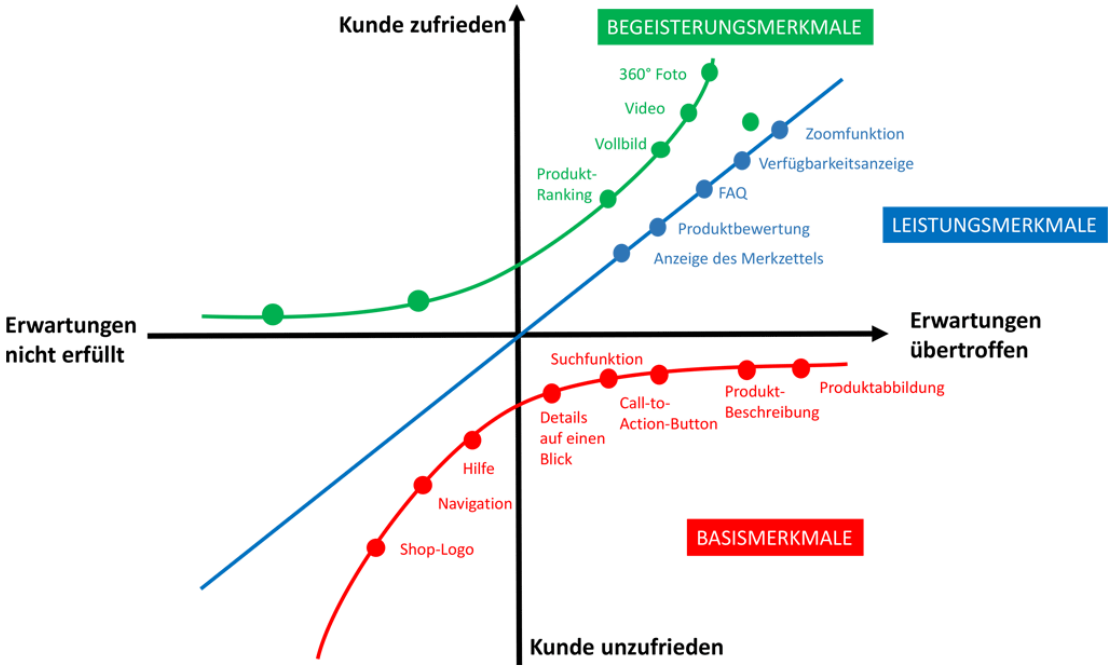
Testfall B1	Anrufe mit Weiterleitungen einer Tiefe bis 2
Testfall B2	Anrufe mit Weiterleitungen einer Tiefe über 2
Testfall B3	Prüfung bei eingerichtetem anschlussbezogenen AB
Testfall B4	Prüfung bei nicht eingerichtetem anschlussbezogenen AB
Testfall B5	Prüfung auf AB-Funktion bei „besetzt“

EA 4 - 20.07.21

Dienstag, 20. Juli 2021 08:07

Produkt und Prozesszufriedenheit messen:

Kano - Modell

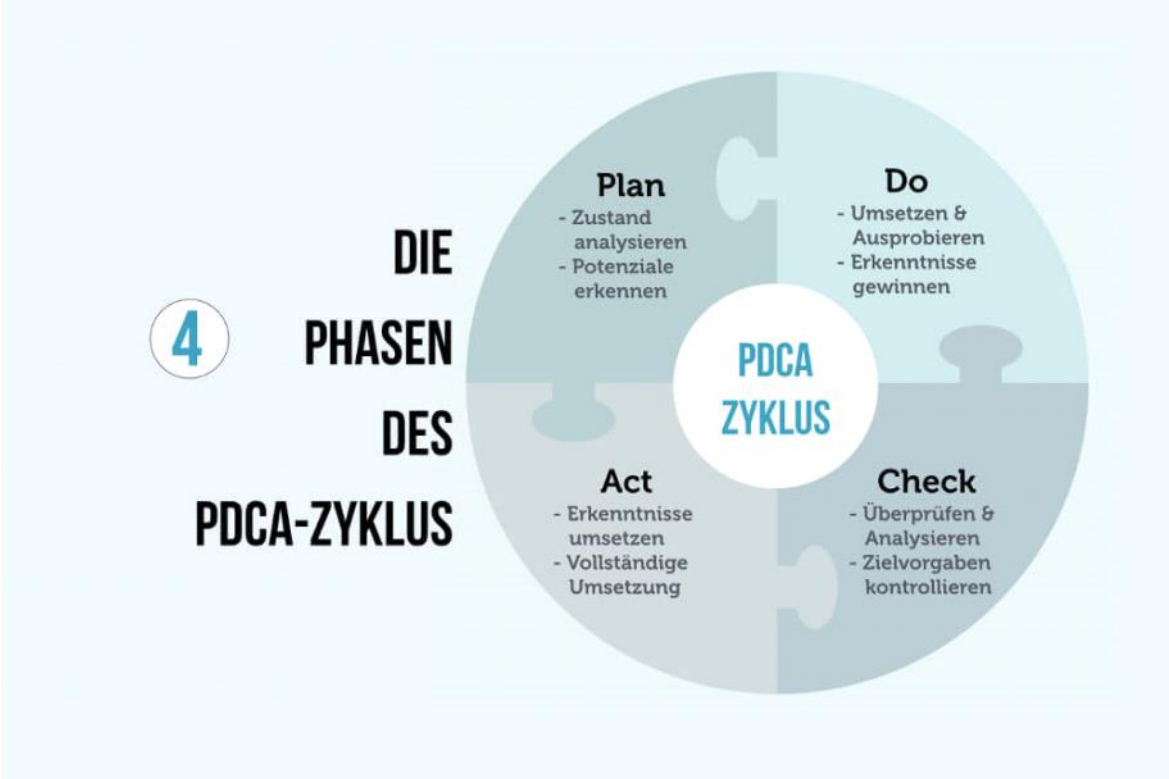


Qualitätsebenen:

- **Basismerkmale**: selbstverständlich, vorausgesetzt, Fehlen wird i. d. R. spät bemerkt, z. B. unterstützte Import - Formate
- **Leistungsmerkmale**: explizit erwartet, Vergleichskriterien, z. B. Speicherkapazität, Datendurchsatz
- **Begeisterungsmerkmale**: nicht erwartet, evtl. Alleinstellungsmerkmale, z. B. unaufgeforderter Anruf beim Kunden

weitere Qualitätsebenen:

- **Unerhebliche Merkmale**: Nicht notwendige Funktionen, keine Zufriedenheitswirkung, z. B. E-Rechnung für Dienstleistungen
- **Rückweisungsmerkmale**:
  - Vorhandensein -> Unzufriedenheit
  - Nicht-Vorhandensein -> Zufriedenheit, z. B. veraltetes Hardwaremodell


4
DIE
PHASEN
DES
PDCA-ZYKLUS
Plan
- Zustand analysieren
- Potenziale erkennen
Do
- Umsetzen & Ausprobieren
- Erkenntnisse gewinnen
PDCA ZYKLUS
Act
- Erkenntnisse umsetzen
- Vollständige Umsetzung
Check
- Überprüfen & Analysieren
- Zielvorgaben kontrollieren

EA 4 - 20.07.21

Dienstag, 20. Juli 2021 09:20

A - Alpha
B - Blackbox Tests
C - Controlling
D - Dialogfenster
E - Effizienz
F - Feedback
G - Graybox Test
H
I - Inkrementelles Vorgehensmodell
J
K - Kano Modell
L - Leistung
M
N - Nutzbarkeit
O
P
Q - Qualitätsmerkmale
R - Release
S - Schreibtischtest
T - Testumgebung
U - Übertragbarkeit
V - Versionsverwaltung
W - Whitebox Tests
X
Y
Z - Zuverlässigkeit