ISTQB® Certified Tester, Foundation Level
Effektiver Softwaretest – Methoden, Techniken und Werkzeuge

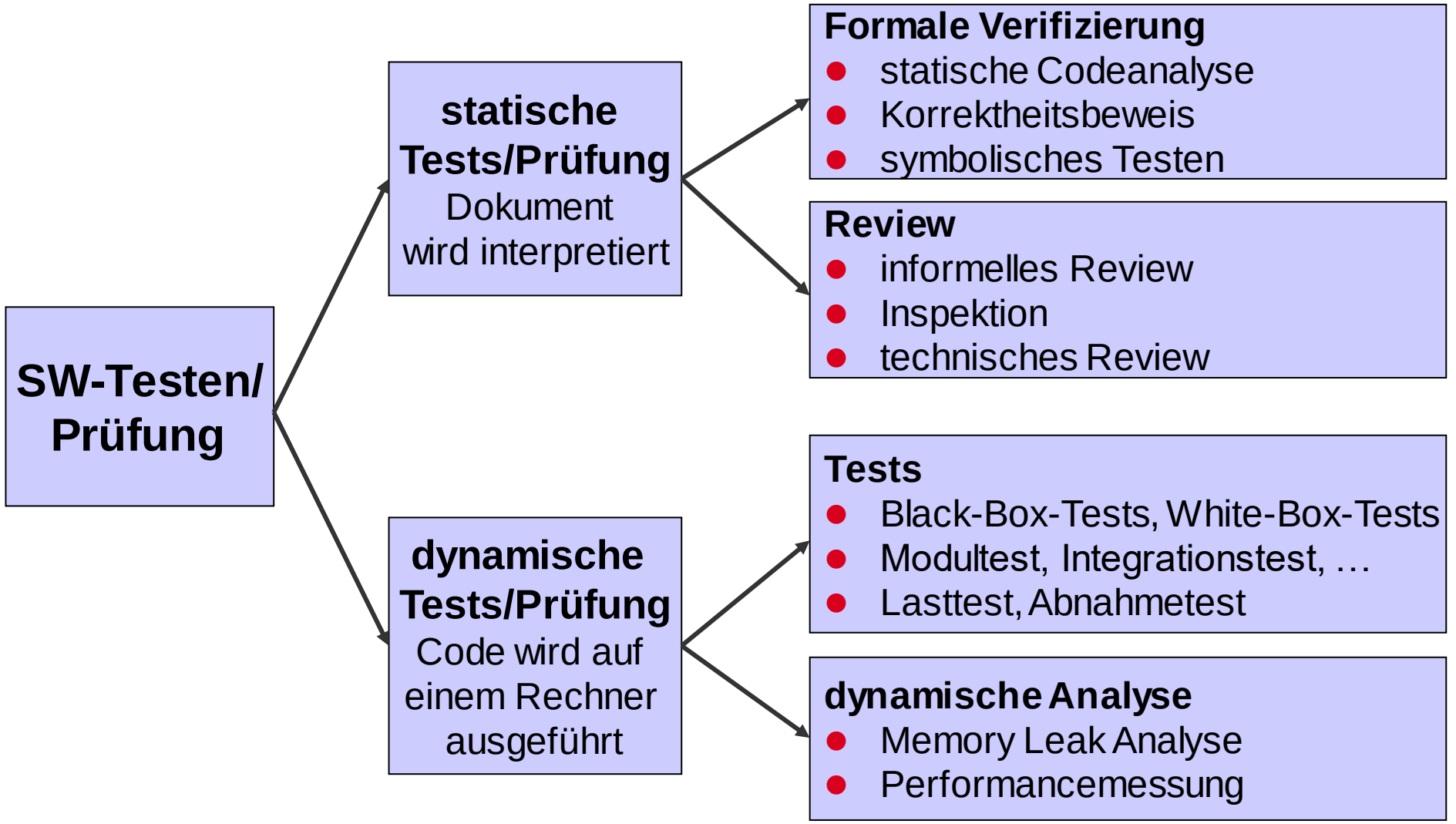
Trainingsbeispiel

philotech academy

Software & Hardware Certification, Support & Logistics, Safety & Reliability, Systems Engineering,
Structure Design & Flight Physics, Management Skills

- eingesetzt in fast allen Lebensbereichen, z. B.
  - Business-Software (z. B. Bankanwendungen)
  - Industrieprodukte (Flugzeuge, Fahrzeuge u. a.)
- unzureichende Qualität führt unter Umständen zu
  - Imageverlust
  - Geldverlust
  - Zeitverlust
  - Verletzungen
  - Tod

- NASA – Erdbeobachtungssatelliten 1979-1985
  - Ozonloch mehrere Jahre unerkannt
  - Ursache: Softwarefehler, die niedrigen Meßwerte der sich verändernden Ozonschicht wurden als Ausreißer behandelt und nicht in die Berechnung einbezogen
- ESA, Kourou, Franz. Guyana, 4. Juni 1996
  - Selbstzerstörung der Ariane 5 beim Jungfernflug 39 Sekunden nach dem Start
  - Ursache: Lageregelungssoftware aus Ariane 4 ohne Test übernommen
- ALG2 LL (Hartz IV) Software, Sept. 2005
  - automatische falsche Krankenkassenmeldungen in mehreren hunderttausend Fällen
  - Ursache: Softwarefehler
- Boing 737 Max, Okt. 2018 und März 2019
  - Absturz zweier Boeing 737 Max mit insgesamt 346 Toten
  - Ursache: Fehlerhafte Sensorwerte übermittelt an Stabilisierungssystem MCAS ("Maneuvering Characteristics Augmentation System")
  - Beim Take-off wurde Nase des Flugzeugs nach unten gedrückt, was Absturz zur folge hatte


SW-Testen/ Prüfung
statische Tests/Prüfung
Dokument wird interpretiert
dynamische Tests/Prüfung
Code wird auf einem Rechner ausgeführt
Formale Verifizierung
statische Codeanalyse
Korrektheitsbeweis
symbolisches Testen
Review
informelles Review
Inspektion
technisches Review
Tests
Black-Box-Tests, White-Box-Tests
Modultest, Integrationstest, …
Lasttest, Abnahmetest
dynamische Analyse
Memory Leak Analyse
Performancemessung

Typische Ziele des Testens (1)

allgemeine Ziele:

- Bewerten von Arbeitsergebnissen, wie:
  - Anforderungen
  - User-Stories
  - Architekturdesign
  - Code
- Verifizieren, ob alle spezifischen Anforderungen erfüllt sind
- Prüfen, ob das Testobjekt vollständig ist
- Validieren, dass Testobjekt so funktioniert, wie es Benutzer und andere Stakeholder erwarten
- Erzeugen von Vertrauen bezüglich des Qualitätsniveaus des Testobjektes
- Vorbeugen von Fehlerzuständen (in Zukunft)
- Aufdecken von Fehlerwirkungen und Fehlerzuständen, wodurch Risiken unzureichender Softwarequalität reduziert werden.
- Liefern von Informationen zur Entscheidungsfindung, insbesondere bezüglich des Qualitätsniveaus des Testobjektes

Typische Ziele des Testens (2)

- Konformität mit vertraglichen, rechtlichen oder regulatorischen Anforderungen oder Standards sicher stellen
- verifizieren der Konformität (compliance) des Testobjekts mit diesen Anforderungen oder Standards

aus Kontext der zu testenden Komponente oder des Systems, von der Teststufe und des Softwareentwicklungslebenszyklus-Modell können Ziele des Testens variieren, z. B.:

- Komponententest:
  - finden möglichst vieler Fehlerwirkungen, um frühzeitiges identifizieren und beheben von Fehlerzuständen zu ermöglichen
  - Erhöhung der Codeüberdeckung
- Abnahmetests:
  - Erwartete Funktionalität des Systems nachweisen
  - Stakeholdern Informationen über Risiko einer Systemfreigabe zu festgelegtem Zeitpunkt zu geben

1. Vollständiges Testen ist nicht möglich

> Beim Test von nicht-trivialen Testobjekten ist das Austesten, bei dem Tests für alle möglichen Eingabewerte und deren Kombinationen unter Berücksichtigung aller unterschiedlichen Vorbedingungen ausgeführt werden, nicht durchführbar.

Quelle: [Spillner 2019]

→ Tests können immer nur Stichproben sein

→ Testaufwand sollte anhand von Risiken und Prioritäten festgelegt werden

2. Testen zeigt die Anwesenheit von Fehlerzuständen

- mit Testen wird das Vorhandensein von Fehlerzuständen gezeigt
- Testen erbringt nicht den Beweis, dass keine Fehlerzustände mehr im Testobjekt enthalten sind, selbst wenn keine Fehlerwirkungen aufgedeckt werden
- ausreichendes Testen verringert die Wahrscheinlichkeit, dass noch unentdeckte Fehlerzustände im Testobjekt vorhanden sind

3. Frühes Testen spart Zeit und Geld

- Testaktivitäten sollten im Entwicklungszyklus bzw. Lebenszyklus des Systems oder der Software möglichst frühzeitig beginnen
- Testaktivitäten sollten immer definierte Ziele verfolgen

  → durch frühzeitiges Prüfen werden Fehler früh erkannt

  → kostenintensive Änderungen werden reduziert oder vermieden

4. Häufung von Fehlerzuständen

- Fehlerzustände sind im Testobjekt oft nicht gleichmäßig verteilt
- in wenigen Teilen sind oft die meisten Fehlerzustände

  → wo Fehlerwirkungen nachgewiesen wurden, finden sich oft noch weitere

  → Testen muss flexibel erfolgen, um auf diesen Umstand zu reagieren

5. Testwiederholungen haben keine Wirksamkeit (Pestizid-Paradoxon)

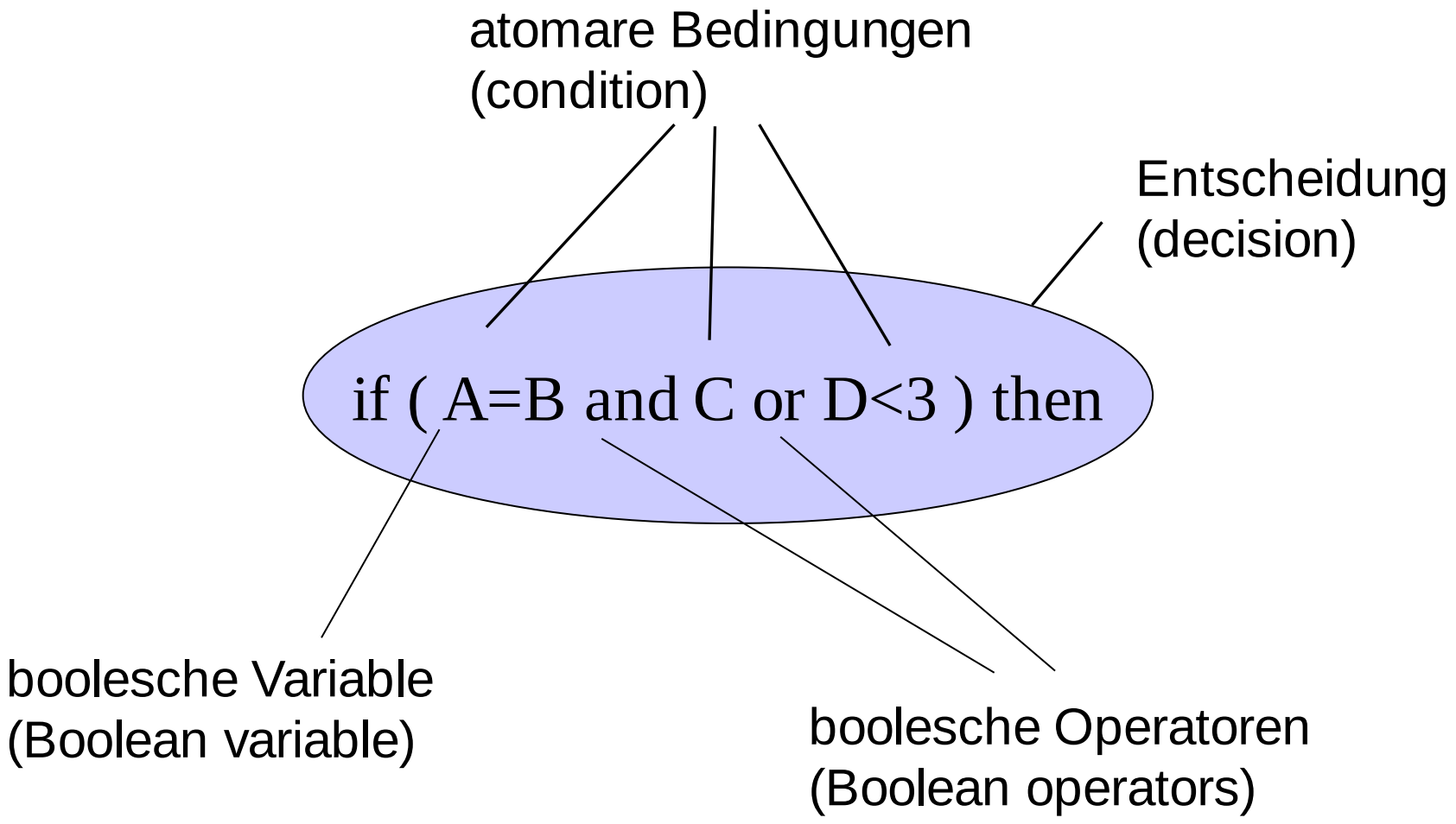
- wiederholtes Durchführen von gleichen Tests bringt keine neuen Informationen
- damit die Effektivität von Tests erhalten bleibt:
  - Tests regelmäßig prüfen und anpassen sowie
  - neue Tests erstellen für nicht geprüfte Teile/Konstellationen

6. Testen ist kontextabhängig

- gute Tests sind abhängig vom Einsatzgebiet und vom Umfeld des Testobjektes
  - → sicherheitskritische Systeme, z. B. im Luftfahrtbereich, müssen intensiver getestet werden als z. B. ein Online-Informationsportal
  - → Testen im agilen Projekten wird anders durchgeführt als Testen im sequenziellem Lebenszyklus

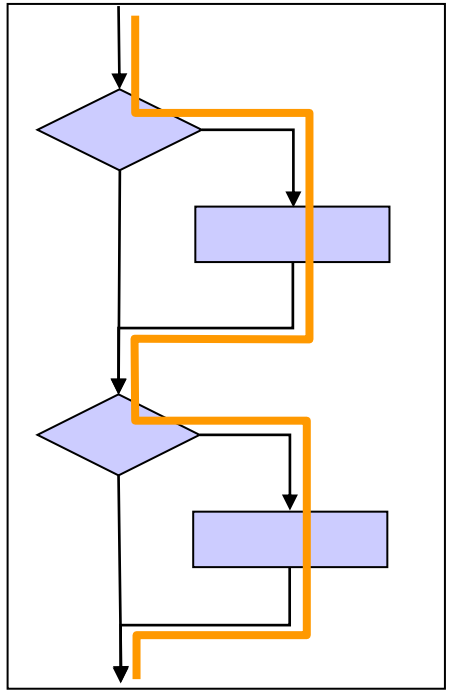
7. „System ohne Fehlerwirkungen“ bedeutet nicht immer „brauchbares System“

- Testen findet Fehlerwirkungen
- Testen bewirkt Fehlerbeseitigung
- Testen referenziert Anforderungen
  - → Testen prüft im Allgemeinen nicht, ob das System den Vorstellungen der Nutzer entspricht
  - → fehlerhafte Anforderungen werden mit dynamischem Test üblicherweise nicht gefunden
  - → z. B. Prototyping beugt solchen Problemen vor

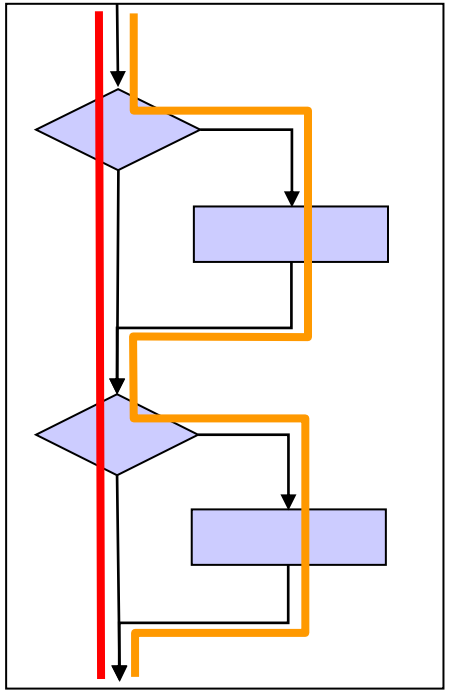

atomare Bedingungen
(condition)
Entscheidung
(decision)
if (A=B and C or D<3) then
boolesche Variable
(Boolean variable)
boolesche Operatoren
(Boolean operators)

Beispiele: strukturelle Überdeckung

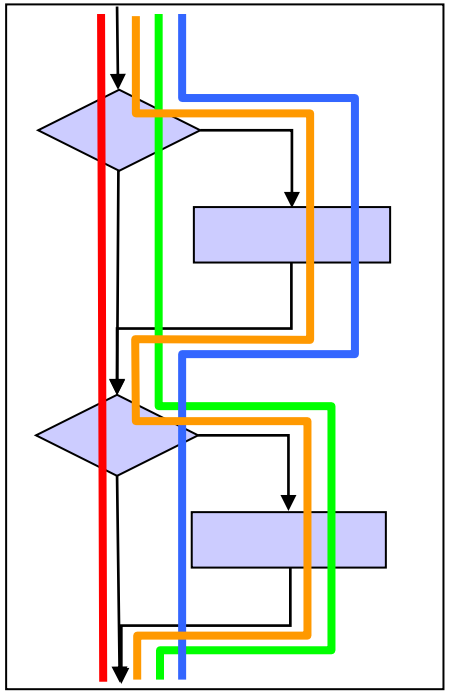
Anweisungs-
überdeckung



Entscheidungs-
überdeckung



Pfad-
überdeckung



Modified Condition/Decision Coverage:
Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken all possible outcomes at least once, every decision in the program has taken all possible outcomes at least once, and each condition in a decision has been shown to independently affect that decision's outcome. A condition is shown to independently affect a decision's outcome by:
(1) varying just that condition while holding fixed all other possible conditions, or
(2) varying just that condition while holding fixed all other possible conditions that could affect the outcome.

- Definition changed between DO-178B and DO-178C. Section "(2)" was added to deal with masking and short circuit problems

Example:
if(A || (B && C))

With Unique Cause MC / DC following tests can be used to show independent influence of conditions:
For A: {0,4}, {1,5}, {2,6}
For B: {1,3}
For C: {2,3}

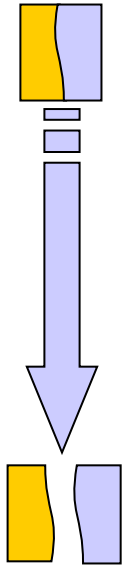
Nr.	A	B	C	Result
0	FALSE	FALSE	FALSE	FALSE
1	FALSE	FALSE	TRUE	FALSE
2	FALSE	TRUE	FALSE	FALSE
3	FALSE	TRUE	TRUE	TRUE
4	TRUE	FALSE	FALSE	TRUE
5	TRUE	FALSE	TRUE	TRUE
6	TRUE	TRUE	FALSE	TRUE
7	TRUE	TRUE	TRUE	TRUE

That results in two usable minimal test sets {1,2,3,5} and {1,2,3,6}.

With masking MC / DC for A all combinations of {0,1,2} x {4,5,6} can be used.
For B and C the combinations stay unchanged.
That results in three usable minimal test sets {1,2,3,5}, {1,2,3,6} and {1,2,3,4}.

Testorganisation: Unabhängigkeit

- Effektivität der Testarbeiten kann durch verschiedene Abstufungen der Unabhängigkeit, wie folgt, beeinflusst werden:



  - Keine unabhängigen Tester
    - Entwickler testen ihren eigenen Code
  - Unabhängige Entwickler oder Tester innerhalb des Entwicklungsteams
    - können Entwickler sein, die Arbeitsergebnisse ihrer Kollegen testen
  - unabhängiges Testteam oder Gruppe innerhalb des Unternehmens
  - unabhängige Tester innerhalb des Unternehmens
    - aus Fachbereich
    - aus der Gruppe der Anwender
    - mit Spezialisierungen auf bestimmte Testarten
  - unabhängige Tester aus einer externen Organisation (Inhouse / Outsourcing)
- auf verschiedenen Stufen können unterschiedliche Grade der Unabhängigkeit verwendet werden
- Entwickler sollten an mindestens einer Teststufe beteiligt sein bzw. sie durchführen um die Möglichkeit zu haben die Qualität ihrer eigenen Arbeit zu kontrollieren

Vor- und Nachteile des unabhängigen Testens

Vorteile

unabhängige Tester:

- sehen andere Fehler, denn sie haben unterschiedliche
  - Voreingenommenheit
  - Techn. Sichtweise
  - Hintergründe
- hinterfragen (implizite) Annahmen der Stakeholder, die schon während der Spezifikation oder während der Implementierung getroffen wurden
- können korrekt und objektiv über zu testendes System berichten, ohne das beauftragendes Unternehmen Druck auf sie ausüben kann

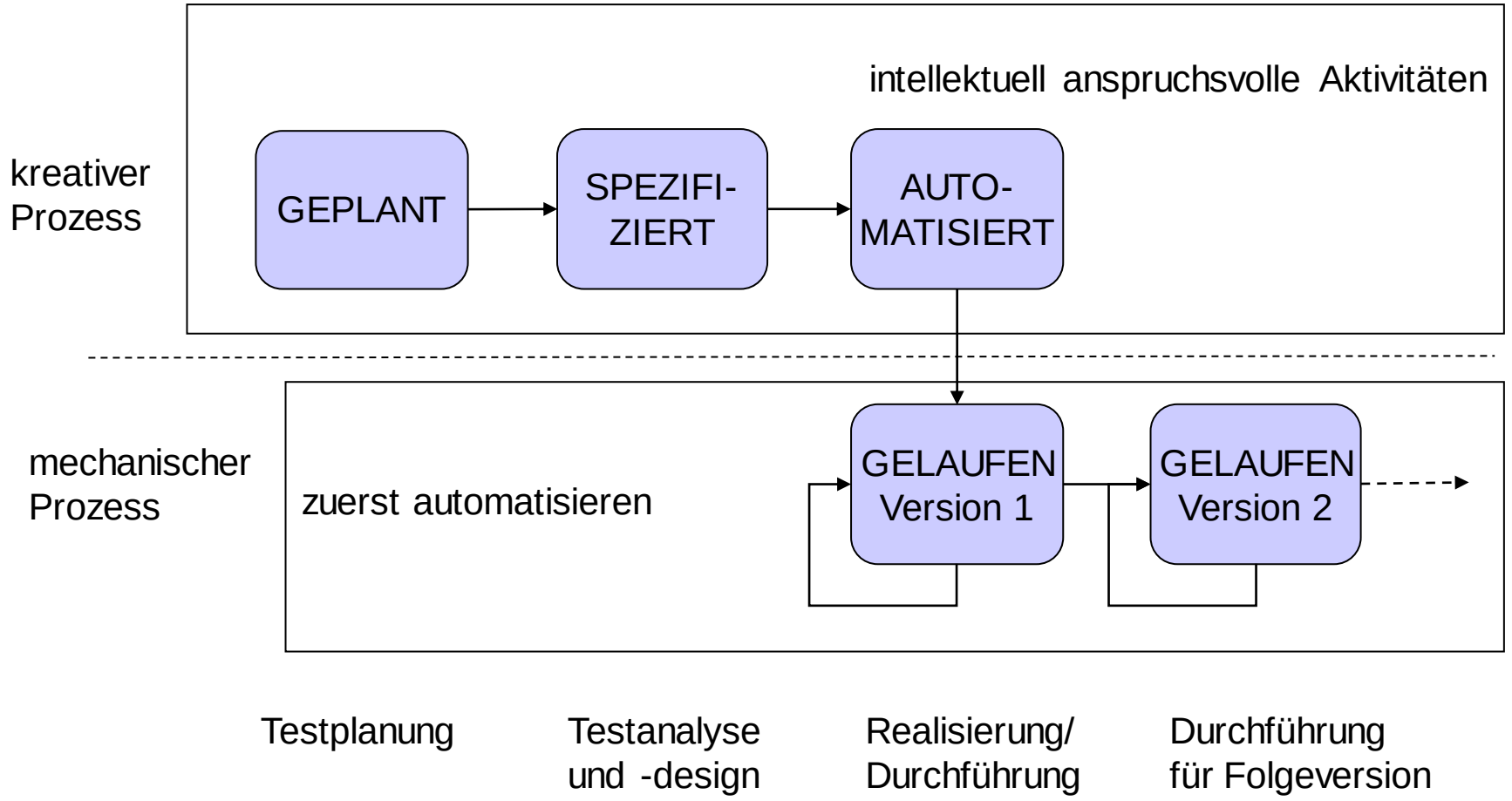
Nachteile

unabhängige Tests:

- Isolation vom Entwicklungsteam kann zu schlechter Kommunikation bzw. Zusammenarbeit führen
- Tester können als Engpass angesehen werden
- Entwickler können Verantwortungsgefühl für die Qualität ihrer Arbeit verlieren
- Wichtige Informationen können unter Umständen fehlen

- umsichtiges Management kann Nachteile vermeiden und Vorteile erhalten

Übersicht: Testlebenszyklus



Quelle: [Spillner 2019] S.296

Typen von Testwerkzeugen

- Testwerkzeuge sind Software oder Hardware, die eine oder mehrere Aktivitäten im Testprozess unterstützen
  - Planung und Steuerung
  - Spezifikation
  - Erstellung von Testdaten
  - Testdurchführung und Bewertung
- verschiedene Klassifikationen möglich, z. B. nach:
  - Zweck
  - Testrolle (z. B. Tester, Testentwickler, Testmanager)
  - Teststufe im allgemeinen V-Modell
  - Testart
  - Aktivität im Testprozess
  - Preis
  - kommerziell/kostenlos/Open Source/Shareware
  - verwendeter Technologie

intrusive Werkzeuge:
verändern oder beeinflussen das Testobjekt

- Instrumentierung ist das Einbringen von zusätzlichen Befehlen in das Testobjekt, z. B. zum Messen von Kommunikation oder Überdeckung von Strukturen
- Verwendung von intrusiven Werkzeugen beeinflusst unter Umständen
  - das Zeitverhalten und
  - die strukturelle Abdeckung von z. B. Anweisungen und Entscheidungen im Programmcode
- intrusives Verhalten kann diesen „Untersuchungseffekt“ hervorrufen

Wir freuen uns auf Ihre Schulungsteilnahme

philotech academy

Software & Hardware Certification, Support & Logistics, Safety & Reliability, Systems Engineering, Structure Design & Flight Physics, Management Skills

Für weiterführende Informationen besuchen sie unsere Hompage unter: https://www.philotech.net/

Oder kontaktieren sie uns direkt: certifiedtester@philotech.de