Willkommen zur Vorlesung
Softwarekonstruktion
im Wintersemester 2012 / 2013

Prof. Dr. Jan Jürjens

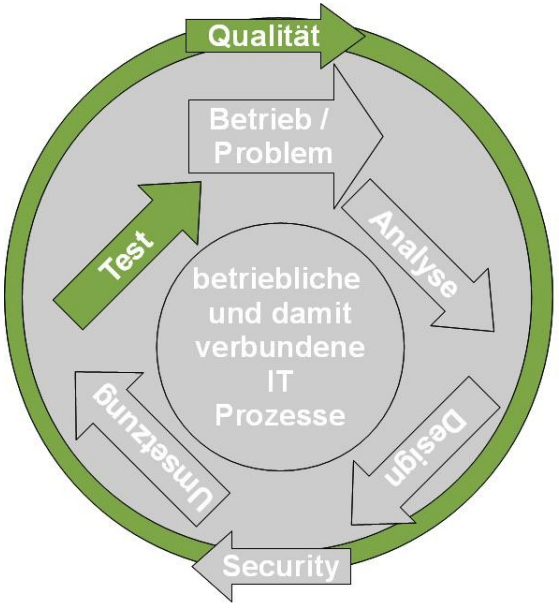
TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

2.1 Grundlagen des Softwaretestens

Basierend auf dem Foliensatz
„Basiswissen Softwaretest - Certified Tester“ des „German Testing Board“
(nach Certified Tester Foundation Level Syllabus, deutschsprachige Ausgabe, Version 2011) (mit freundlicher Genehmigung)

Einordnung
Grundlagen Testen

- Qualitätsmanagement
- Testen
  - Einführung
  - Grundlagen Softwaretesten
  - Testen im Softwarelebenszyklus
  - Statischer Test
  - Black-Box-Test
  - White-Box-Test
  - Test-Management
  - Testwerkzeuge
- Modellgetriebene SW-Entwicklung



2.1: Grundlagen des Softwaretestens

Grundlagen des Softwaretestens

- Begriffe und Motivation
- Grundsätze des Softwaretestens
- Fundamentaler Testprozess
- Testfälle, Sollwerte und Testorakel
- Psychologie des Testens
- Ethik des Testens

Was gilt als Fehler oder Mangel

- Eine Situation kann nur dann als fehlerhaft eingestuft werden, wenn vorab festgelegt wurde, wie die erwartete, korrekte, also nicht fehlerhafte Situation aussehen soll.
- Ein **Fehler**
  ist somit die Nichterfüllung einer festgelegten Anforderung, eine Abweichung zwischen dem Ist-Verhalten (während der Ausführung der Tests oder des Betriebs festgestellt) und dem Soll-Verhalten (in der Spezifikation oder den Anforderungen festgelegt).
- Ein **Mangel**
  liegt vor, wenn eine gestellte Anforderung oder eine berechtigte Erwartung in Bezug auf einen beabsichtigten Gebrauch nicht angemessen erfüllt wird. Ein Mangel ist z.B. die Beeinträchtigung der Verwendbarkeit bei gleichzeitiger Erfüllung der Funktionalität oder die Nichterfüllung einer angemessenen Erwartung.

Ursachenkette für Fehler

- Jeder Fehler oder Mangel ist seit dem Zeitpunkt der Fertigstellung in der Software vorhanden. Er kommt jedoch erst bei der Ausführung der Software zum Tragen.
- Beschreibung dieses Sachverhalts als **Fehlerwirkung** (*failure*) (auch als Fehlfunktion, äußerer Fehler, Ausfall bezeichnet).
- Ursache einer Fehlerwirkung: **Fehlerzustand** (*fault*) in der Software (auch als Defekt, innerer Fehler bezeichnet).
- Ursache eines Fehlerzustands: **Fehlhandlung** (*error*) einer Person.
- Beachte **Fehlermaskierung**: »Ein Umstand, bei dem ein Fehlerzustand die Aufdeckung eines anderen verhindert« [IEEE 610].

Angelehnt an: DIN 66271:1995
Informationstechnik - Software-Fehler und ihre Beurteilung durch Lieferanten und Kunden, Beuth Verlag, Berlin, 1995

Fehlerhandlung – Definition

- **Fehlhandlung *(error)***
  1. Die menschliche Handlung (des Entwicklers), die zu einem Fehlerzustand in der Software führt.
  2. Eine menschliche Handlung (des Anwenders), die ein unerwünschtes Ergebnis (im Sinne von Fehlerwirkung) zur Folge hat (Fehlbedienung).
  3. Unwissentlich, versehentlich oder absichtlich ausgeführte Handlung oder Unterlassung, die unter gegebenen Umständen (Aufgabenstellung, Umfeld) dazu führt, dass eine geforderte Funktion eines Produkts beeinträchtigt ist.

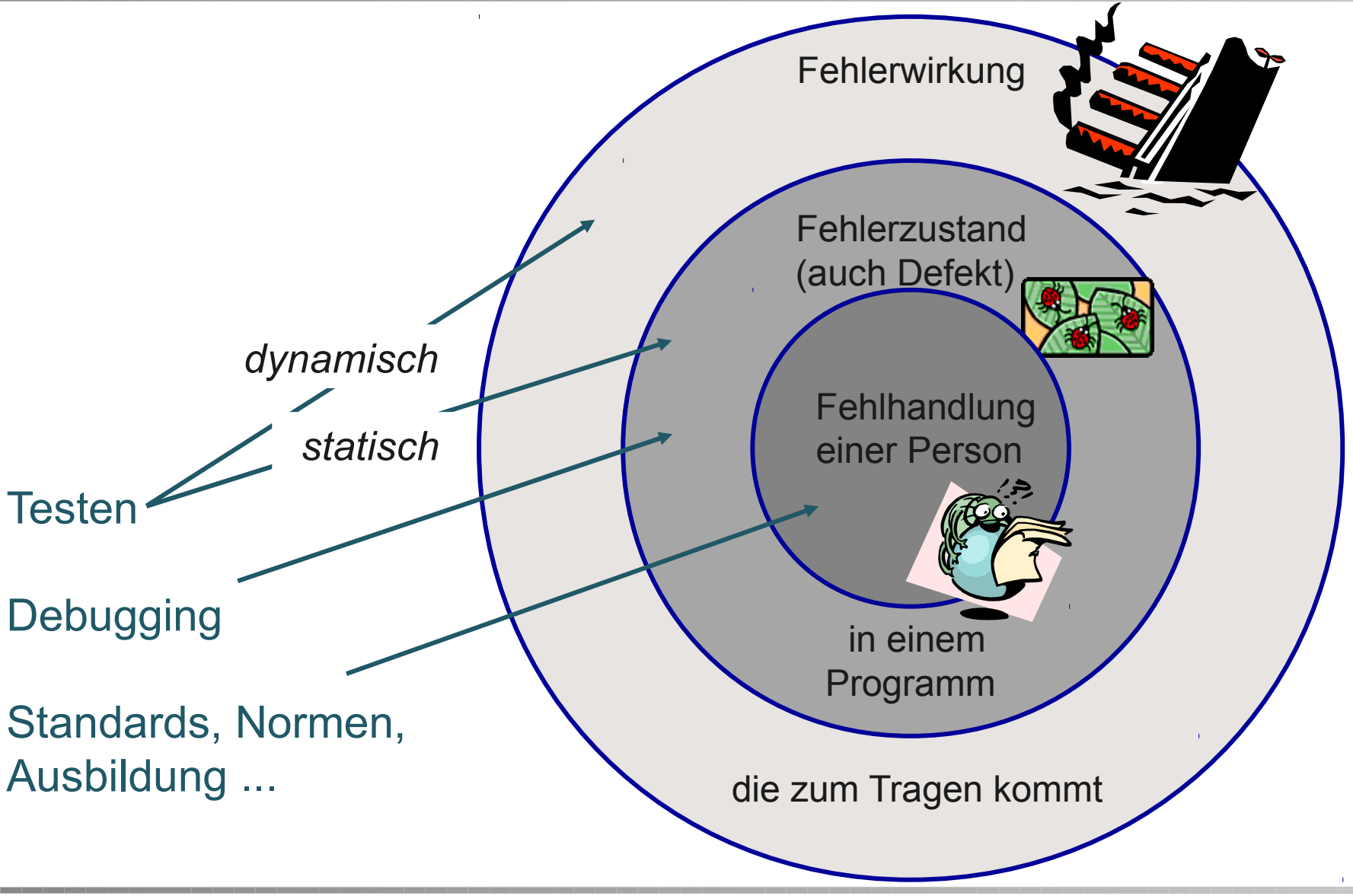
Fehlerzustand / -wirkung – Definition

- **Fehlerzustand / Defekt (fault)**
  1. Inkorrektes Teilprogramm (z.B. mit inkorrekter Anweisung oder Datendefinition), das Ursache für eine Fehlerwirkung sein kann.
  2. Zustand eines (Software-)Produkts oder einer seiner Komponenten, der unter spezifischen Bedingungen (z.B. bei einer hohen Belastung) eine geforderte Funktion des Produkts beeinträchtigen kann bzw. zu einer Fehlerwirkung führt.

- **Fehlerwirkung (failure)**
  1. Wirkung eines Fehlerzustands, die bei der Ausführung eines Programms (Testobjekt) nach »außen« in Erscheinung tritt.
  2. Abweichung zwischen (spezifizierten) Soll-Wert und (beobachtetem) Ist-Wert (bzw. Soll- und Ist-Verhalten).
  3. Abweichung einer Komponente/eines Systems von der erwarteten Lieferung, Leistung oder dem Ergebnis [nach Fenton].

- Anmerkung: Fehlerwirkungen können (selten) auch durch Höhenstrahlung, elektromagnetische Felder oder Hardwarefehler hervorgerufen werden.
- Anmerkung: So kann eine Fehlerwirkung zu einer langsamen Ausführung, zu inkorrekten Ausgaben oder zu einem Abbruch der Ausführung führen.

Entstehen von Fehlern und Gegenmaßnahmen



Testen – Definition und Ziele

- Prozess, der sich (sowohl statisch als auch dynamisch) mit der Planung, Vorbereitung und Bewertung einer Software und den hierzu in Beziehung stehenden Arbeitsergebnissen befasst, um die Software mit dem Ziel zu bewerten,
  - dass diese allen festgelegten Anforderungen und
  - ihren Zweck erfüllt und
  - um etwaige Fehlerzustände zu finden.

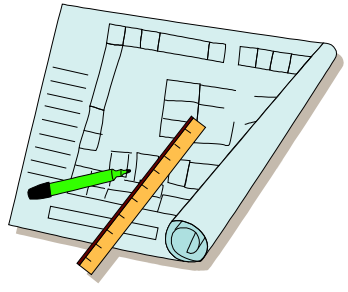
Validierung und Verifizierung – Definition

- **Validierung**

  Prüfung, ob ein Entwicklungsergebnis die individuellen Anforderungen bezüglich einer speziellen beabsichtigten Nutzung erfüllt.
  - Haben wir das **richtige System** realisiert?

- **Verifizierung**

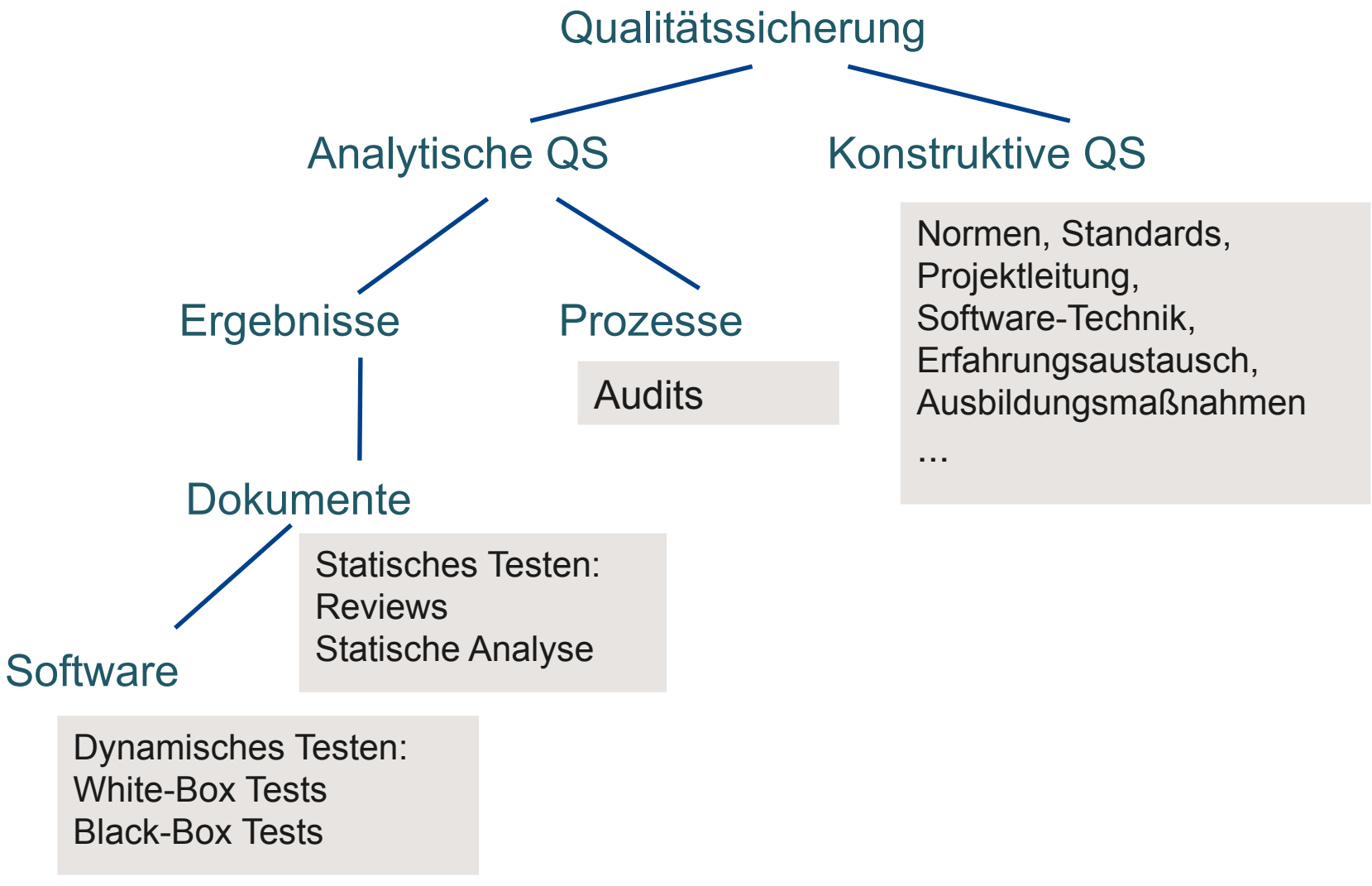
  Prüfung, ob die Ergebnisse einer Entwicklungsphase die Vorgaben der Phaseneingangs-Dokumente erfüllen.
  - Haben wir das System **richtig realisiert**?



Testen und Qualität

- Testen misst die Qualität z.B. anhand der Anzahl gefundener Fehlerwirkungen.
- Testen erhöht indirekt die Qualität, da Fehler(zustände) vor der Auslieferung entdeckt und korrigiert werden können.
- Testen erhöht indirekt die Prozessqualität, da Fehler dokumentiert, analysiert und damit Fehlhandlungen in Zukunft vermieden werden können.
- Testen erhöht das Vertrauen in die Qualität des Systems, wenn wenige oder keine Fehler gefunden werden.

Qualitätssicherung QS



Übersicht Prüfverfahren

- **verifizierend**
  - Programmverifikation
  - symbolische Programmausführung
- **analysierend**
  - Metriken
  - Graphen, Tabellen
  - Anomalienanalyse
- **testend**
  - statisch
    - statische Programm-Analysatoren
    - Reviews
    - Audits
  - dynamisch
    - strukturorientiert
      - kontrollflußbezogen
        - Anweisungsüberdeckung
        - Zweigüberdeckung
        - Bedingungsüberdeckung
          - einfache
          - minimale mehrfache
          - mehrfache
        - Boundary-interior Pfadtest
        - Pfadüberdeckung
      - datenflußbezogen
        - Defs/Uses-Kriterien
          - all defs-Kriterien
          - all uses-Kriterien
          - all du-paths-Kriterien
        - Datenkontext-Überdeckung
    - funktional
      - funktionale Äquivalenzklassenbildung
      - Ursache-Wirkungs-Analyse
    - diversifizierend
      - Mutationen-Test
      - Perturbationen-Test
      - Back-to-Back-Test
    - Zufallstest
    - Test spezieller Werte
    - Grenzwertanalyse

Analytisches Qualitätsmanagement

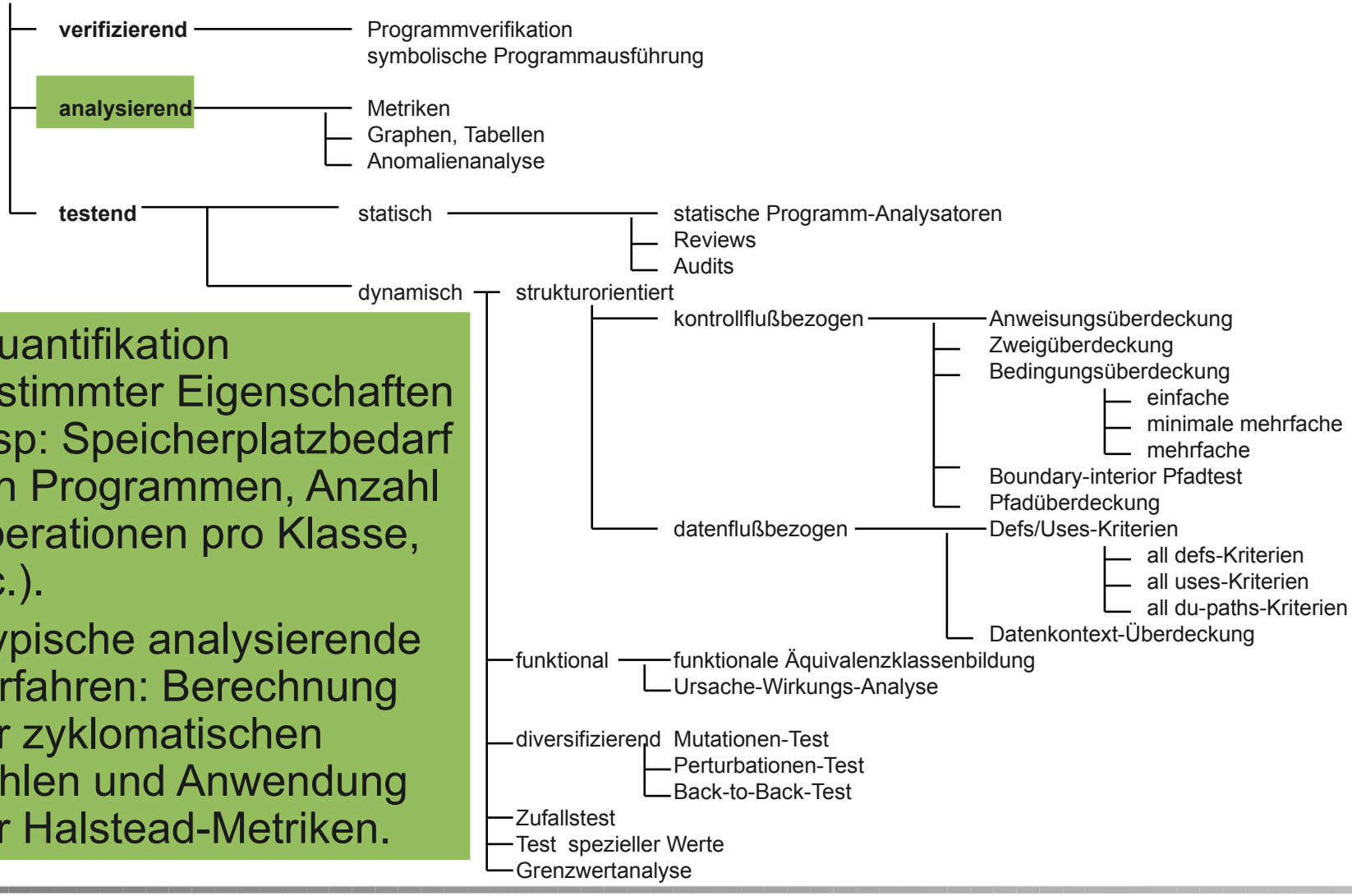
Übersicht Prüfverfahren

- **verifizierend**
  - Programmverifikation
  - symbolische Programmausführung
- **analysierend**
  - Metriken
  - Graphen, Tabellen
  - Anomalienanalyse
- **testend**
  - statisch
    - statische Programm-Analysatoren
    - Reviews
    - Audits
  - dynamisch
    - strukturorientiert
      - kontrollflußbezogen
        - Anweisungsüberdeckung
        - Zweigüberdeckung
        - Bedingungsüberdeckung
          - einfache
          - minimale mehrfache
          - mehrfache
        - Boundary-interior Pfadtest
        - Pfadüberdeckung
      - datenflußbezogen
        - Defs/Uses-Kriterien
          - all defs-Kriterien
          - all uses-Kriterien
          - all du-paths-Kriterien
        - Datenkontext-Überdeckung
    - funktional
      - funktionale Äquivalenzklassenbildung
      - Ursache-Wirkungs-Analyse
    - diversifizierend
      - Mutationen-Test
      - Perturbationen-Test
      - Back-to-Back-Test
    - Zufallstest
    - Test spezieller Werte
    - Grenzwertanalyse

Beweis der Korrektheit eines Programms (Hinweis: Korrektheit wird gegen Anforderungsdokument und Spezifiktion geprüft!)

Analytisches Qualitätsmanagement

Übersicht Prüfverfahren



- Quantifikation bestimmter Eigenschaften (Bsp: Speicherplatzbedarf von Programmen, Anzahl Operationen pro Klasse, etc.).
- Typische analysierende Verfahren: Berechnung der zyklomatischen Zahlen und Anwendung der Halstead-Metriken.

Analytisches Qualitätsmanagement

Übersicht Prüfverfahren

- **verifizierend**
  - Programmverifikation
  - symbolische Programmausführung
- **analysierend**
  - Metriken
  - Graphen, Tabellen
  - Anomalienanalyse
- **testend**
  - statisch
    - statische Programm-Analysatoren
    - Reviews
    - Audits
  - dynamisch
    - strukturorientiert
      - kontrollflußbezogen
        - Anweisungsüberdeckung
        - Zweigüberdeckung
        - Bedingungsüberdeckung
          - einfache
          - minimale mehrfache
          - mehrfache
        - Boundary-interior Pfadtest
        - Pfadüberdeckung
      - datenflußbezogen
        - Defs/Uses-Kriterien
          - all defs-Kriterien
          - all uses-Kriterien
          - all du-paths-Kriterien
        - Datenkontext-Überdeckung
    - funktional
      - funktionale Äquivalenzklassenbildung
      - Ursache-Wirkungs-Analyse
    - diversifizierend
      - Mutationen-Test
      - Perturbationen-Test
      - Back-to-Back-Test
    - Zufallstest
    - Test spezieller Werte
    - Grenzwertanalyse

- Aufdecken von Fehlern
- Tests verschiedener Art stellen somit einen Bestandteil der Qualitätssicherung dar.
- In der Praxis sind Tests die am weitesten verbreitete Qualitätssicherungsmaßnahme

Analytisches Qualitätsmanagement

Übersicht Prüfverfahren

- **verifizierend**
  - Programmverifikation
  - symbolische Programmausführung
- **analysierend**
  - Metriken
  - Graphen, Tabellen
  - Anomalienanalyse
- **testend**
  - statisch
    - statische Programm-Analysatoren
    - Reviews
    - Audits
  - dynamisch
    - strukturorientiert
      - kontrollflußbezogen
        - Anweisungsüberdeckung
        - Zweigüberdeckung
        - Bedingungsüberdeckung
          - einfache
          - minimale mehrfache
          - mehrfache
        - Boundary-interior Pfadtest
        - Pfadüberdeckung
      - datenflußbezogen
        - Defs/Uses-Kriterien
          - all defs-Kriterien
          - all uses-Kriterien
          - all du-paths-Kriterien
        - Datenkontext-Überdeckung
    - funktional
      - funktionale Äquivalenzklassenbildung
      - Ursache-Wirkungs-Analyse
    - diversifizierend
      - Mutationen-Test
      - Perturbationen-Test
      - Back-to-Back-Test
    - Zufallstest
    - Test spezieller Werte
    - Grenzwertanalyse

- Statische Verfahren sind dadurch gekennzeichnet, dass die Software NICHT ausgeführt wird (Reviews, statische Analyse)

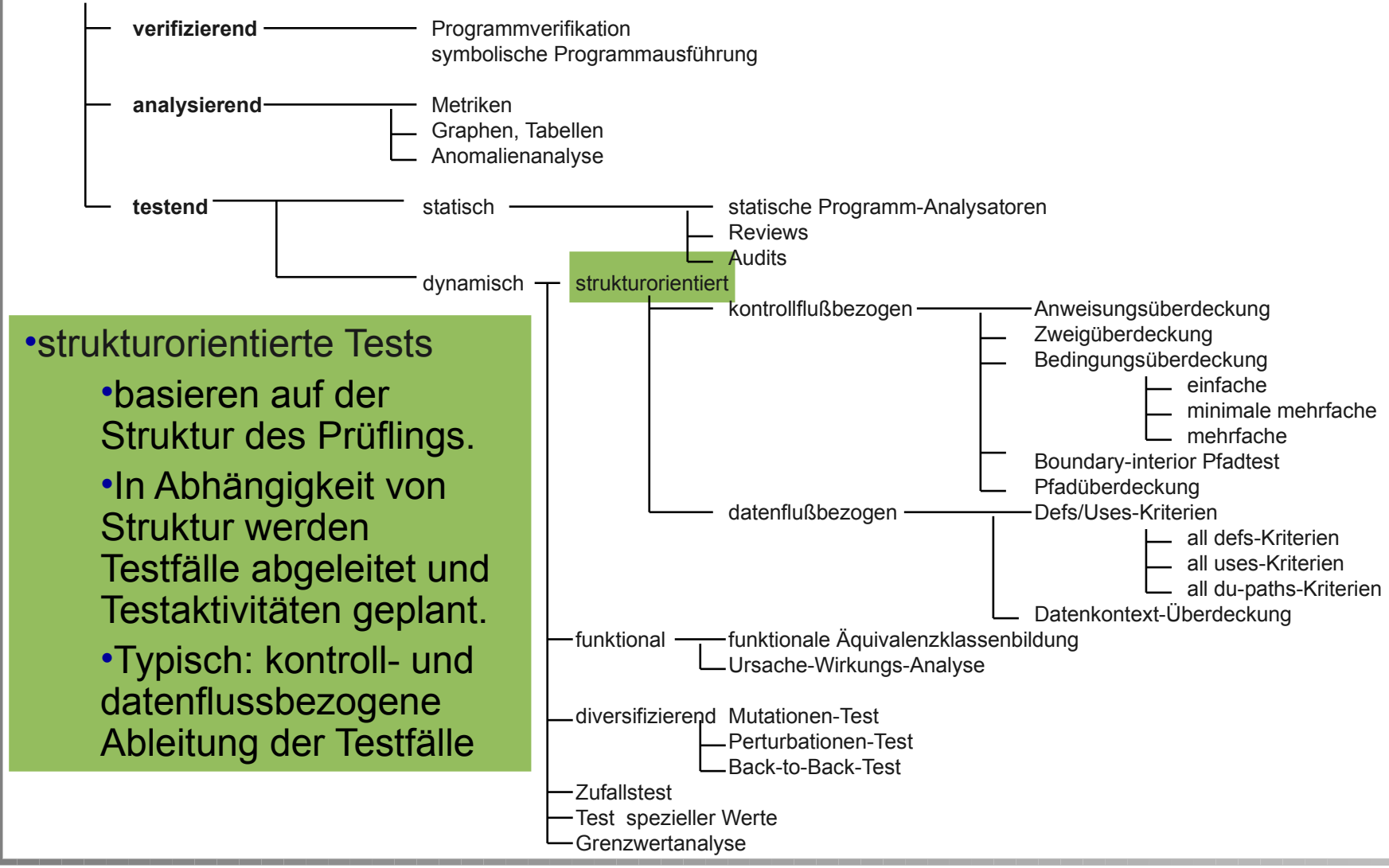
Übersicht Prüfverfahren

- **verifizierend**
  - Programmverifikation
  - symbolische Programmausführung
- **analysierend**
  - Metriken
  - Graphen, Tabellen
  - Anomalienanalyse
- **testend**
  - statisch
    - statische Programm-Analysatoren
    - Reviews
    - Audits
  - dynamisch
    - strukturorientiert
      - kontrollflußbezogen
        - Anweisungsüberdeckung
        - Zweigüberdeckung
        - Bedingungsüberdeckung
          - einfache
          - minimale mehrfache
          - mehrfache
        - Boundary-interior Pfadtest
        - Pfadüberdeckung
      - datenflußbezogen
        - Defs/Uses-Kriterien
          - all defs-Kriterien
          - all uses-Kriterien
          - all du-paths-Kriterien
        - Datenkontext-Überdeckung
    - funktional
      - funktionale Äquivalenzklassenbildung
      - Ursache-Wirkungs-Analyse
    - diversifizierend
      - Mutationen-Test
      - Perturbationen-Test
      - Back-to-Back-Test
    - Zufallstest
    - Test spezieller Werte
    - Grenzwertanalyse

- Dynamische Verfahren setzen zur Fehlererkennung die Ausführbarkeit des Prüflings voraus

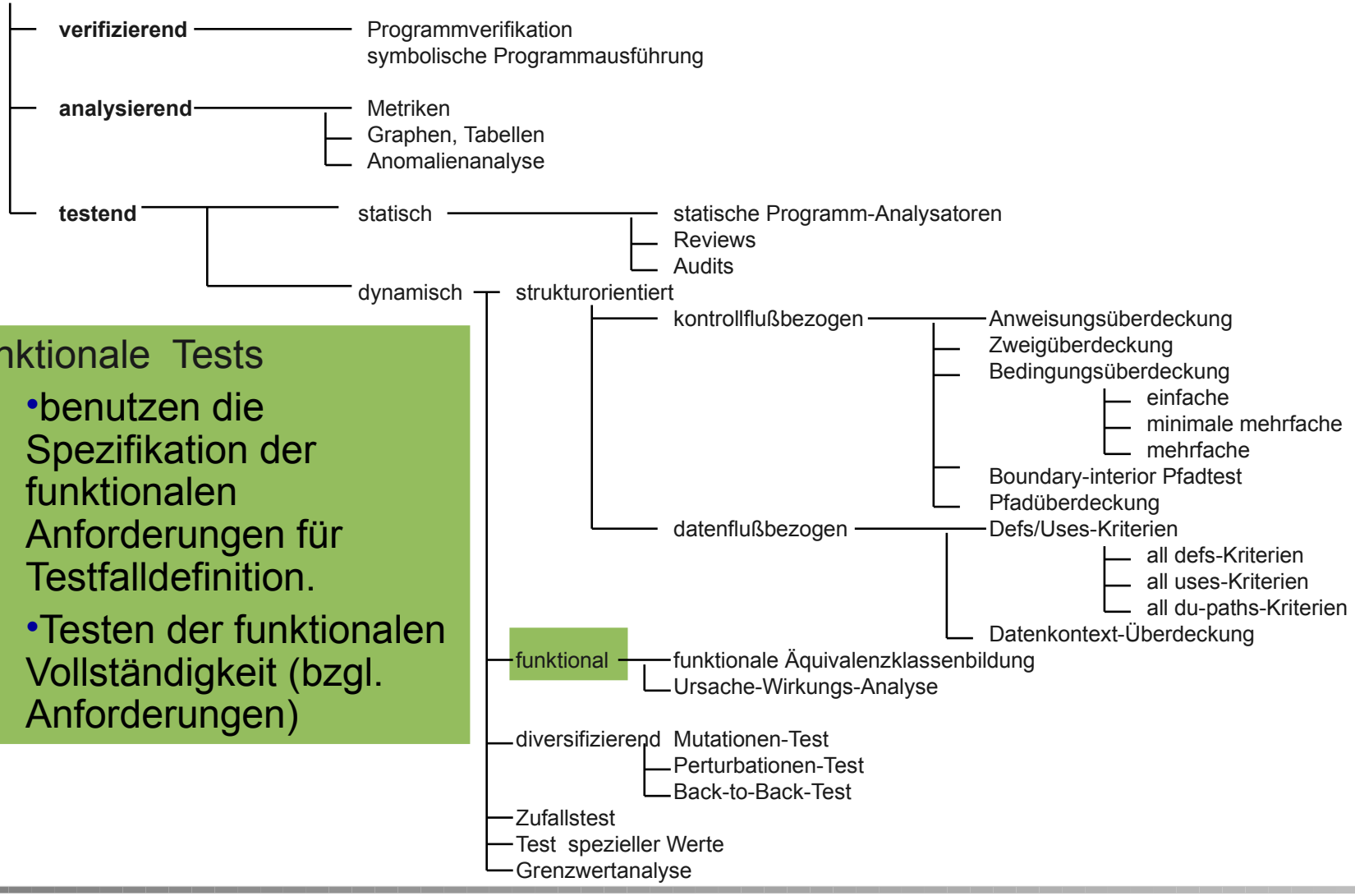
Analytisches Qualitätsmanagement

Übersicht Prüfverfahren



- strukturorientierte Tests
  - basieren auf der Struktur des Prüflings.
  - In Abhängigkeit von Struktur werden Testfälle abgeleitet und Testaktivitäten geplant.
  - Typisch: kontroll- und datenflussbezogene Ableitung der Testfälle

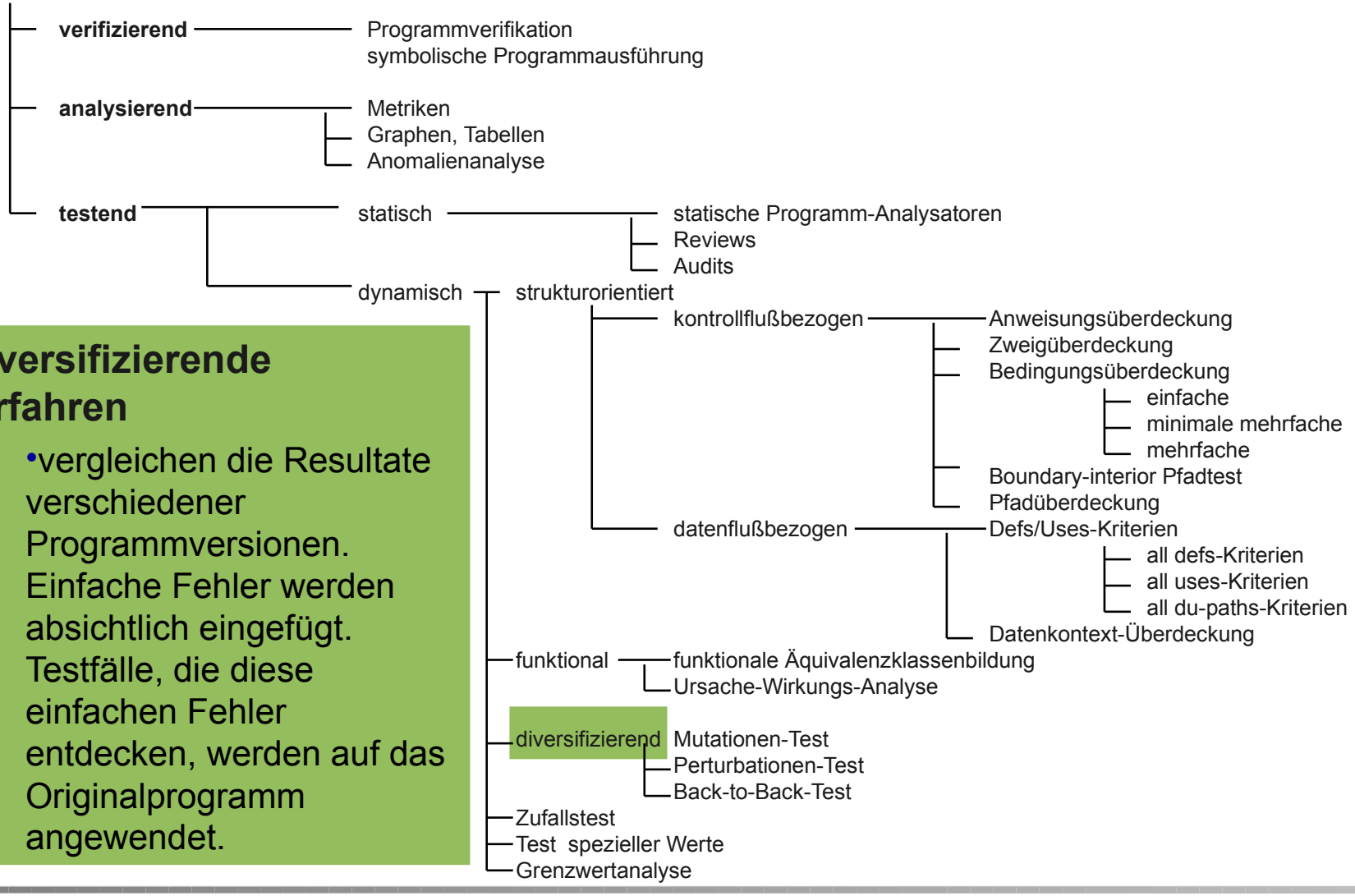
Übersicht Prüfverfahren



- funktionale Tests
  - benutzen die Spezifikation der funktionalen Anforderungen für Testfalldefinition.
  - Testen der funktionalen Vollständigkeit (bzgl. Anforderungen)

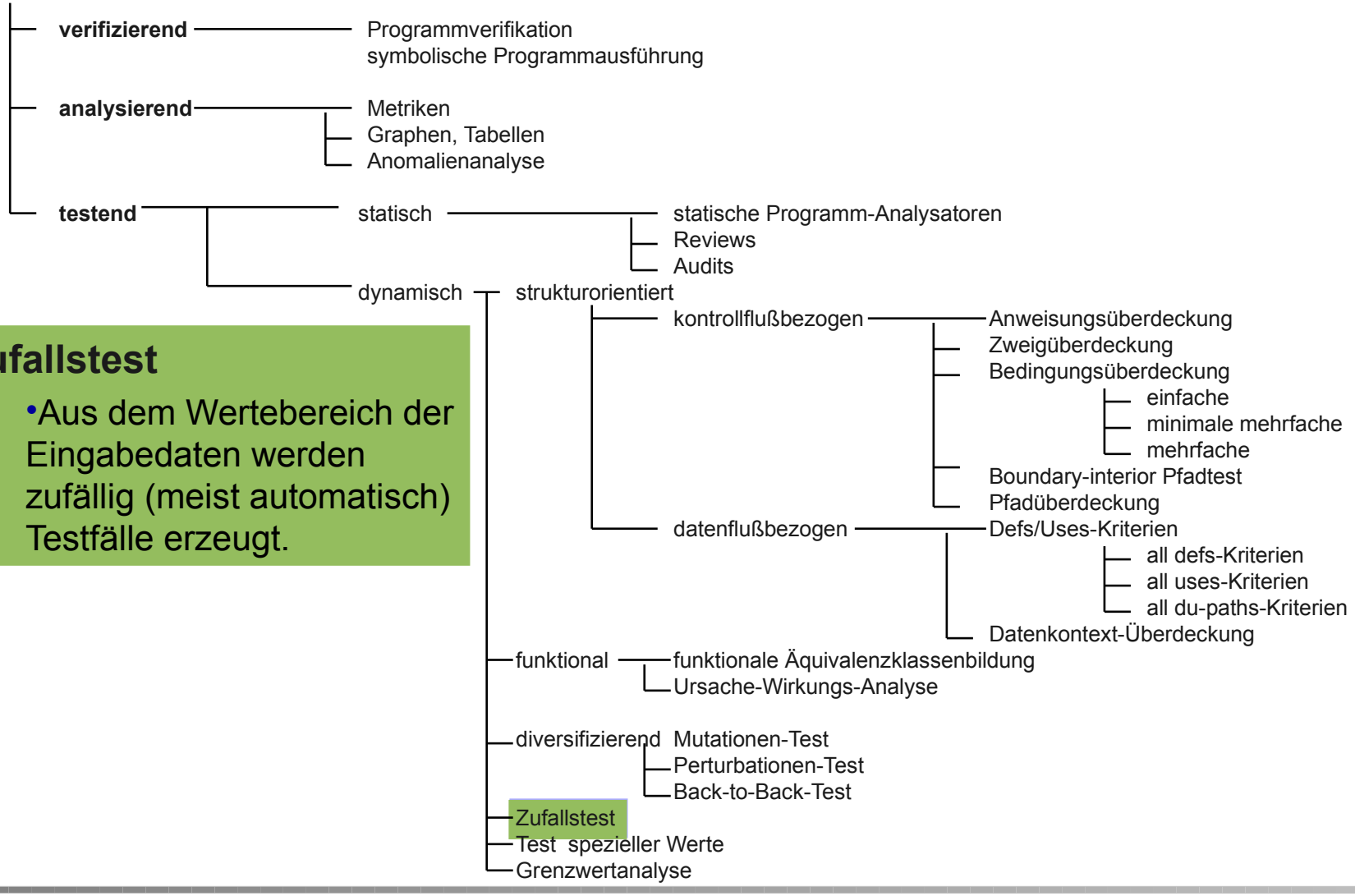
Analytisches Qualitätsmanagement

Übersicht Prüfverfahren



- **diversifizierende Verfahren**
  - vergleichen die Resultate verschiedener Programmversionen. Einfache Fehler werden absichtlich eingefügt. Testfälle, die diese einfachen Fehler entdecken, werden auf das Originalprogramm angewendet.

Übersicht Prüfverfahren



- **Zufallstest**
  - Aus dem Wertebereich der Eingabedaten werden zufällig (meist automatisch) Testfälle erzeugt.

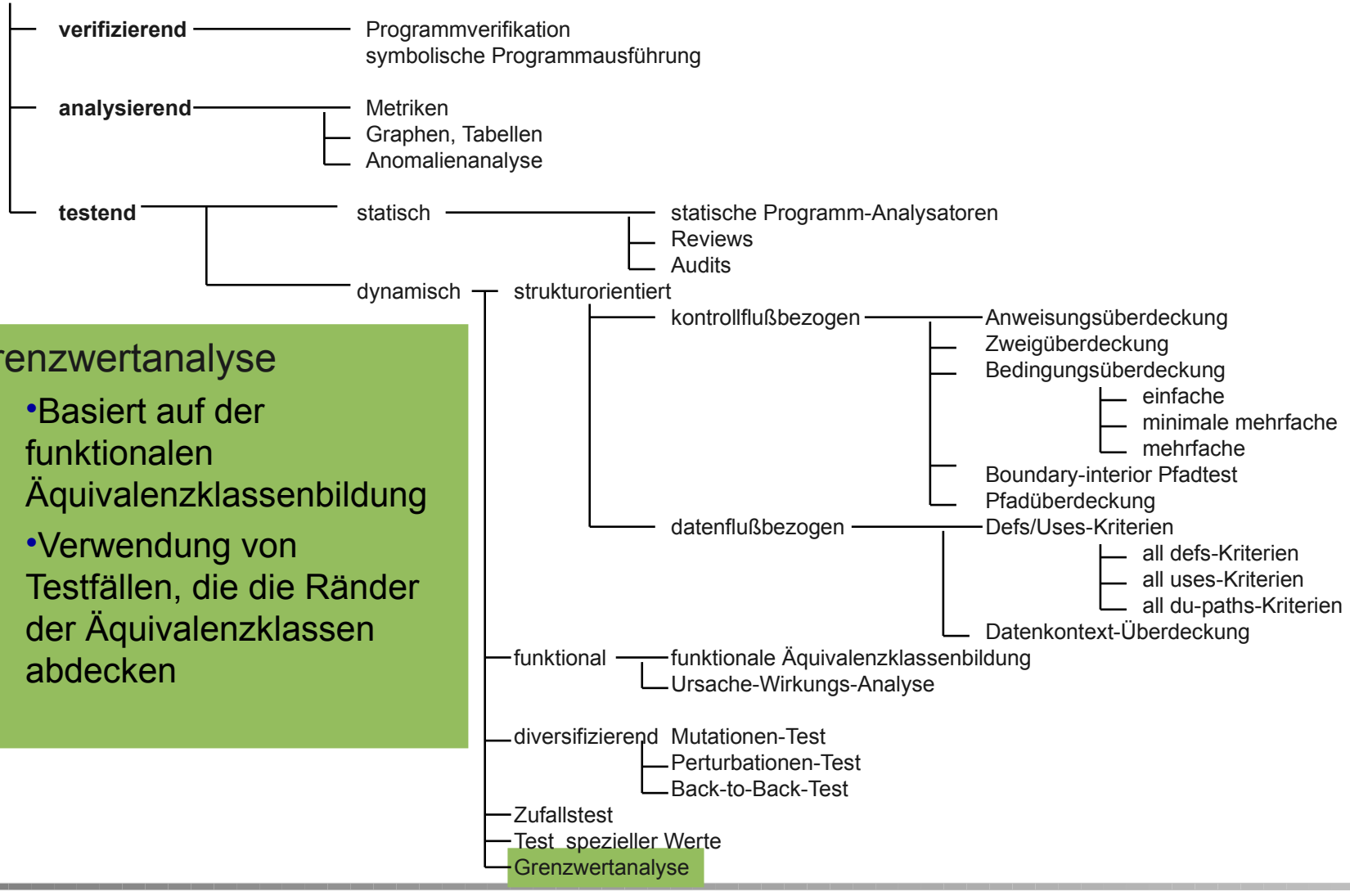
Analytisches Qualitätsmanagement

Übersicht Prüfverfahren

- **verifizierend**
  - Programmverifikation
  - symbolische Programmausführung
- **analysierend**
  - Metriken
  - Graphen, Tabellen
  - Anomalienanalyse
- **testend**
  - statisch
    - statische Programm-Analysatoren
    - Reviews
    - Audits
  - dynamisch
    - strukturorientiert
      - kontrollflußbezogen
        - Anweisungsüberdeckung
        - Zweigüberdeckung
        - Bedingungsüberdeckung
          - einfache
          - minimale mehrfache
          - mehrfache
        - Boundary-interior Pfadtest
        - Pfadüberdeckung
      - datenflußbezogen
        - Defs/Uses-Kriterien
          - all defs-Kriterien
          - all uses-Kriterien
          - all du-paths-Kriterien
        - Datenkontext-Überdeckung
    - funktional
      - funktionale Äquivalenzklassenbildung
      - Ursache-Wirkungs-Analyse
    - diversifizierend
      - Mutationen-Test
      - Perturbationen-Test
      - Back-to-Back-Test
    - Zufallstest
    - Test spezieller Werte
    - Grenzwertanalyse

- **Test spezieller Werte**
  - Auswahl der Testdaten auf Basis von Erfahrung

Übersicht Prüfverfahren



- Grenzwertanalyse
  - Basiert auf der funktionalen Äquivalenzklassenbildung
  - Verwendung von Testfällen, die die Ränder der Äquivalenzklassen abdecken

Grundlagen des Softwaretestens

- Begriffe und Motivation
- **Grundsätze des Softwaretestens**
- Fundamentaler Testprozess
- Testfälle, Sollwerte und Testorakel
- Psychologie des Testens
- Ethik des Testens

Grundsätze zum Testen (1)

In den letzten 40 Jahren haben sich folgende Grundsätze zum Testen herauskristallisiert und können somit als Leitlinien dienen:

- Grundsatz 1: »**Testen zeigt die Anwesenheit von Fehlern**«

Mit Testen wird die Anwesenheit von Fehlerwirkungen nachgewiesen. Testen kann nicht zeigen, dass keine Fehlerzustände im Testobjekt vorhanden sind !

»Program testing can be used to show the presence of bugs, but never to show their absence!« Edsger W. Dijkstra, 1970

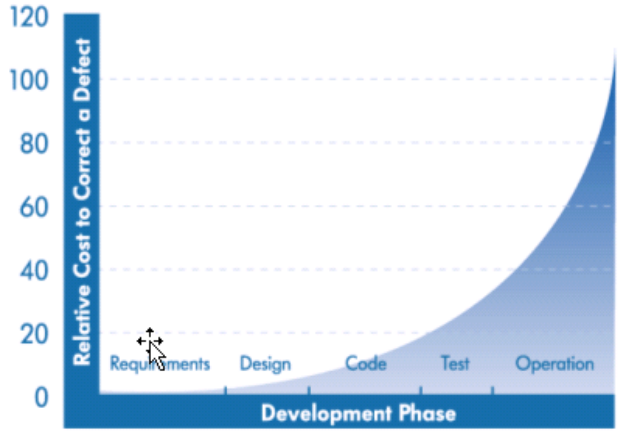
- Grundsatz 2: »**Vollständiges Testen ist nicht möglich**«

Vollständiges (erschöpfender) Testen – Austesten – ist nicht möglich.
(s. Folien in Abschnitt 4.0 zum Austesten)

Grundsätze zum Testen (2)

- Grundsatz 3: »**Mit dem Testen frühzeitig beginnen**«

  Testen ist keine späte Phase in der Softwareentwicklung, es soll damit so früh wie möglich begonnen werden. Durch frühzeitiges Prüfen (z.B. Reviews) parallel zu den konstruktiven Tätigkeiten werden Fehler(zustände) früher erkannt und somit Kosten gesenkt.



[„Finding and fixing a problem late in the development process can be 100 times more expensive than finding and fixing it during the requirement or design phase“. Barry Boehm: Software Engineering Economics (Prentice Hall, 1981)].

- Grundsatz 4: »**Häufung von Fehlern ("Fehlercluster")**«

  Der Testaufwand soll sich proportional zu der erwarteten und später beobachteten Fehlerdichte auf die Module fokussieren. Ein kleiner Teil der Module enthält gewöhnlich die meisten Fehlerzustände, die während der Testphase entdeckt werden oder ist für die meisten Fehlerwirkungen im Betrieb verantwortlich.

Grundsätze zum Testen (3)

- Grundsatz 5: »**Wiederholungen haben keine Wirksamkeit**«

  Tests nur zu wiederholen, bringt keine neuen Erkenntnisse. Testfälle sind zu prüfen, zu aktualisieren und zu modifizieren.

- Grundsatz 6: »**Testen ist abhängig vom Umfeld**«

  Testen ist abhängig vom Umfeld. Sicherheitskritische Systeme sind anders (intensiver, mit anderen Verfahren, ...) zu testen als beispielsweise der Internetauftritt einer Einrichtung.

- Grundsatz 7: »**Trugschluss: Keine Fehler heißt, dass das System brauchbar ist**«

  Ein System ohne Fehlerwirkungen bedeutet nicht, dass das System auch den Vorstellungen der späteren Nutzer entspricht.

Testaufwand ?

- »Testen ist ökonomisch sinnvoll, solange die Kosten für das Finden und Beseitigen eines Fehlers im Test niedriger sind als die Kosten, die mit dem Auftreten eines Fehlers bei der Nutzung verbunden sind.« M. Pol, T. Koomen, A. Spillner: Management und Optimierung des Testprozesses. dpunkt-Verlag, 2. Auflage, 2002

- »Ein guter Test ist wie eine Haftpflichtversicherung: Er kostet richtig Geld, lässt aber den Projektleiter und den Kunden ruhig schlafen. Zum guten Schlaf gehört auch eine gute Versicherung, die alle möglichen Risiken abdeckt. Zum Vertrauen in die Software gehört ein guter Test, der die ganze Produktionswirklichkeit abdeckt.«

  Siedersleben, J. (Hrsg): Softwaretechnik, Praxiswissen für Softwareingenieure. 2. Auflage, Hanser, 2003

- Erfolgreiches Testen (Nachweis von Fehlerwirkungen) senkt die (Gesamt-)Kosten.

Weiteres zum Testen (1)

- Testaufwand in der Praxis:
  25% bis 50% des Entwicklungsaufwands
- Testintensität und -umfang in Abhängigkeit vom Risiko und der Kritikalität festlegen.
- Fehler können hohe Kosten verursachen
  - Geschätzte Verluste durch Softwarefehler in Mittelstands- und Großunternehmen in Deutschland ca. 84,4 Mrd. Euro p.a.
  - Produktivitätsverluste durch Computerausfälle aufgrund fehlerhafter Software ca. 2,6% des Umsatzes - 70 Mrd. Euro p.a.

Studie der LOT Consulting Karlsruhe, IT-Services 3/2001, S. 31

Weiteres zum Testen (2)

Testen unterliegt immer beschränkten Ressourcen (besonders Zeit, wenn Testen erst am Ende der Entwicklung in Angriff genommen wird)

Besonders wichtig:

- Adäquate (zum Testobjekt und den Qualitätszielen passende) Testverfahren auswählen.
- Unnötige Tests (die keine neuen Erkenntnisse liefern) vermeiden.
- Sowohl Positiv-Tests als auch Negativ-Tests berücksichtigen.
- Auch Tests auf Funktionalität, die nicht gefordert ist, können sinnvoll sein.

Priorisierung der Tests

- Limitierte Ressourcen (Zeit und Personal) erfordern, dass die wichtigsten Testfälle zuerst durchgeführt werden!
- Kriterien für die Priorisierung:
  - Welche Kriterien würden Sie vorschlagen ?

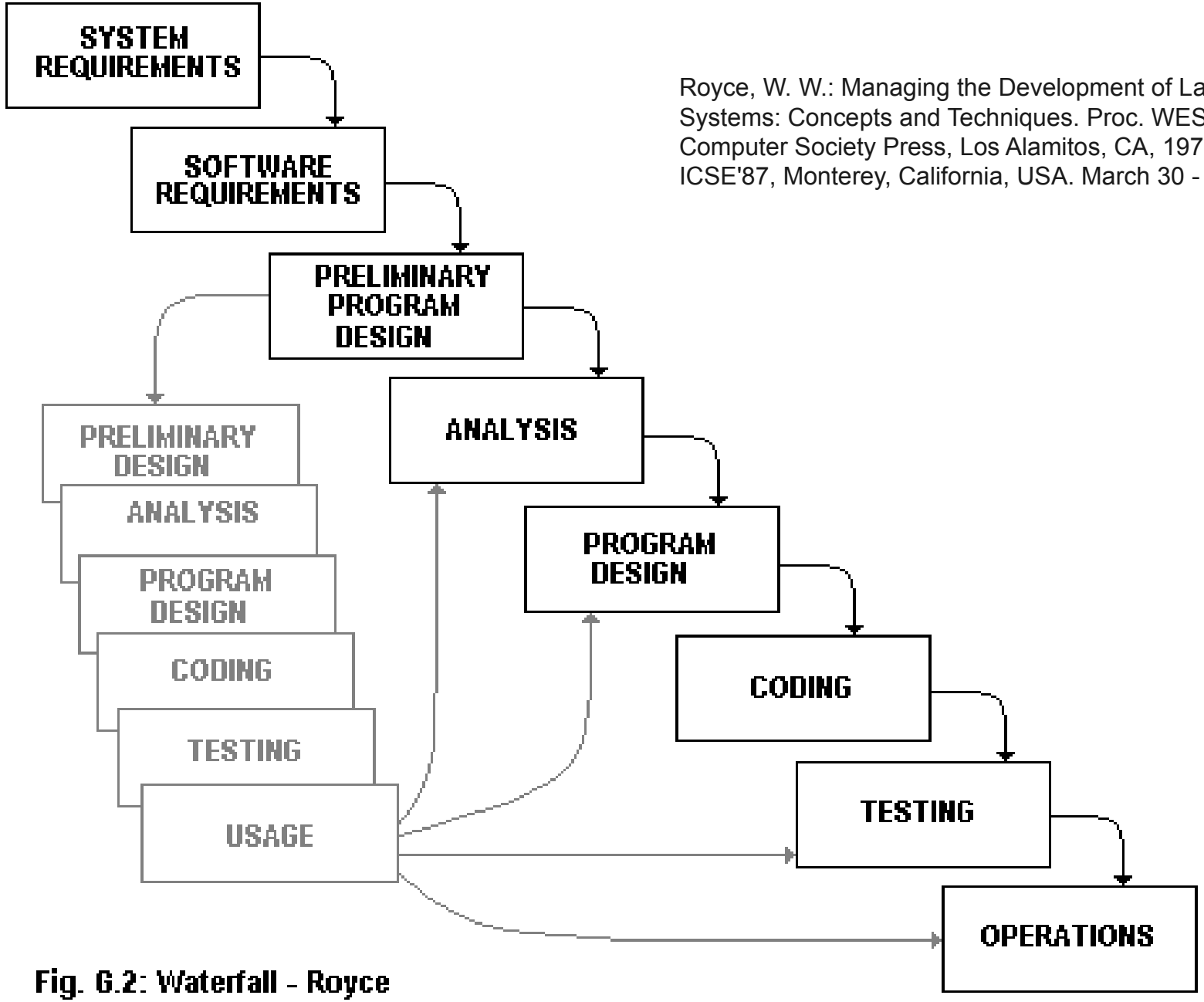
Priorisierung der Tests

- Limitierte Ressourcen (Zeit und Personal) erfordern, dass die wichtigsten Testfälle zuerst durchgeführt werden!
- Kriterien für die Priorisierung:
  - Erwartete Fehlerschwere bzw. Höhe des Schadens
  - Eintrittswahrscheinlichkeit einer Fehlerwirkung
  - Kombinierte Betrachtung von Schwere und Eintrittswahrscheinlichkeit (Risiko = Schadenshöhe * Eintrittswahrscheinlichkeit)
  - Wahrnehmung der Fehlerwirkung
  - Priorität der Anforderungen durch den Kunden
  - Gewichtung der Qualitätsmerkmale durch den Kunden
  - Priorität der Testfälle aus Sicht der Entwicklung (schwerwiegende Auswirkungen oder/und Komplexes zuerst)
  - Hohes Projektrisiko
  - Wo viele Fehler sind, finden sich meist noch mehr. Eine Änderung der Priorität muss daher möglich sein.

Grundlagen des Softwaretestens

- Begriffe und Motivation
- Grundsätze des Softwaretestens
- **Fundamentaler Testprozess**
- Testfälle, Sollwerte und Testorakel
- Psychologie des Testens
- Ethik des Testens

SW-Entwicklungsmodelle: Wasserfall-Modell - 1970



Royce, W. W.: Managing the Development of Large Software Systems: Concepts and Techniques. Proc. WESCON, IEEE Computer Society Press, Los Alamitos, CA, 1970. Reprinted at the ICSE'87, Monterey, California, USA. March 30 - April 2, 1987

Fig. G.2: Waterfall - Royce

SW-Entwicklungsmodelle: Wasserfall-Modell - 1981

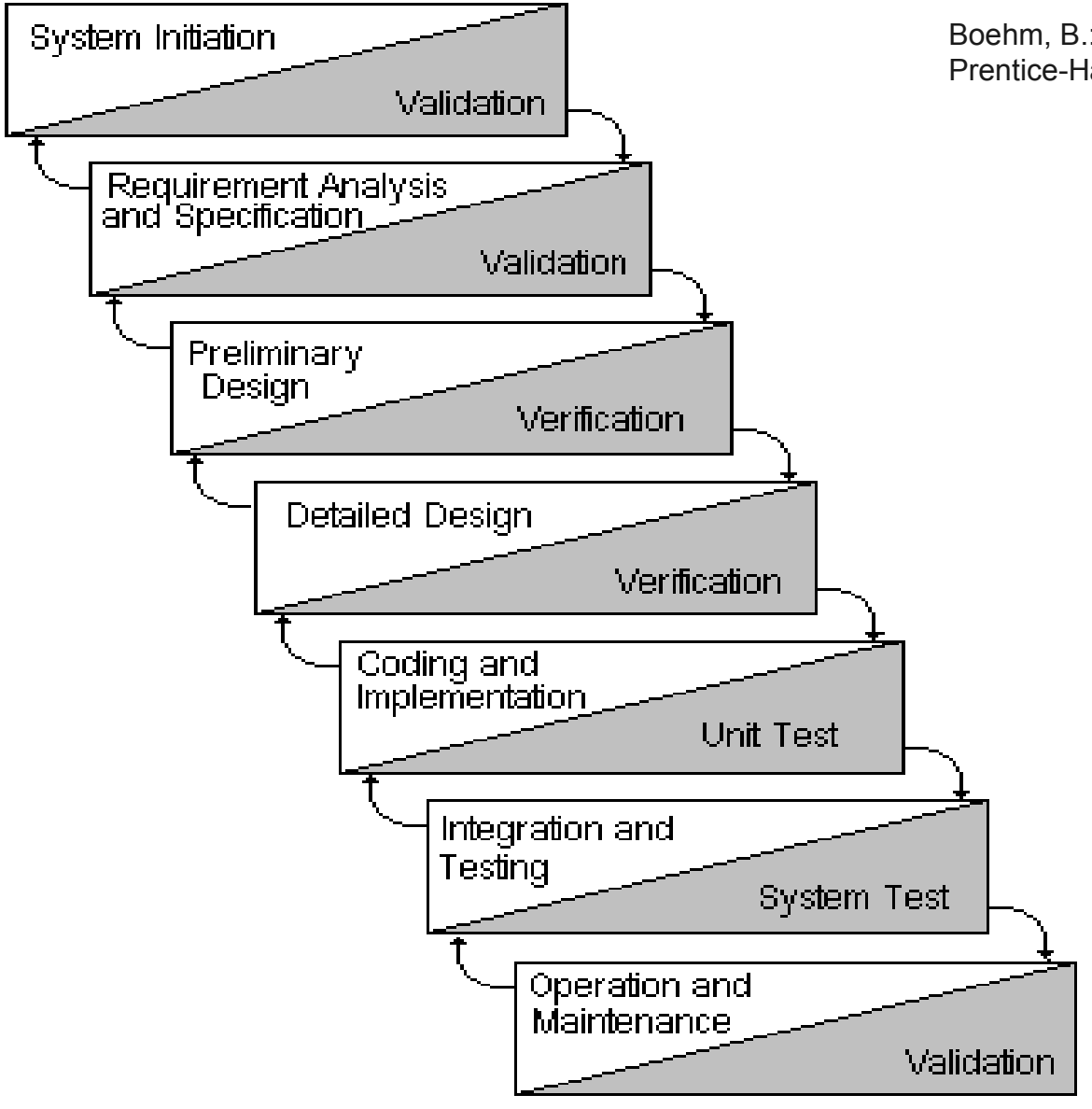
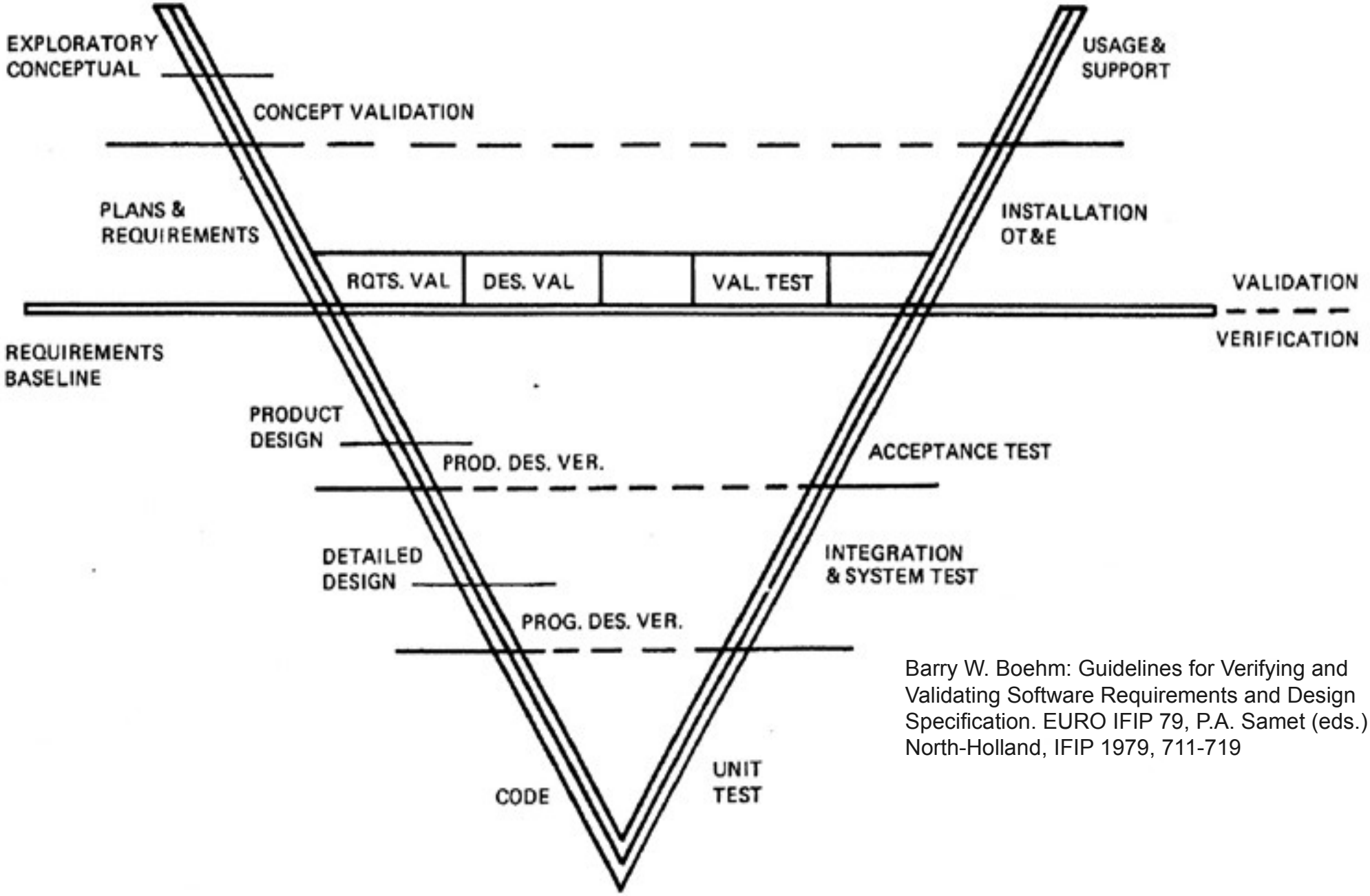


Fig. G.1 : Waterfall Model

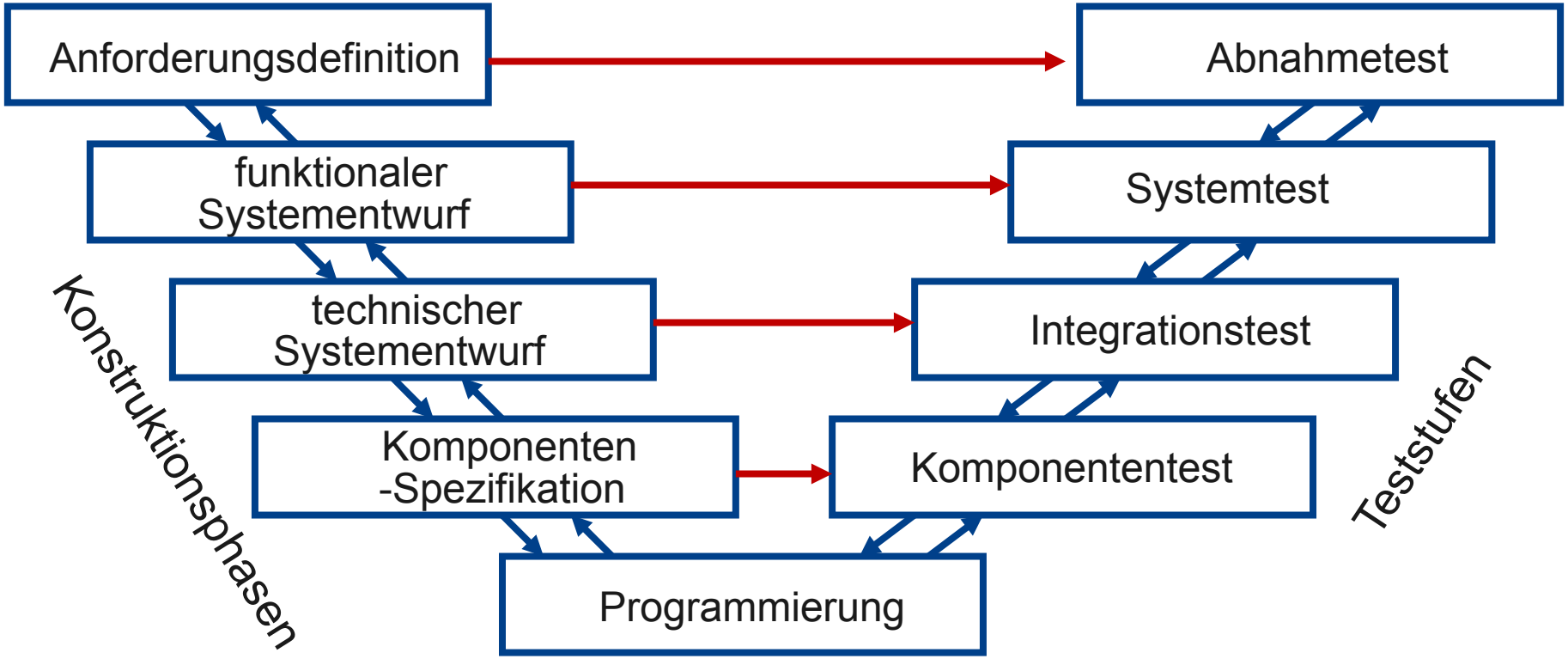
Boehm, B.: Software Engineering Economics. Prentice-Hall Inc., London, 1981

V-Modell (B. Boehm, 1979)



Barry W. Boehm: Guidelines for Verifying and Validating Software Requirements and Design Specification. EURO IFIP 79, P.A. Samet (eds.) North-Holland, IFIP 1979, 711-719

Allgemeines V-Modell



Legende
Testfälle basieren auf den entsprechenden Dokumenten

Testprozess

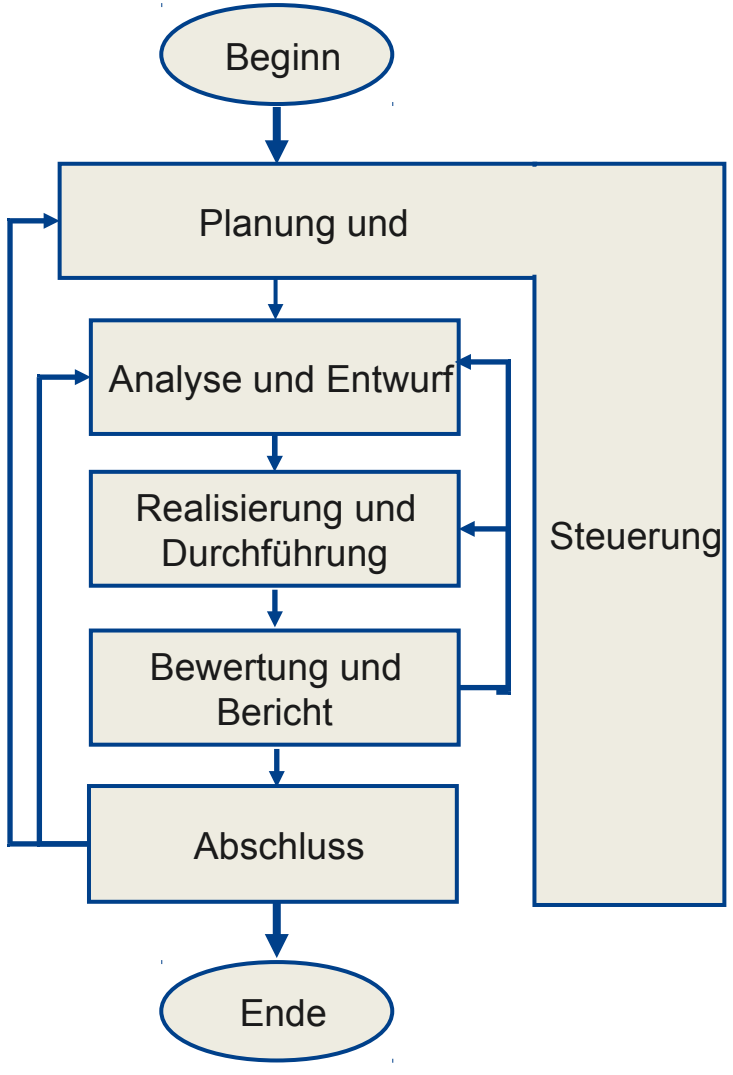
- Ist eng verzahnt mit der Softwareentwicklung.
- Ist jedoch ein eigenständiger Prozess.
- Es ist ein verfeinerter Ablaufplan für die Tests jeder Teststufe notwendig.
- Die Entwicklungsaufgabe »Test« ist in kleine Arbeitsabschnitte (Aktivitäten) aufzuteilen.

Aktivitäten des Testprozesses: Überblick

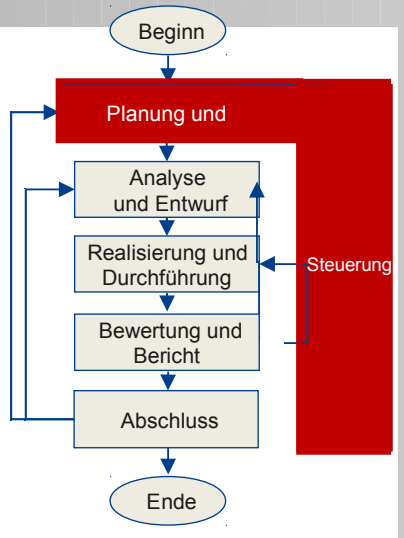
- Testplanung und Steuerung
- Testanalyse und Testentwurf
- Testrealisierung und Testdurchführung
- Bewertung von Ausgangskriterien (Test-Ende-Kriterien) und Bericht
- Abschluss der Testaktivitäten

- Diese Aktivitäten werden z.T. zeitlich überlappend oder parallel ausgeführt.
- Der fundamentale Testprozess ist für jede Teststufe geeignet zu gestalten.

[Einzelzeiten s. Nächste Folien]



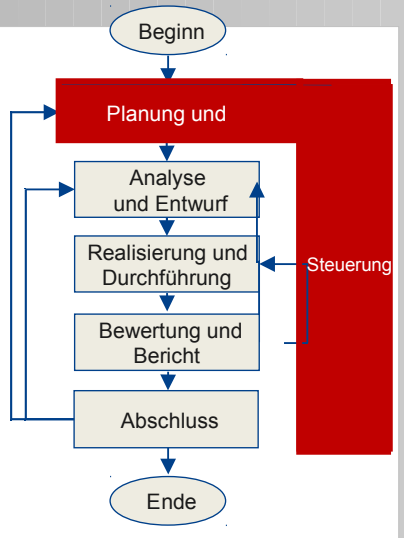
Testaufgaben: Testplanung und Steuerung (1)



- Festlegung der Teststrategie
  - Testziele, Umfang und Risiken des Testens festlegen.
  - Testvorgehensweise spezifizieren (Techniken, Testobjekte, Überdeckung, Teams, die am Test beteiligt sind, Testmittel).
- Detaillierung des Testplans
  - Testressourcen (z.B. Personal, Testumgebung, PCs).
  - Wie intensiv sollen welche Systemteile getestet werden?
  - Welche Testverfahren sollen verwendet werden?
  - Ausgangskriterien (Testendekriterien) bestimmen. Welcher Überdeckungsgrad soll erreicht werden?
  - Priorisierung der Tests (unter Berücksichtigung des Risikos im Fehlerfall).
  - Werkzeugunterstützung einplanen (Einsatz, Beschaffung, Einarbeitung).
- Testplanung und Teststrategie sind im Testkonzept/Mastertestkonzept (engl. test plan) festzuhalten.

Testverfahren = Eine Kombination von Tätigkeiten zum systematischen Erzeugen eines Testproduktes. Testverfahren sind unter Anderem verfügbar für: Testschätzung, Fehlermanagement, Produktrisikoanalyse, Testentwurf, Testdurchführung und Reviews

Testaufgaben: Testplanung und Steuerung (2)



- Grobe Zeitplanung festlegen
  - Testanalyse und -spezifikationsaufgaben terminieren.
  - Testrealisierung, -durchführung und -auswertung terminieren.
- Teststeuerung
  - Messen und Analysieren der Resultate.
  - Überwachen und Dokumentieren von Testfortschritt, erreichter Testabdeckung und der Ausgangskriterien (Testendekriterien).
  - Anstoß von Korrekturmaßnahmen.
  - Anpassung der Zeit- und Ressourcenplanung.
  - Treffen von Entscheidungen.

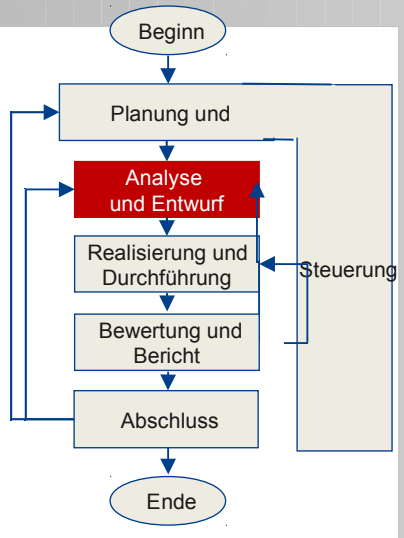
Exkurs: Testkonzept-Begriffe aus Glossar

Mastertestkonzept:
Ein Testkonzept, das sich typischerweise auf mehrere Teststufen bezieht.

Stufentestkonzept:
Ein Testkonzept, das typischerweise für genau eine Teststufe gilt.

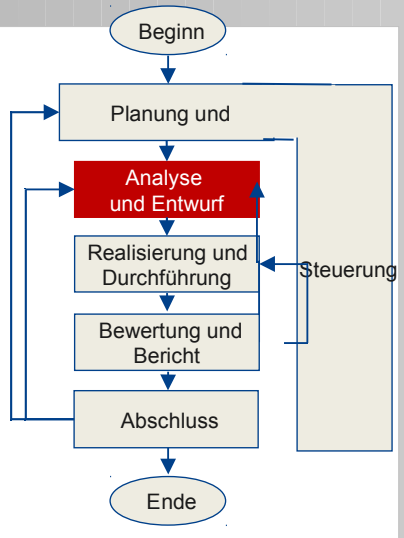
Testkonzept:
Ein Dokument, das u.a. den Gültigkeitsbereich, die Vorgehensweise, die Ressourcen und die Zeitplanung der beabsichtigten Tests mit allen Aktivitäten beschreibt. Es identifiziert u.a. die Testobjekte, die zu testenden Features und die Testaufgaben. Es ordnet den Testaufgaben die Tester zu und legt den Unabhängigkeitsgrad der Tester fest. Es beschreibt die Testumgebung, die Testentwurfsverfahren und die anzuwendenden Verfahren zur Messung der Tests, und begründet deren Auswahl. Außerdem werden Risiken beschrieben, die eine Planung für den Fall des Eintretens erfordern. Ein Testkonzept ist somit die Niederschrift des Testplanungsprozesses [nach IEEE 829].

Testaufgaben: Testanalyse und -entwurf (1)



- Allgemeine Testziele werden zu konkreten Testbedingungen und Kriterien detailliert.
- Grundlage für die Überlegungen sind alle Dokumente, aus denen Anforderungen an das Testobjekt hervorgehen (Testbasis).
- Review der Testbasis (Anforderungen, Architektur, Entwurf, Schnittstellen, ...).
- Testbarkeit von Anforderungen und System bewerten.
- Identifizierung von Testbedingungen / Testanforderungen.
- Entwurf der Testumgebung (Infrastruktur, Werkzeuge, ...).
- Unter Anwendung der gewählten Testentwurfsverfahren sind die entsprechenden Testfälle zu spezifizieren.
- Unterscheidung zwischen abstrakten (logischen) und konkreten Testfällen.
- Sicherstellung der Rückverfolgbarkeit zwischen Testbasis und Testfällen in beiden Richtungen.

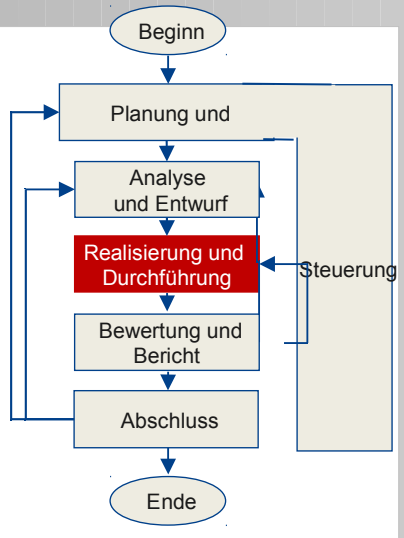
Testaufgaben: Testanalyse und -entwurf (2)



- Testfälle umfassen mehr als nur Testdaten (die Eingaben)!
  - Ausgangsbedingung (Vorbedingung), die vor der Durchführung erfüllt sein muss, z.B. Zustand des Testobjekts und globaler Daten.
  - Vor der Testdurchführung ist festzulegen, welches Ergebnis bzw. welches Verhalten erwartet wird (Soll- Ergebnis bzw. -Verhalten).
  - Nachbedingung, die nach dem Test erfüllt sein muss, z.B. Zustand des Testobjekts und der Datenbank.

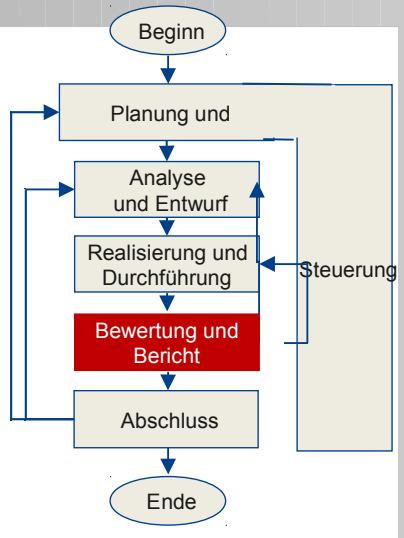
  Anmerkung: Zudem gelten Randbedingungen, die einzuhalten sind, z.B. Netzwerkstatus.
- Testfälle werden in der Testrealisierung teilweise zu Testsuiten (Testmengen, Testablaufspezifikationen) verknüpft:
  - Jeder Testfall sollte nur einen oder wenige elementare Bearbeitungsschritte oder Systemaufrufe (Testschritte) umfassen.
  - Schon beim Testentwurf darauf achten, welche Nachbedingungen von Testfällen welche Vorbedingungen von Testfällen erfüllen.

Testaufgaben: Testrealisierung



- Konkretisierung und Priorisierung der Testfälle.
- Erstellung von Testdaten und Testszenarien.
- Zusammenstellung der Testsuiten.
- Vorbereitung der Testrahmen und Entwicklung von Skripten zur Testautomatisierung.
- Kontrolle, ob das Testsystem korrekt aufgesetzt wurde und Sicherstellung der richtigen Konfigurationen.
- Vorab: Prüfung auf Vollständigkeit der zu testenden Teile.

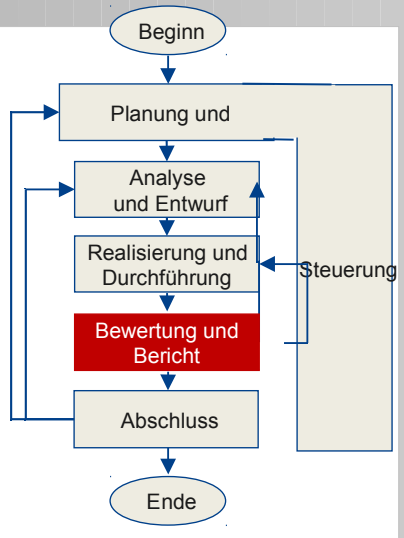
Testaufgaben: Bewertung von Ausgangskriterien und Bericht



- Ergebnisse der Testdurchführung mit definierten Zielen des Tests vergleichen.
- Fehler gefunden ? Fehlerwirkung nachweisbar ?
  Nichtübereinstimmung von Soll- und Ist-Ergebnis !
- Aber: Es kann auch
  - das Soll-Ergebnis
  - die Testumgebung
  - die Testfallspezifikation

  falsch bzw. fehlerhaft sein.
- Fehlerschweregrad (Fehlerklasse) bestimmen, Priorität für die Beseitigung festlegen
  - Testende erreicht ?
  - Überdeckungsgrad erreicht ?
  - Problem: unerreichbarer Programmcode

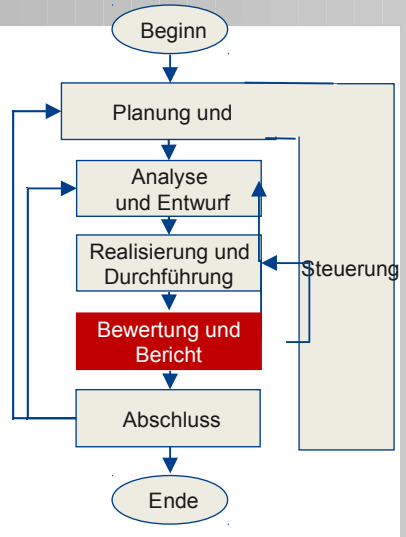
Testaufgaben: Testbericht (1)

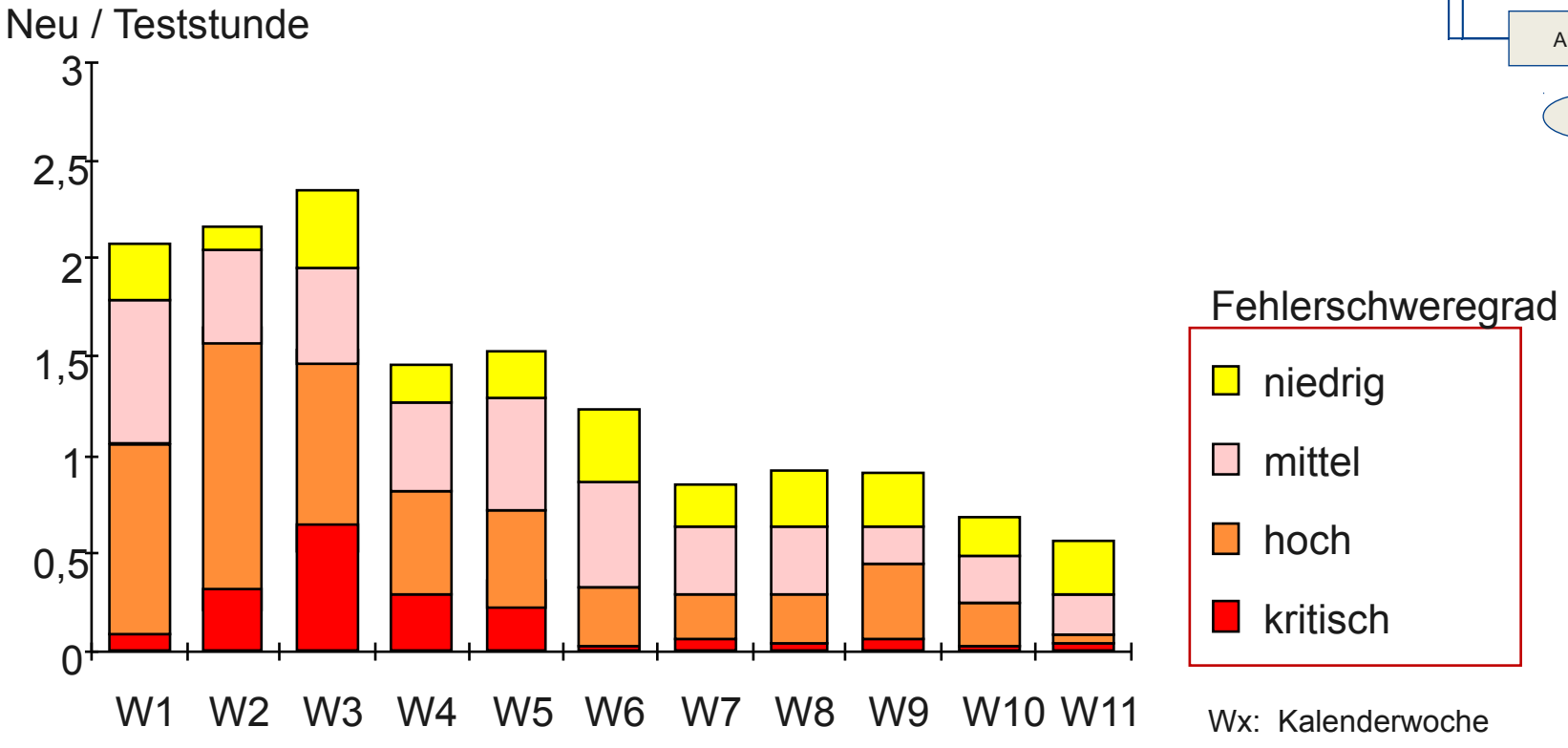


- Aus dem Testprotokoll muss hervorgehen, welche Teile wann, von wem, wie intensiv und mit welchem Ergebnis getestet wurden.
- Zum einen muss anhand dieser Testprotokolle die Testdurchführung auch für nicht direkt beteiligte Personen, zum Beispiel für den Kunden, nachvollziehbar sein
- Zum anderen muss nachweisbar sein, ob die geplante Teststrategie tatsächlich umgesetzt wurde
- Auswertung der Testprotokolle in Hinblick auf die im Testplan festgelegten Ausgangskriterien (Testendekriterien).
- Weiterer Aufwand gerechtfertigt ? Faktoren für das Testende in der Praxis: Zeit und Kosten.
- Entscheidung, ob mehr Tests durchgeführt oder ob die festgelegten Ausgangskriterien angepasst werden müssen.

Testaufgaben: Testbericht (2)

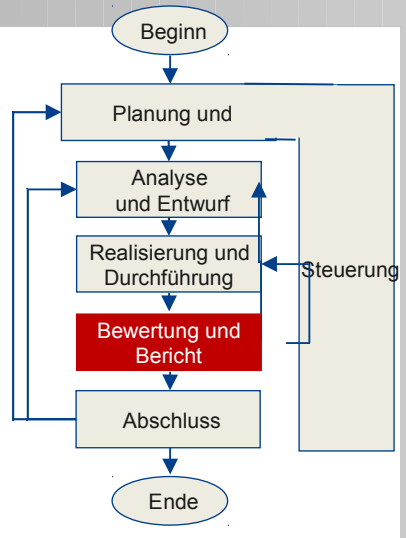
Beispiel für Ausgangskriterien: Gefundene Fehler nach Fehlerschweregrad



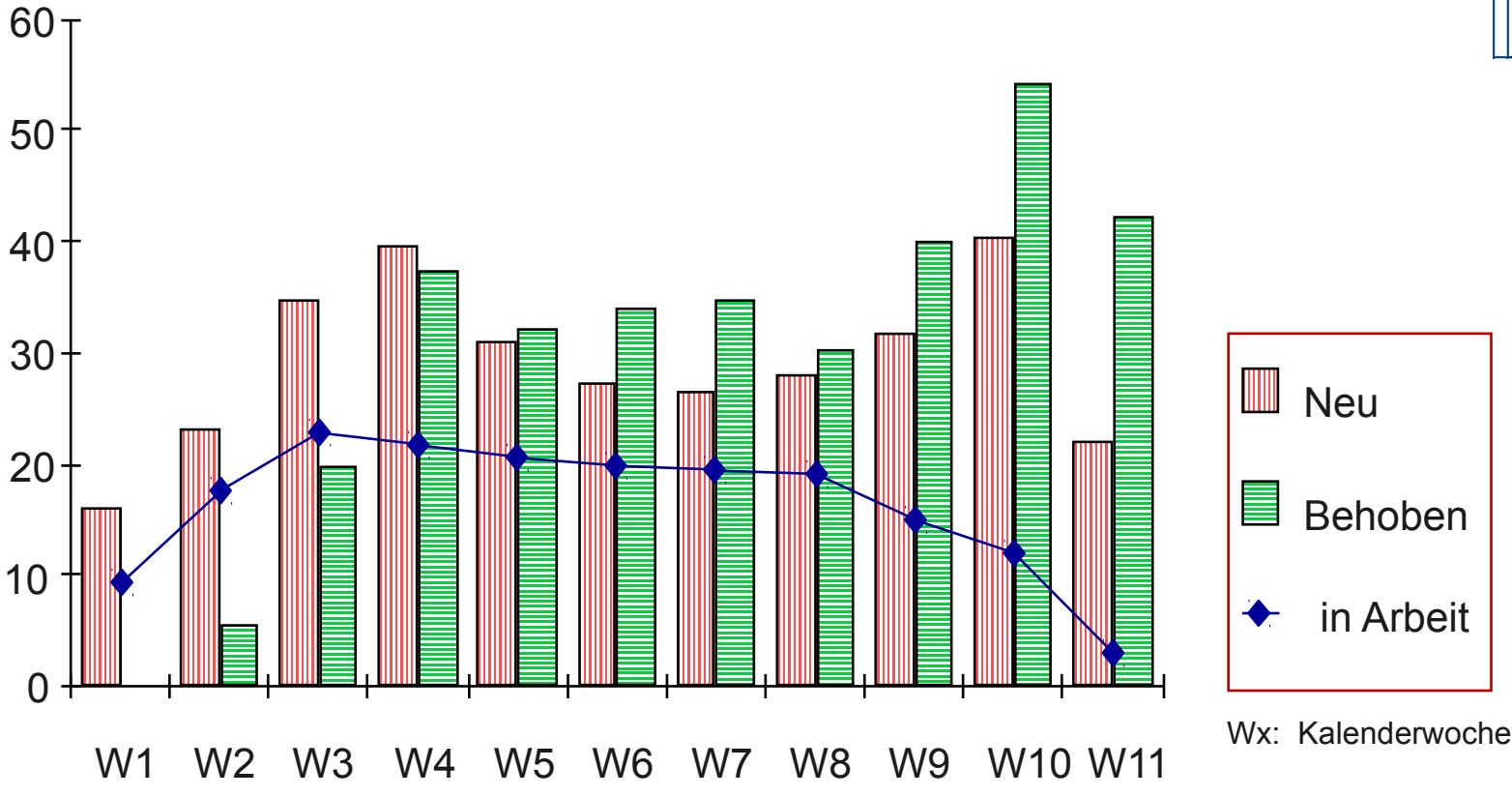


Wx: Kalenderwoche

Testaufgaben: Testbericht (3)

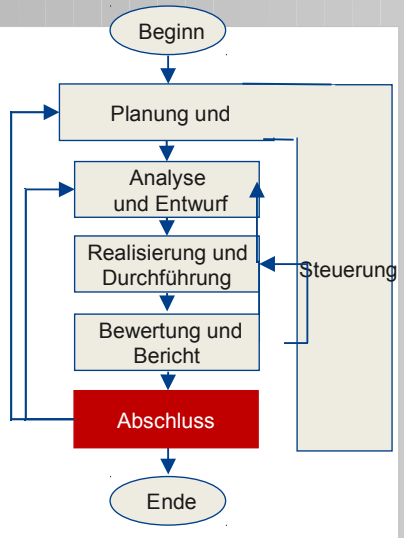


Beispiel für Ausgangskriterien:
Vergleich gefundene / behobene Fehler



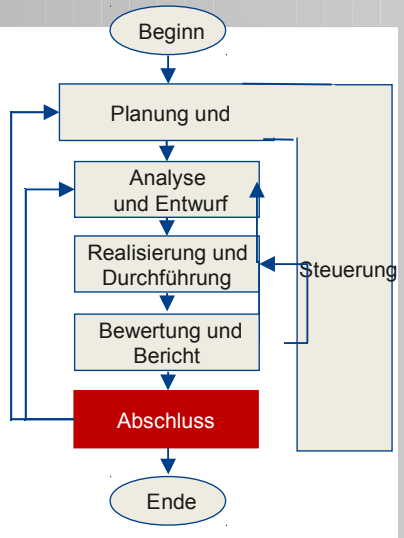
Wx: Kalenderwoche

Testaufgaben: Abschluss der Testaktivitäten (1)



- Daten von abgeschlossenen Testphasen sammeln und konsolidieren (Erfahrungen, Testmittel, Fakten, Zahlen, ...).
- Kontrolle, welche geplanten Arbeitsergebnisse geliefert wurden.
- Schließung der Fehler/Abweichungsmeldungen oder Erstellung von Änderungsanforderungen für weiter bestehende Fehler/ Abweichungen.
- Dokumentation der Abnahme des Systems.

Testaufgaben: Abschluss der Testaktivitäten (2)



- Dokumentation und Archivierung der Testmittel, Testumgebung und der Infrastruktur.
- Konservierung der Testware und Übergabe an die Wartungsorganisation - alles soll während der Wartung und Pflege wieder benutzbar sein und muss daher übertragbar und aktualisierbar sein.
- Analyse und Dokumentation von »*lessons learned*« für spätere Projekte und Verbesserung der Testreife:
  - Evaluation des Testprozesses
  - Einschätzung des Testprozesses
  - Erkennen von Verbesserungspotential
- Verbesserung des Testprozesses durch Umsetzung der Erkenntnisse.

Grundlagen des Softwaretestens

- Begriffe und Motivation
- Grundsätze des Softwaretestens
- Fundamentaler Testprozess
- **Testfälle, Sollwerte und Testorakel**
- Psychologie des Testens
- Ethik des Testens

Kriterien für Testfälle

- Testfälle zur Prüfung der spezifizierten und vom Testobjekt zu liefernden Ergebnisse und Reaktionen.
  - Positiv-Test; erwartete Eingaben bzw. erwartete Bedienung (*normal*).
- Es sind auch die Testfälle zu berücksichtigen, die die spezifizierte Behandlung von Ausnahme- und Fehlersituationen überprüfen.
  - Negativ-Test; erwartete Fehleingaben bzw. erwartete Fehlbedienung (*exceptional*).
  - Anmerkung: Allerdings ist es oft schwierig, die für die Ausführung der Testfälle notwendigen Randbedingungen herzustellen (z. B. die Überlastung einer Netzverbindung).
- Testfälle zur Prüfung der Reaktion des Testobjekts auf ungültige und unerwartete Eingaben bzw. Randbedingungen, für die keine Ausnahmebehandlungen (engl. *exception handling*) spezifiziert wurden.
  - Negativ-Test bzw. Robustheitstest; unerwartete Fehleingaben bzw. unerwartete Fehlbedienung (*catastrophical*).

Beispiel Testspezifikation – abstrakte und konkrete Testfälle (1)

Beispiel

Eine Firma hat ein Programm in Auftrag gegeben, das die Weihnachtsgratifikation der Mitarbeiter in Abhängigkeit von der Firmenzugehörigkeit berechnen soll. In der Beschreibung der Anforderungen findet sich folgende Textpassage:
»Mitarbeiter erhalten ab einer Zugehörigkeit zur Firma von mehr als drei Jahren 50% des Monatsgehalts als Weihnachts-gratifikation. Mitarbeiter, die länger als fünf Jahre in der Firma tätig sind, erhalten 75%. Bei einer Firmenzugehörigkeit von mehr als acht Jahren wird eine Gratifikation von 100% gewährt.«

Wie sehen die Testfälle aus ?

- Welche Fallunterscheidung lässt sich aus dem obigen Text zu dem erwarteten Ein- Ausgabeverhalten der Software ableiten ?
- Wie könnte eine Menge von Testfällen aussehen, die jeden dieser Fälle einmal abdeckt ?

Beispiel Testspezifikation – abstrakte und konkrete Testfälle (2)

Aus dem Text lassen sich folgende Zusammenhänge für die Gewährung der Gratifikation in Abhängigkeit der Firmenzugehörigkeit aufstellen:

	Firmenzugehörigkeit	<= 3 ergibt Gratifikation	=	0 %
3 <	Firmenzugehörigkeit	<= 5 ergibt Gratifikation	=	50 %
5 <	Firmenzugehörigkeit	<= 8 ergibt Gratifikation	=	75 %
	Firmenzugehörigkeit	> 8 ergibt Gratifikation	=	100 %

Beispiel Testspezifikation – abstrakte und konkrete Testfälle (3)

Abstrakter (logischer) Testfall	1	2	3	4
Eingabewert x (Firmenzugehörigkeit)	x<=3	3<x<=5	5<x<=8	x>8
Vorausgesagtes Ergebnis (Gratifikation in %)	0	50	75	100

Konkreter Testfall	1	2	3	4
Eingabewert x (Firmenzugehörigkeit)	2	4	7	12
vorausgesagtes Ergebnis (Gratifikation in %)	0	50	75	100

Anmerkung: Es wurde keine Rand-, Vor- und Nachbedingungen vermerkt,
die Testfälle wurden nicht systematisch hergeleitet,
nur positive Tests mit erwarteten Ergebnissen.

Sollwerte und Testorakel

- Nach jedem durchgeführten Testfall muss entschieden werden, ob eine Fehlerwirkung vorliegt oder nicht.
- Um das entscheiden zu können, muss das beobachtete Ergebnis (Ist-Ergebnis/Ist-Verhalten) mit dem erwarteten Ergebnis (Soll-Ergebnis/Soll-Verhalten) verglichen werden.
- Das Soll-Verhalten des Testobjekts für jeden Testfall muss dazu vorab bestimmt werden.
- Diese Information muss sich der Tester bei der Spezifikation eines Testfalls aus geeigneten Quellen beschaffen.
- In diesem Zusammenhang wird auch von einem Orakel oder Testorakel gesprochen, das befragt werden muss und die jeweiligen Soll-Ergebnisse vorhersagt.

Testorakel

- Die Soll-Werte sind aus der Spezifikation abzuleiten.
- Folgende drei Möglichkeiten:
  - Der Tester leitet das Soll-Datum aus dem Eingabedatum auf Grundlage der Spezifikation des Testobjekts ab. Dies ist das in der Praxis übliche Vorgehen.
  - Liegt eine formale Spezifikation vor, kann daraus ein ausführbarer Prototyp werkzeuggestützt erzeugt werden. Die Ergebnisse des Prototypen können dann als Testorakel für den Test des eigentlichen Programms dienen.
  - Das Softwareprogramm wird mehrfach von unabhängigen Entwicklungsgruppen parallel erstellt. Jede Programmversion wird dann mit den gleichen Testdaten gegen eine andere getestet. Bei unterschiedlichen Ergebnissen der Programmversionen ist ein Fehlerzustand in einer Version vorhanden. Nur diejenigen Fehlerzustände, die in allen Versionen mit den gleichen Fehlerwirkungen vorhanden sind, bleiben unentdeckt.
    Dieses Vorgehen wird als *Back-to-back*-Test bezeichnet.

Weitere Testorakel

- Benutzungshandbuch verwenden
- System selbst (Nutzungsprofil erstellen):
  schlechteste Möglichkeit
- (Getestete!) Vorgängerversionen, Programme mit ähnlicher Funktionalität
- Nicht immer lassen sich exakte Werte vorhersagen oder ermitteln.
  Toleranzbereich festlegen, Plausibilität prüfen
- Erfahrung ist wichtig

2.1 Grundlagen des Softwaretestens

Grundlagen des Softwaretestens

- Begriffe und Motivation
- Grundsätze des Softwaretestens
- Fundamentaler Testprozess
- Testfälle, Sollwerte und Testorakel
- Psychologie des Testens
- Ethik des Testens

Psychologie des Testens – Entwicklertests

Was sind Vor- und Nachteile, wenn der Entwickler sein eigenes Programm testet ?

Psychologie des Testens – Entwicklertests

Was sind Vor- und Nachteile, wenn der Entwickler sein eigenes Programm testet ?

Vorteil:

- Kein Einarbeitungsaufwand (der Entwickler kennt sein Testobjekt).

Nachteile:

- Interessenkonflikt (Errare humanum est - aber wer gibt es schon gerne zu ?).
- Beim Testen sind andere Begabungen / Interessen gefordert als beim Entwickeln. (Entwicklung = konstruktiv / Test = destruktiv ?).
  (Aber auch: »Testen ist eine extrem kreative und intellektuell herausfordernde Aufgabe.«) Myers, Glenford J.: Methodisches Testen von Programmen, Oldenbourg, 2001 (7. Auflage)
- Blindheit gegenüber den eigenen Fehlern: Falls der Entwickler einen grundsätzlichen Entwurfsfehler eingebaut hat, z.B. weil er die Aufgabenstellung falsch verstanden hat, kann er das auch durch seine Tests nicht herausfinden. Der passende Testfall wird ihm überhaupt nicht in den Sinn kommen.

Psychologie des Testens – Unabhängiges Testteam

Vorteile:

- Unvoreingenommen
  - Es ist nicht »sein« Produkt, und mögliche Annahmen und Missverständnisse des Entwicklers sind nicht zwangsläufig auch die Annahmen und Missverständnisse des Testers.
- Test-Know-how
  - bringt der Tester mit
  - ein Entwickler nicht bzw. er muss es sich erst aneignen (oder besser müsste, da dazu die notwendige Zeit oft nicht vorhanden ist).
  - Erfahrene Tester können auch spezialisierte Werkzeuge anwenden, die besonders effizient, effektiv, verlässlich und nachvollziehbar testen (z.B. automatisches / modell-basiertes Testen)

Nachteil:

- Einarbeitung notwendig
  - Um Testfälle zu erstellen, muss der Tester sich aber das benötigte Wissen über das Testobjekt aneignen, was Zeit kostet.
  - Andererseits können in dieser Phase bereits Probleme (von unabhängiger Seite) identifiziert werden, die Einarbeitungszeit ist also nicht völlig „nutzlos“.

Psychologie des Testens – Mögliche Abstufungen

- Tests werden
  - vom Entwickler selbst
  - vom Kollegen des Entwicklers im gleichen Projekt
  - von Personen anderer Abteilungen
  - von Personen anderer Organisationen

  durchgeführt.
- Bei allen Möglichkeiten können Werkzeuge eingesetzt werden.
- Die Aufteilung ist produkt- und projektabhängig.
- Wichtig ist die richtige Mischung und Ausgewogenheit zwischen »unabhängigen Tests« und Entwicklertests.

Psychologie des Testens – Mitteilung von Fehlern

- Mitteilung von gefundenen Fehlerwirkungen oder Abweichungen an den Entwickler oder/und dem Management verlangen:
  - neutrale, objektive und konstruktive Art und Weise der Mitteilung und
  - ungestörte, offene Kommunikation.
- Anderen Menschen Fehlhandlungen nachzuweisen ist keine leichte Aufgabe, die viel »Fingerspitzengefühl« erfordert.
- Reproduzierbarkeit ist wichtig !
  - Dokumentation der Testumgebung
  - Unterschiede zur Entwicklungsumgebung
- Eindeutige Anforderungen, präzise Spezifikation
  - »It’s not a bug, it’s a feature!«
- Gegenseitiges Verständnis
  - Ein Mit- und kein Gegeneinander !

Ethik des Testens

Softwaretester erhalten oft Zugang zu vertraulichen und rechtlich privilegierten Informationen. Diese sind nicht zu missbrauchen. Daher Formulierung der **ethische Leitlinien des ISTQB** in Anlehnung an den Ethik-Kodex von ACM und IEEE.

1. ÖFFENTLICHKEIT: Das Verhalten zertifizierter Softwaretester soll nicht im Widerspruch zum öffentlichen Interesse stehen.
2. KUNDE UND ARBEITGEBER: Das Verhalten zertifizierter Softwaretester soll den Interessen ihrer Kunden und Arbeitgeber entsprechen und dabei nicht im Widerspruch zum öffentlichen Interesse stehen.
3. PRODUKT: Zertifizierte Softwaretester sollen sicherstellen, dass die Arbeitsergebnisse, die sie liefern (für die von ihnen getesteten Produkte und Systeme), höchste fachliche Anforderungen erfüllen.
4. URTEILSVERMÖGEN: Zertifizierte Softwaretester sollen bei ihrer professionellen Beurteilung aufrichtig und unabhängig sein.
5. MANAGEMENT: Zertifizierte Softwaretestmanager und -testleiter sollen eine ethische Haltung beim Management des Softwaretestens haben und fördern.
6. BERUFSBILD: Zertifizierte Softwaretester sollen Integrität und Ansehen ihres Berufs fördern und dabei nicht im Widerspruch zum öffentlichen Interesse stehen.
7. KOLLEGEN: Zertifizierte Softwaretester sollen sich ihren Kolleginnen und Kollegen gegenüber fair und hilfsbereit verhalten und die Kooperation mit Softwareentwicklern fördern.
8. PERSÖNLICH: Zertifizierte Softwaretester sollen sich in ihrem Beruf lebenslang fort- und weiterbilden und eine ethische Haltung in ihrer Berufsausübung vertreten.

Zusammenfassung

- Fehlerwirkung - Nichterfüllung einer Anforderung!
  - Beschreibung dieses Sachverhalts als Fehlerwirkung (*failure*)
  - Ursache: Fehlerzustand (*fault*) in der Software
  - die durch eine Fehlhandlung (*error*) einer Person verursacht wurde.
- Tests sind wichtige Maßnahmen zur Sicherung der Qualitätsmerkmale:
  - Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit [nach ISO 9126]
- Austesten ist nicht möglich, Tests sind immer Stichproben, sie sollen deshalb mit einer gewissen Wahrscheinlichkeit Fehlerwirkungen nachweisen.
- Intensität und Umfang der Tests ist abhängig vom erwarteten Risiko bei fehlerhaftem Verhalten des Programms.
- Priorisierung der Tests ist vorzunehmen.

Zusammenfassung

- Der fundamentale Testprozess besteht aus den Hauptaktivitäten
  - Testplanung und -steuerung
  - Testanalyse und Testentwurf
  - Testrealisierung und -durchführung
  - Bewertung von Ausgangskriterien und Bericht
  - Abschluss der Testaktivitäten
- Ein Testfall besteht aus Eingabewert und Soll-Ergebnis sowie Vorbedingungen, unter denen der Testfall ablaufen muss, und Nachbedingungen, die nach Ablauf erfüllt sein müssen.
- Wird der Testfall ausgeführt, zeigt das Testobjekt ein Ist-Verhalten.
  - Sind Soll- und Ist-Verhalten verschieden, liegt ggfs. eine Fehlerwirkung vor.
  - Es ist zu prüfen, ob ein Fehler im Soll-Verhalten, im Testfall, in der Spezifikation oder im Testobjekt liegt.
- Ein Testorakel bestimmt die Soll-Werte für jeden Testfall vor Testdurchführung.
- Menschen machen Fehler – aber sie geben es nur ungern zu !

Sie sollten jetzt ...

- Anhand von Beispielen beschreiben können, auf welche Art ein Softwarefehler Menschen, der Umwelt oder einem Unternehmen Schaden zufügen kann
- Wissen, was unter den Begriffen Mangel, Fehler, Fehlerwirkung, Fehlerzustand/Defekt und Fehlhandlung zu verstehen ist und diese anhand von Beispielen erklären und vergleichen können
- Zwischen der Ursache eines Fehlerzustands und seinen Auswirkungen unterscheiden können
- Anhand von Beispielen herleiten können, warum Testen notwendig ist
- Beschreiben können, warum Testen Teil der Qualitätssicherung ist und Beispiele angeben, wie Testen zu einer höheren Qualität beiträgt
- Die allgemeinen Zielsetzungen und Grundsätze des Testens kennen
- Zwischen Testen und Debugging unterscheiden können
- Wissen, wie der fundamentale Testprozess aussieht
- Beschreiben können, wie Soll-Werte beim Testen ermittelt werden
- Die psychologischen Probleme beim Testen kennen und charakterisieren können
- Zwischen der unterschiedlichen Denkweise eines Testers und eines Entwicklers und eines Entwicklers differenzieren können
- Die ethischen Leitlinien für die Tätigkeit eines Softwaretesters kennen

Folgende Fragen sollten Sie jetzt beantworten können

- Definieren Sie die Begriffe Fehlerwirkung, Fehlerzustand, Fehlhandlung.
- Was ist Fehlermaskierung ?
- Erläutern Sie den Unterschied zwischen Testen und Debugging.
- Definieren Sie die Begriffe Verifizierung und Validierung.
- Erläutern Sie, warum jeder Test eine stichprobenartige Prüfung ist.
- Nennen Sie die Hauptmerkmale der Softwarequalität nach ISO 9126. (vgl. Abschnitt 1.0)
- Definieren Sie den Begriff Zuverlässigkeit eines Systems.
- Erläutern Sie die Hauptaktivitäten des fundamentalen Testprozesses.
- Was ist ein Testorakel ?
- Warum sollte ein Entwickler nicht seine eigenen Programme testen ?