Vorlesung (WS 2013/14)

Softwarekonstruktion

Prof. Dr. Jan Jürjens

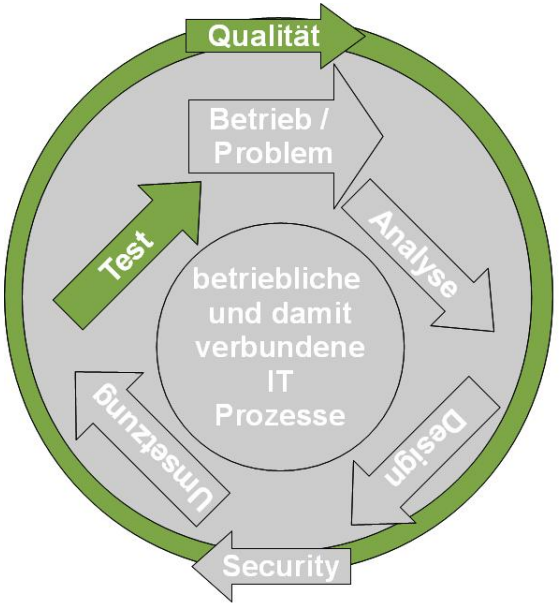
TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

Teil 3.0: Grundlagen der Softwareverifikation

v. 29.11.2013

Einordnung
Grundlagen Softwareverifikation

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



[Basierend auf dem Foliensatz „Basiswissen Softwaretest - Certified Tester“ des „German Testing Board“, 2011]

Literatur (s. Webseite):

- Andreas Spillner, Tilo Linz: Basiswissen Softwaretest.
- Eike Riedemann: Testmethoden für sequentielle und nebenläufige Software-Systeme.

Was gilt als Fehler oder Mangel

Fehler:

- **Nichterfüllung** festgelegter Anforderung.
- **Abweichung** zwischen Ist-Verhalten und Soll-Verhalten.

Mangel:

- Gestellte Anforderung oder berechtigte **Erwartung** in Bezug auf beabsichtigten Gebrauch **nicht angemessen erfüllt**.
- Z.B. Beeinträchtigung der Verwendbarkeit bei gleichzeitiger Erfüllung der Funktionalität oder Nichterfüllung angemessener Erwartung.

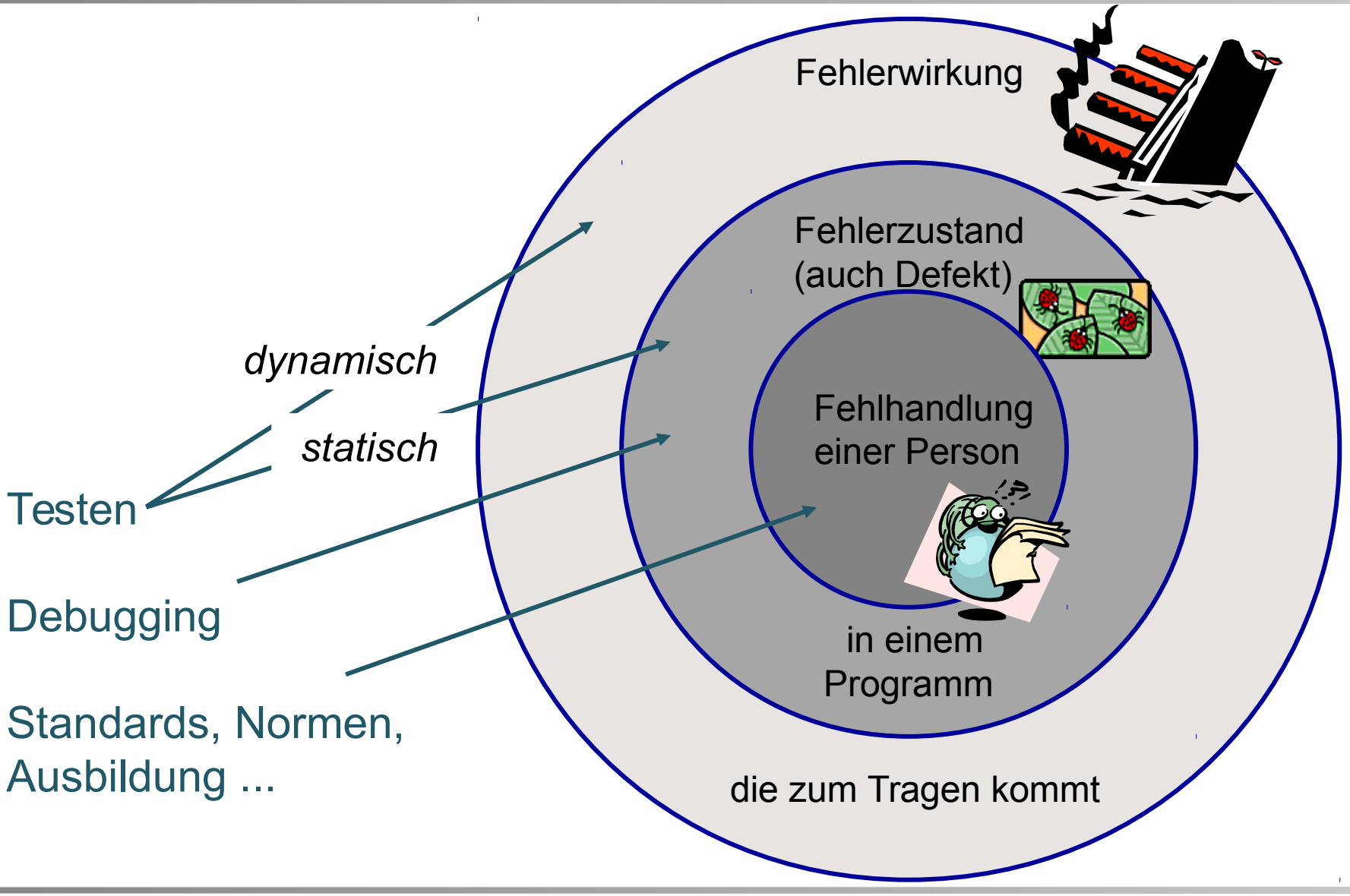
Ursachenkette für Fehler

- Jeder Fehler oder Mangel: seit Zeitpunkt der Fertigstellung in Software vorhanden.

  → Kommt erst bei Ausführung der Software zum Tragen.
- Beschreibung dieses Sachverhalts als **Fehlerwirkung** (*failure*).
- Ursache einer Fehlerwirkung: **Fehlerzustand** (*fault*) in 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

Entstehen von Fehlern und Gegenmaßnahmen



Wiederholung: Verifizierung vs. Validierung ???

V?????ierung

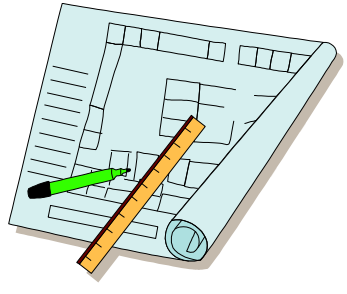
Prüfung, ob Entwicklungsergebnis individuelle Anforderungen bezüglich einer speziellen beabsichtigten Nutzung erfüllt.

- Haben wir das **richtige System** realisiert?

V?????ierung

Prüfung, ob Ergebnisse einer Entwicklungsphase Vorgaben der Phaseneingangs-Dokumente erfüllen.

- Haben wir das System **richtig realisiert**?



Wiederholung: Verifizierung vs. Validierung

Validierung

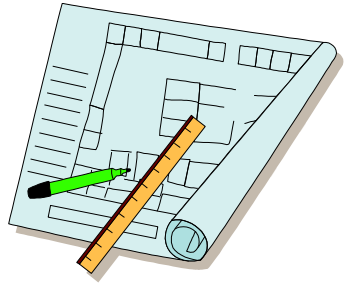
Prüfung, ob Entwicklungsergebnis individuelle Anforderungen bezüglich einer speziellen beabsichtigten Nutzung erfüllt.

- Haben wir das **richtige System** realisiert?

Verifizierung

Prüfung, ob Ergebnisse einer Entwicklungsphase Vorgaben der Phaseneingangs-Dokumente erfüllen.

- Haben wir das System **richtig realisiert**?



Testen und Qualität

Testen misst Qualität z.B. anhand Anzahl gefundener Fehlerwirkungen.

Erhöht indirekt

- **Qualität**, da Fehler(zustände) vor Auslieferung entdeckt und korrigiert werden können.
- **Prozessqualität**, da Fehler dokumentiert, analysiert und damit Fehlhandlungen in Zukunft vermieden werden können.
- **Vertrauen** in Qualität des Systems, wenn wenige oder keine Fehler gefunden werden.

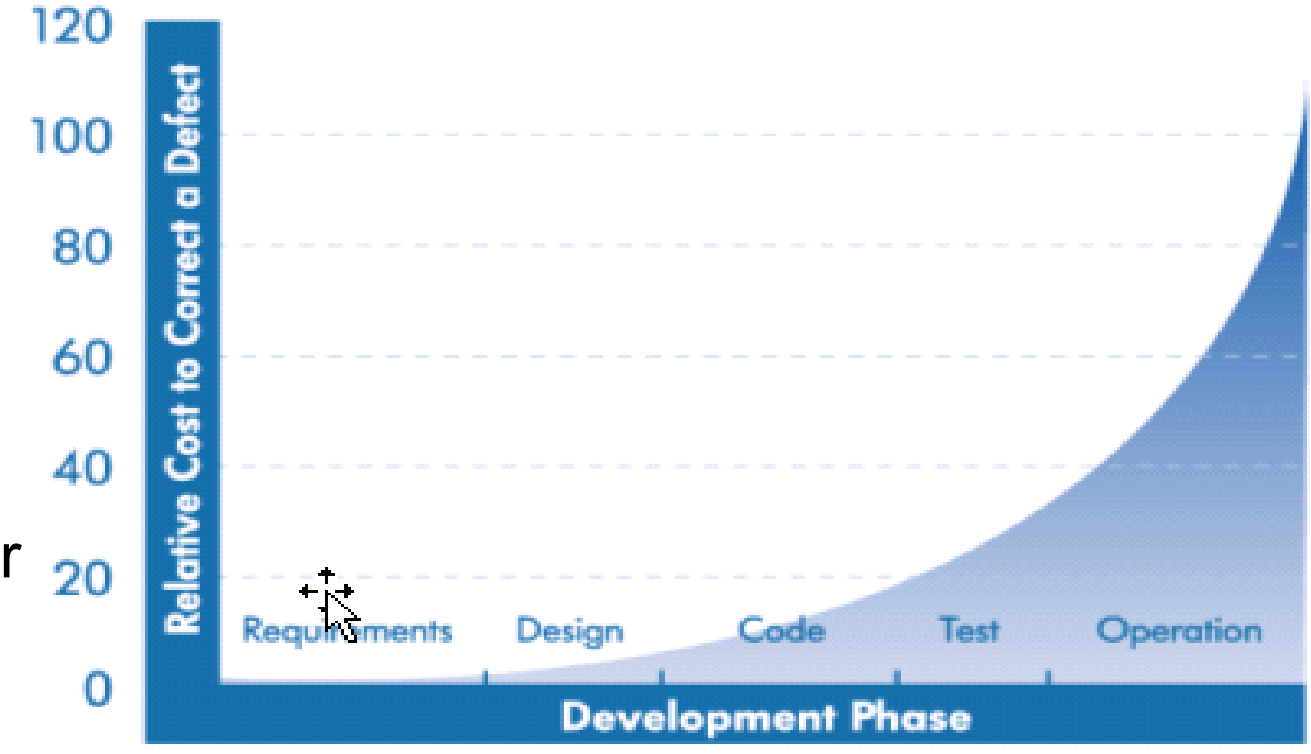
Testen vs. Kosten (1)

- **Testaufwand** in Praxis:
  - 25% bis 50% des Entwicklungsaufwands
- **Testintensität** und **-umfang** in Abhängigkeit vom Risiko und 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

Testen vs. Kosten (2)

- **Frühzeitiges Prüfen:** Fehler (zustände) früher erkennen und somit Kosten senken.



[„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)].

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

Testfälle

- **Testfall** besteht aus Eingabewert, Soll-Ergebnis, 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. Fehlerwirkung vor.
  - Prüfen, ob Fehler im Soll-Verhalten, Testfall, in der Spezifikation oder im Testobjekt liegt.
- **Testorakel** bestimmt Soll-Werte für jeden Testfall vor Testdurchführung.

Positiv- vs. Negativtest

- Prüfung der spezifizierten und vom Testobjekt zu liefernden Ergebnisse und Reaktionen:
  - **Positiv-Test:** erwartete Eingaben bzw. erwartete Bedienung (*normal*).
- Spezifizierte Behandlung von Ausnahme- und Fehlersituationen überprüfen:
  - **Negativ-Test:** erwartete Fehleingaben bzw. erwartete Fehlbedienung.
- Prüfung der Reaktion des Testobjekts auf ungültige / unerwartete Eingaben / Randbedingungen, für die keine Ausnahmebehandlungen spezifiziert.
  - Negativ-Test bzw. **Robustheitstest:** unerwartete **Fehleingaben** bzw. unerwartete **Fehlbedienung**.

Beispiel Testspezifikation – abstrakte und konkrete Testfälle (1)

LEHRSTUHL 14
SOFTWARE ENGINEERING

Beispiel
Berechnung der Weihnachtsgratifikation der Mitarbeiter in Abhängigkeit von Firmenzugehörigkeit einer Firma.

Anforderungen:

- Ab einer Firmenzugehörigkeit von mehr als 3 Jahren 50% des Monatsgehalts.
- Mitarbeiter, die länger als 5 Jahre in der Firma tätig sind, erhalten 75%.
- Bei einer Firmenzugehörigkeit von mehr als 8 Jahren wird 100% gewährt.

Wie sehen **Testfälle** aus ?

- Welche Fallunterscheidung lässt sich aus obigem Text zum erwarteten **Ein- Ausgabeverhalten der Software** ableiten ?
- Wie könnte Menge von Testfällen aussehen, die jeden dieser Fälle einmal abdeckt ?

Beispiel Testspezifikation – abstrakte und konkrete Testfälle (2)

LEHRSTUHL 14
SOFTWARE ENGINEERING

Abstrakter (logischer) Testfall	1	2	3	4
Eingabewert x (Firmenzugehörigkeit)				
Vorausgesagtes Ergebnis (Gratifikation in %)				

Konkreter Testfall	1	2	3	4
Eingabewert x (Firmenzugehörigkeit)				
vorausgesagtes Ergebnis (Gratifikation in %)				

- Ab einer Firmenzugehörigkeit von mehr als 3 Jahren 50% des Monatsgehalts.
- Mitarbeiter, die länger als 5 Jahre in der Firma tätig sind, erhalten 75%.
- Bei einer Firmenzugehörigkeit von mehr als 8 Jahren wird 100% gewährt.

Beispiel Testspezifikation – abstrakte und konkrete Testfälle (2)

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

Anmerkungen:

- Beispiel für Äquivalenzklassenbasiertes Testen.
- Keine Rand-, Vor- und Nachbedingungen vermerkt, Testfälle wurden nicht systematisch hergeleitet, nur positive Tests mit erwarteten Ergebnissen.

Sollwerte und Testorakel

- Nach jedem durchgeführten Testfall muss entschieden werden, ob **Fehlerwirkung** vorliegt oder nicht.
- Dafür beobachtetes Ergebnis mit erwartetem Ergebnis vergleichen.
- **Soll-Verhalten** des Testobjekts für jeden Testfall vorab bestimmen.
- Diese Information muss sich Tester bei Spezifikation eines Testfalls aus geeigneten Quellen beschaffen.

  → **Orakel / Testorakel:** Muss befragt werden und vorhersagt jeweilige Soll-Ergebnisse.

Testorakel

Welche Möglichkeiten können Sie sich für Testorakel vorstellen ?

-
-
-

Drei hauptsächliche Möglichkeiten:

- **Übliches Vorgehen** in Praxis: Soll-Datum aus Eingabedatum auf Grundlage der Spezifikation des Testobjekts ableiten.
- **Prototyp: z.B. werkzeuggestützte Erzeugung** durch formale Spezifikation. Prototyp Testorakel für Test des eigentlichen Programms.
- ***Back-to-back*-Test:** Softwareprogramm mehrfach von unabhängigen Entwicklungsgruppen parallel erstellt. **Programmversionen** mit Testdaten gegeneinander testen. **Unterschiedliche** Ergebnisse => Fehlerzustand in einer Version. Nur diejenigen Fehlerzustände unentdeckt, die in allen Versionen mit gleichen Fehlerwirkungen vorhanden.

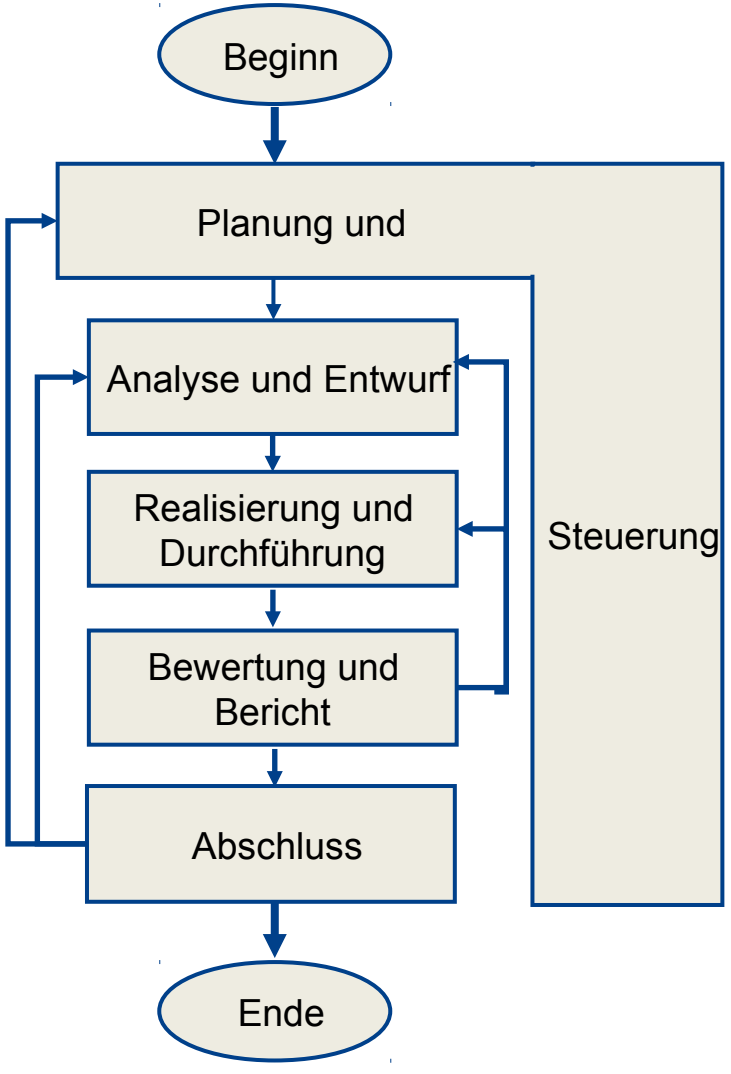
Aktivitäten des Testprozesses: Überblick

LEHRSTUHL 14
SOFTWARE ENGINEERING

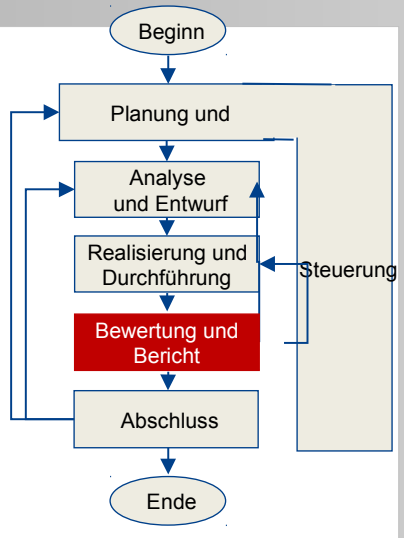
- **Testplanung** und Steuerung
- Testanalyse und **Testentwurf**
- **Testrealisierung** und Testdurchführung
- **Bewertung** von Ausgangskriterien (Test-Ende-Kriterien) und Bericht
- **Abschluss** der Testaktivitäten

→ Werden z.T. zeitlich überlappend oder parallel ausgeführt.

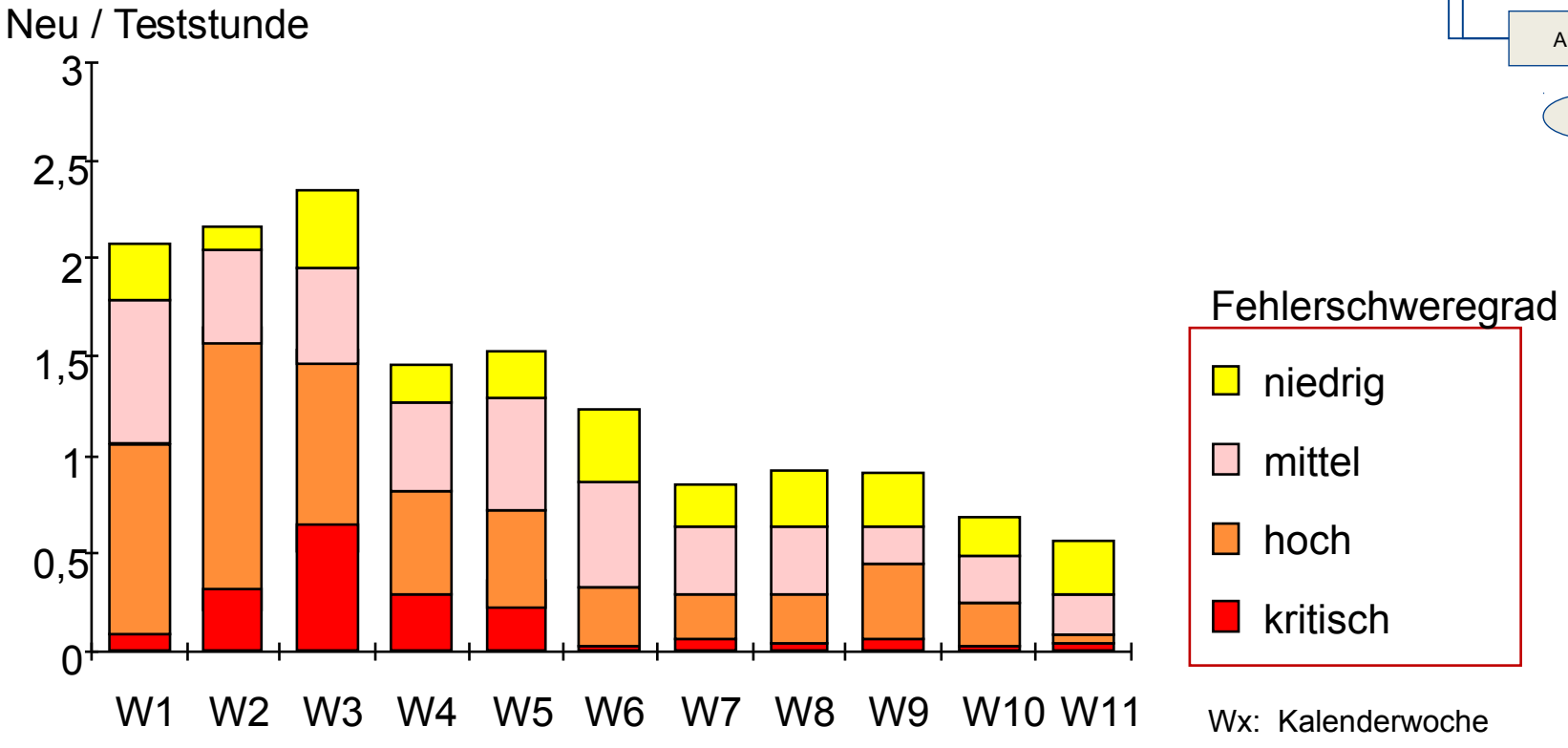
→ **Fundamentaler Testprozess:** Für jede Teststufe geeignet zu gestalten.



Testaufgaben: Testbericht (2)

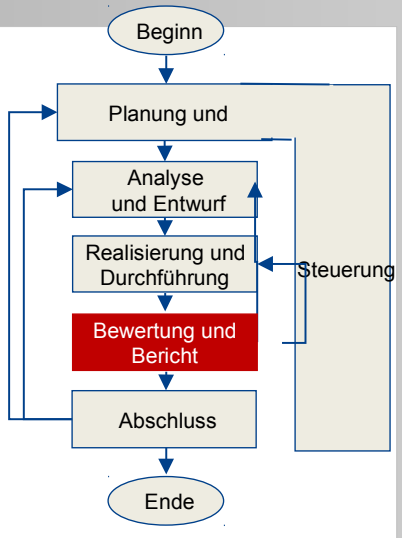


Beispiel für **Ausgangskriterien**:
Gefundene Fehler nach Fehlerschweregrad

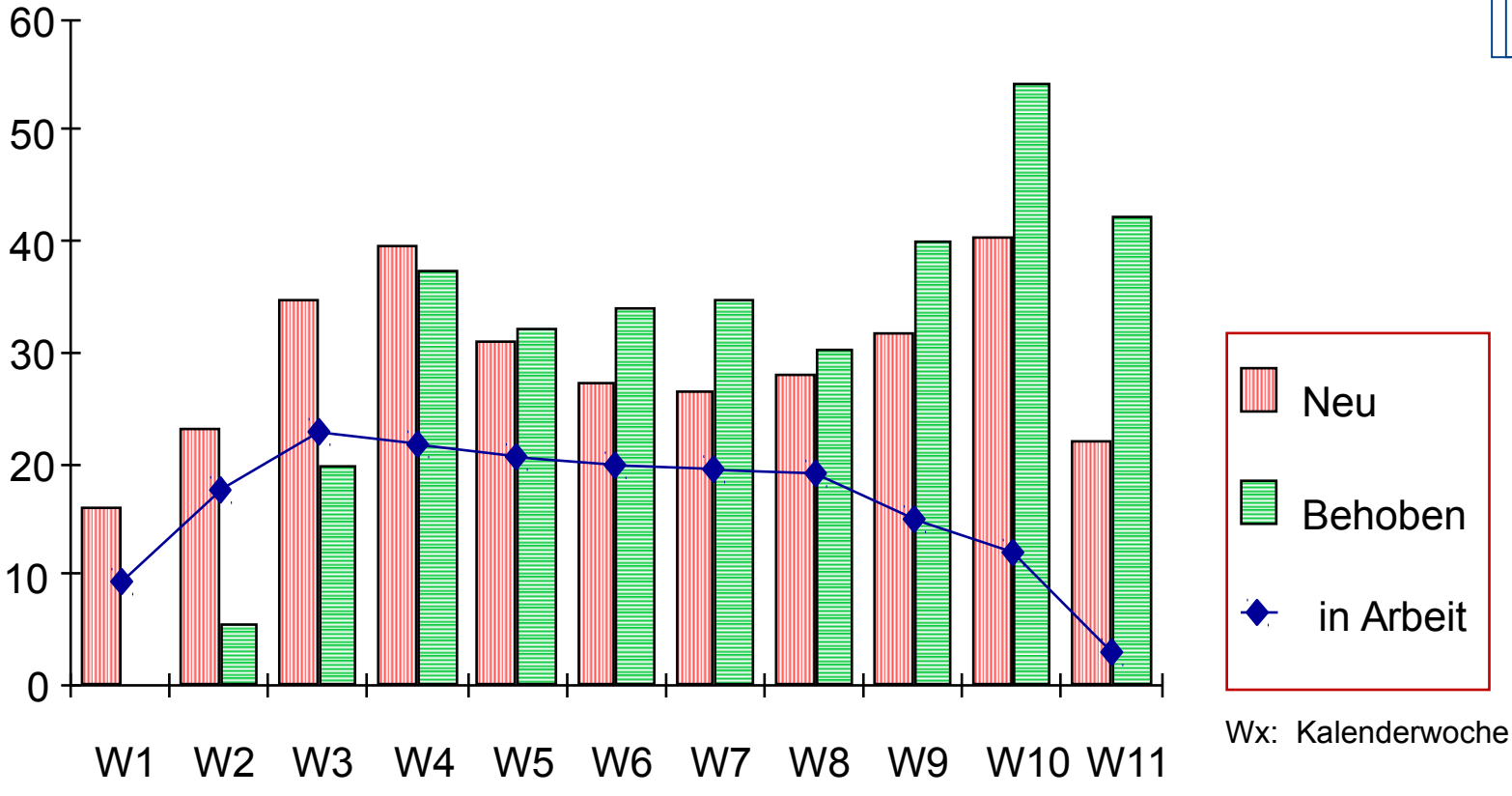


Wx: Kalenderwoche

Testaufgaben: Testbericht (3)

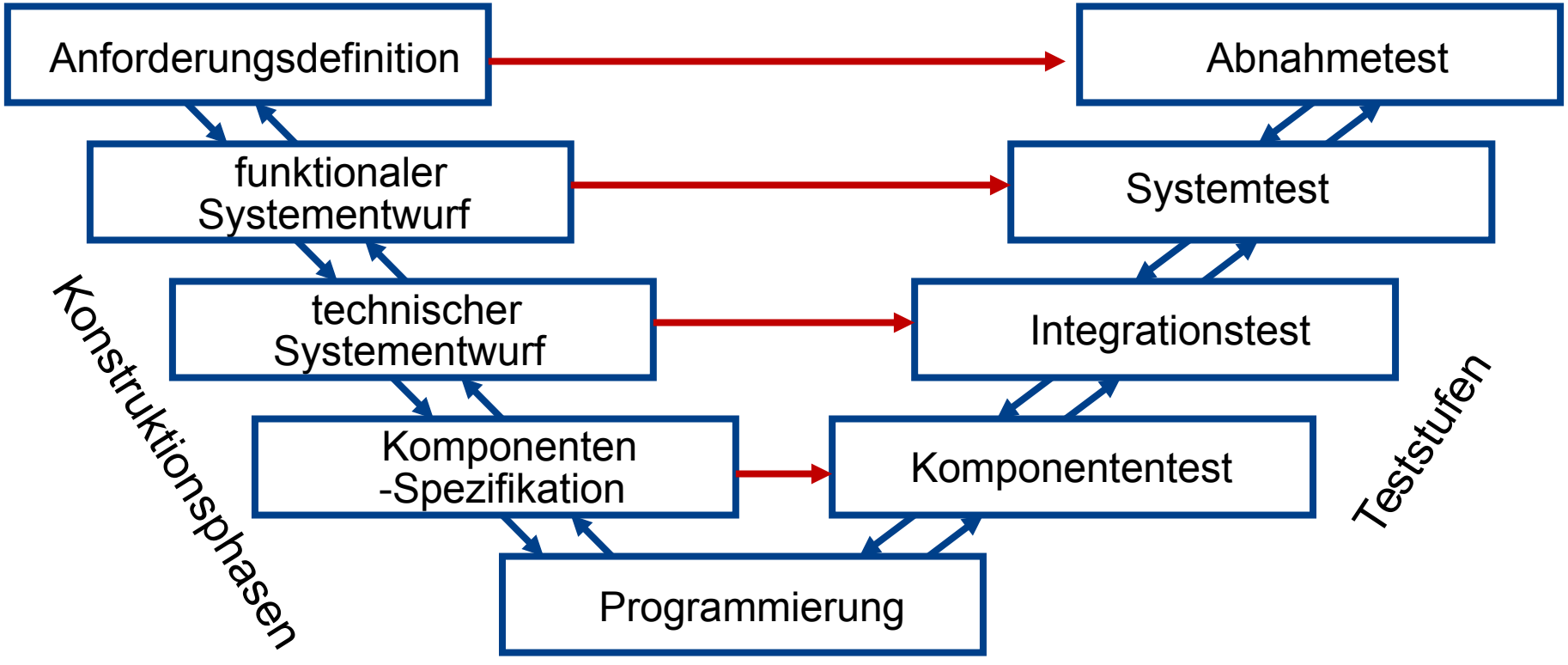


Beispiel für **Ausgangskriterien:**
Vergleich gefundene / behobene Fehler



Wx: Kalenderwoche

Testen vs. Konstruktion: V-Modell



Legende
Testfälle basieren auf den entsprechenden Dokumenten

Einige Arten der Software-Prüfung

- Dynamisch
  - White-box Testen
  - Black-Box Testen
- Statisch
  - Software-Metriken

Zusammenfassung

- Fehlerwirkung - **Nichterfüllung** einer Anforderung!
  - Beschreibung dieses Sachverhalts als Fehlerwirkung (*failure*).
  - Ursache: Fehlerzustand (*fault*) in Software.
  - die durch Fehlhandlung (*error*) einer Person verursacht wurde.
- Software-Prüfung: **Wichtige Maßnahmen** zur Sicherung der Softwarequalität

Als nächstes: Dynamisches Testen, zunächst White-Box-Testen.

Anhang
(zusätzliche Einzelheiten und Beispiele)

Quelle Kapitel 3

- Ein großer Teil der Folien wurde übernommen aus einer gemeinsamen Hochschul-übergreifenden Lehrveranstaltung und vom GTB entsprechend des derzeitigen Syllabus 2011 aktualisiert.
- Dank gilt den Kollegen:
  - Dr. Falk Fraikin, TU Darmstadt
  - Dr. Eike Hagen Riedemann, ehemals TU Dortmund
  - Prof. Dr. Andreas Spillner, Hochschule Bremen
  - Prof. Dr. Mario Winter, FH Köln
- HS@GTB bezeichnet die Hochschul-Mitglieder des German Testing Boards (Stand 07.2011):
  - Dipl.-Inf. Timea Illes-Seifert, EnBW SIS GmBH
  - Dipl.-Inf. Horst Pohlmann, Hochschule Ostwestfalen-Lippe
  - Prof. Dr.-Ing. Ina Schieferdecker, FU Berlin
  - Prof. Dr. Mario Winter, FH Köln

aktuelle Informationen unter http://www.german-testing-board.info

Fehlerhandlung – Definition

- **Fehlhandlung *(error)***
  1. **Menschliche Handlung** (des Entwicklers), die zu einem Fehlerzustand in der Software führt.
  2. Menschliche Handlung (des Anwenders), die **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 geforderte Funktion eines Produkts beeinträchtigt ist.

Fehlerzustand / -wirkung – Definition

- **Fehlerzustand / Defekt (fault)**
  - **Inkorrektes Teilprogramm** (z.B. mit inkorrekter Anweisung oder Datendefinition), das Ursache für Fehlerwirkung sein kann.

  - Zustand eines (Software-)Produkts oder einer seiner Komponenten, der unter spezifischen Bedingungen geforderte Funktion des Produkts beeinträchtigen kann bzw. zur Fehlerwirkung führt.
- **Fehlerwirkung (failure)**
  - Wirkung eines Fehlerzustands, die bei Ausführung eines Programms nach »außen« in Erscheinung tritt.

  - **Abweichung** zwischen (spezifizierten) Soll-Wert und (beobachtetem) Ist-Wert (bzw. Soll- und Ist-Verhalten).
  - Abweichung einer Komponente/eines Systems von erwarteter Lieferung, Leistung oder dem Ergebnis [nach Fenton].

•Anmerkung: Fehlerwirkungen können (selten) durch Höhenstrahlung, elektromagnetische Felder oder Hardwarefehler hervorgerufen werden.

•Anmerkung: So kann Fehlerwirkung zur langsamen Ausführung, zu inkorrekten Ausgaben oder zum Abbruch der Ausführung führen.

Testen – Definition und Ziele

Prozess, der sich (sowohl statisch als auch dynamisch) mit

- **Planung**
- **Vorbereitung**
- **Bewertung** einer Software
- hierzu in Beziehung stehenden **Arbeitsergebnissen**

befasst.

Um Software mit dem **Ziel** zu bewerten, dass diese

- allen festgelegten Anforderungen
- ihren Zweck erfüllt und
- etwaige Fehlerzustände zu finden.

Grundsätze zum Testen (1)

Grundsätze zum Testen in letzten 40 Jahren und können als Leitlinien dienen:

- Grundsatz 1: »**Testen zeigt die Anwesenheit von Fehlern**«
  - Nachweisen der Anwesenheit von Fehlerwirkungen.
  - 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 Testen – Austesten – nicht möglich.

Grundsätze zum Testen (2)

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

  - **Frühzeitiges Prüfen:** Fehler(zustände) früher erkannen und somit Kosten senken.

[„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")**«

  - Testaufwand: Proportional zur erwarteten und später beobachteten Fehlerdichte auf Module fokussieren.
  - Kleiner Teil der Module enthält meiste Fehlerzustände.
    → Während Testphase entdeckt.
    → Für meisten Fehlerwirkungen im Betrieb verantwortlich.

Grundsätze zum Testen (3)

- Grundsatz 5: »**Wiederholungen haben keine Wirksamkeit**«
  - Tests nur zu wiederholen, bringt keine neuen Erkentnisse.
  - Testfälle sind zu prüfen, zu aktualisieren und zu modifizieren.
- Grundsatz 6: »**Testen ist abhängig vom Umfeld**«
  - Vom Umfeld abhängig.
  - Sicherheitskritische Systeme intensiver testen
- Grundsatz 7: »**Trugschluss: Keine Fehler heißt, dass das System brauchbar ist**«
  - System ohne Fehlerwirkungen ≠ System entspricht den Vorstellungen der späteren Nutzer.

Testaufwand ?

- »Testen ist ökonomisch sinnvoll, solange **Kosten** für Finden und Beseitigen eines Fehlers im Test niedriger sind als Kosten, die mit Auftreten eines Fehlers bei **Nutzung** verbunden sind.«

  M. Pol, T. Koomen, A. Spillner: Management und Optimierung des Testprozesses. dpunkt-Verlag, 2. Auflage, 2002

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

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

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

Weiteres zum Testen (1)

- **Testaufwand** in Praxis:
  - 25% bis 50% des Entwicklungsaufwands
- **Testintensität** und **-umfang** in Abhängigkeit vom Risiko und 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.

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!**

Priorisierung der Tests
Kriterien für Priorisierung

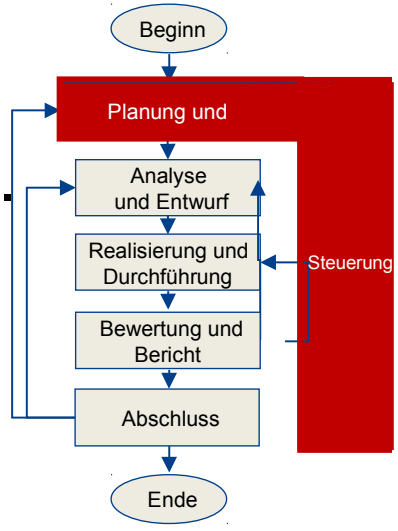
- **Limitierte Ressourcen** (Zeit und Personal) erfordern, dass wichtigste Testfälle zuerst durchgeführt werden!
- Kriterien für Priorisierung:
  - Erwartete Fehlerschwere bzw. Höhe des Schadens
  - **Eintrittswahrscheinlichkeit** einer Fehlerwirkung
  - Kombinierte Betrachtung von Schwere und Eintrittswahrscheinlichkeit (Risiko = Schadenshöhe * Eintrittswahrscheinlichkeit)
  - **Wahrnehmung** der Fehlerwirkung
  - Priorität der Anforderungen durch Kunden
  - Gewichtung der Qualitätsmerkmale durch 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.
    → Möglichkeit zur Änderung der Priorität.

Testprozess

- Eng verzahnt mit Softwareentwicklung.
- **Eigenständiger Prozess**.
- **Verfeinerter Ablaufplan** für Tests jeder Teststufe notwendig.
- Entwicklungsaufgabe »Test« ist in kleine Arbeitsabschnitte (Aktivitäten) aufzuteilen.

Testaufgaben: Testplanung und Steuerung (2)

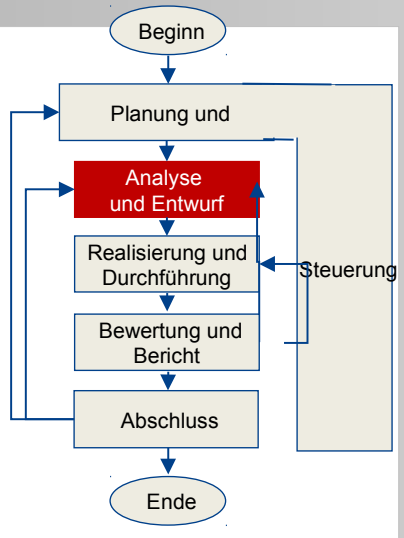
LEHRSTUHL 14
SOFTWARE ENGINEERING



- 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 Ressourcenmaßnahmen.
  - Treffen von Entscheidungen.

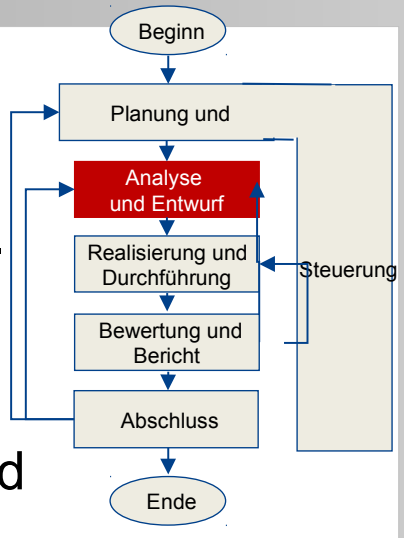
Testaufgaben: Testanalyse und -entwurf (1)

LEHRSTUHL 14
SOFTWARE ENGINEERING



- 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.

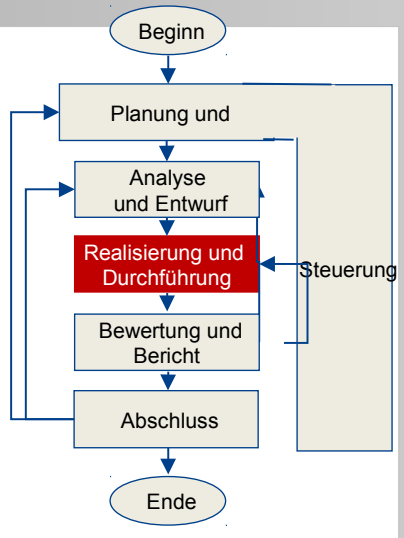
Testaufgaben: Testanalyse und -entwurf (2)



- 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.

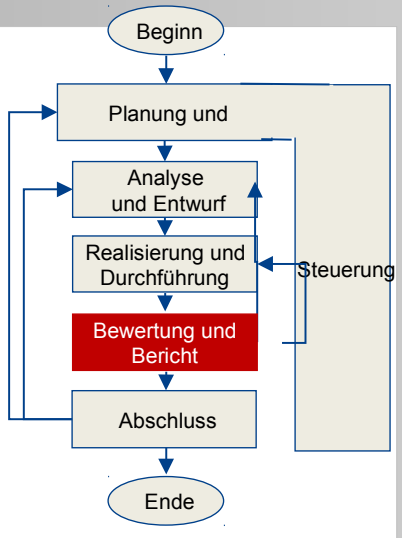
Testaufgaben: Testrealisierung



- 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

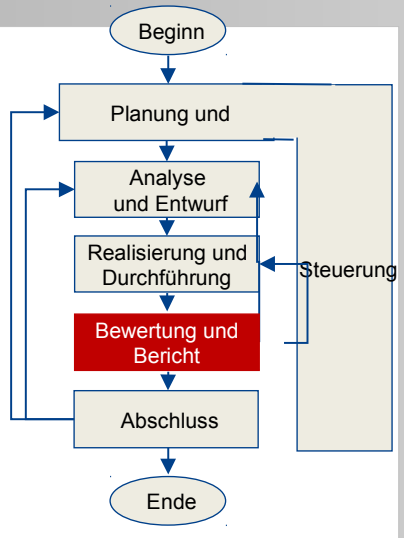
LEHRSTUHL 14
SOFTWARE ENGINEERING



- 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

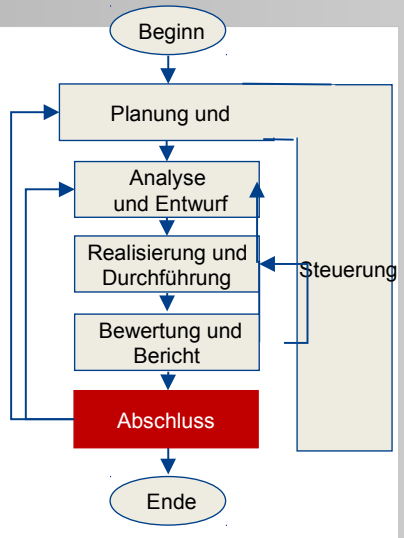
  falsch bzw. fehlerhaft sein.
- **Fehlerschweregrad** (Fehlerklasse) bestimmen, Priorität für die Beseitigung festlegen
  - Testende erreicht ?
  - Überdeckungsgrad erreicht ?
  - Problem: unerreichbarer Programmcode

Testaufgaben: Testbericht (1)



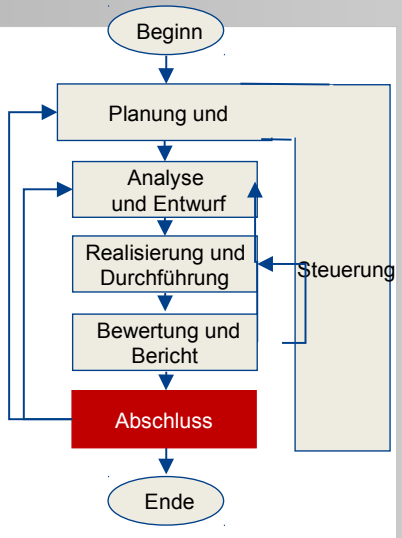
- Aus dem **Testprotokoll** muss hervorgehen, welche Teile wann, von wem, wie intensiv und mit welchem Ergebnis getestet wurden.
- Anhand dieser Testprotokolle muss die **Testdurchführung** für nicht direkt beteiligte Personen (Kunden) nachvollziehbar sein.
- Es muss nachweisbar sein, ob die geplante **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: 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

  → Während der Wartung und Pflege soll wieder alles benutzbar sein.

  → 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.