Zürcher Hochschule
für Angewandte Wissenschaften

zh aw School of Engineering

Bachelor of Science (BSc) in Informatik

Modul Advanced Software Engineering 2 (ASE2)

LE 10 – Software Testing

6 Testmanagement

Institut für Angewandte Informationstechnologie (InIT)
Walter Eich (eicw) / Matthias Bachmann (bacn)
https://www.zhaw.ch/de/engineering/institute-zentren/init/

Agenda

6 Testmanagement

6.1 Testorganisation

6.2 Teststrategie

6.3 Testplanung, Teststeuerung und Testüberwachung

6.4 Fehlermanagement

6.5 Konfigurationsmanagement

6.6 Relevante Normen und Standards

6.7 Wrap-up

Lernziele nach Syllabus ISTQB CTFL (1/2)

5.1 Testorganisation

FL-5.1.1 (K2) Vor- und Nachteile unabhängigen Testens erklären können

FL-5.1.2 (K1) Die Aufgaben eines Testmanagers und eines Testers benennen können

5.2 Testplanung und -schätzung

FL-5.2.1 (K2) Den Zweck und Inhalt eines Testkonzepts zusammenfassen können

FL-5.2.2 (K2) Zwischen verschiedenen Teststrategien unterscheiden können

FL-5.2.3 (K2) Beispiele für mögliche Eingangs- und Endekriterien geben können

FL-5.2.4 (K3) Wissen über Priorisierung sowie technische und logische Abhängigkeiten anwenden können, um die Testdurchführung für ein gegebenes Testfallset zu planen

FL-5.2.5 (K1) Faktoren benennen können, die den Testaufwand beeinflussen

FL-5.2.6 (K2) Den Unterschied zwischen zwei Schätzverfahren erklären können: das metrikbasierte Verfahren und das expertenbasierte Verfahren

5.3 Testüberwachung und -steuerung

FL-5.3.1 (K1) Testmetriken wiedergeben können

FL-5.3.2 (K2) Zweck, Inhalte und Zielgruppen für Testberichte zusammenfassen können

Lernziele nach Syllabus ISTQB CTFL (2/2)

5.4 Konfigurationsmanagement

FL-5.4.1 (K2) Zusammenfassen können, wie Konfigurationsmanagement das Testen unterstützt

5.5 Risiken und Testen

FL-5.5.1 (K1) Risikostufe anhand der Wahrscheinlichkeit (des Eintritts) und Auswirkung (im Schadensfall) definieren können

FL-5.5.2 (K2) Zwischen Projekt- und Produktrisiken unterscheiden können

FL-5.5.3 (K2) Anhand von Beispielen beschreiben können, wie die Produktrisikoanalyse Intensität und Umfang des Testens beeinflussen kann

5.6 Fehlermanagement

FL-5.6.1 (K3) Einen Fehlerbericht schreiben können, der einen während des Testens gefundenen Fehler enthält

6.1.1 Testorganisation – Unabhängiges Testen (1/3)

Modelle bzw. Optionen einer Aufgabenteilung zwischen Entwicklern und Testern:

1. **Entwicklertest**
   Es gibt keine separate Rolle «Tester». Testen liegt ausschliesslich in der Verantwortung des einzelnen Entwicklers. Jeder Entwickler testet seine eigenen Programme.
2. **Unabhängiger Tester**
   Es gibt ein Rolle «Tester». Mindestens ein Mitglied des Entwicklerteams ist für Testarbeiten abgestellt. Es erledigt alle Testarbeiten des Teams.
3. **Unabhängige Testteams**
   Es gibt dedizierte Testteams innerhalb des Unternehmens oder des Projekts mit eigener Team- oder Teilprojektleitung, die an die über geordnete Unternehmens(bereichs)leitung oder das jeweilige Projektmanagement berichten.

6.1.1 Testorganisation – Unabhängiges Testen (2/3)

4. **Testspezialisten**
   Ein Pool von Mitarbeitern mit Ausbildung und Spezialisierungen auf bestimmte Testthemen (Branchen-, Anwendungs-, Fachabteilungswissen) oder Testarten (z.B. Test nicht funktionaler Eigenschaften wie Performanz, Benutzbarkeit, Sicherheit oder für den Nachweis der Konformität mit Standards und Regularien), die zeitweise oder fallweise von Projekten angefordert werden, um spezielle Testaufgaben zu erledigen oder beratend zur Seite stehen und methodisch unterstützen (Training, Coaching, Testautomatisierung u.Ä.).
5. **Testdienstleister**
   Ein externer Dienstleister übernimmt alle oder wesentliche Teile der Testaufgaben des Unternehmens oder einzelner Projekte (z.B. den Systemtest). Die Arbeiten können vor Ort (Insourcing) oder ausserhalb des Betriebs (Outsourcing) ausgeführt werden.

6.1.1 Testorganisation – Wann welches Modell? (1/4)

Welches der oben genannten Modelle zweckmässig ist, hängt in hohem Mass von der betrachteten Teststufe ab:

- **Komponententest**
  Hier muss relativ entwicklungsnah gearbeitet werden. Daher ist meistens ein Entwicklertest nach Option 1 anzutreffen. «Gegenseitiges Testen» kann in den meisten Fällen organisiert werden und bringt sicher Qualitätsgewinne. Modell 2 ist sinnvoll, sofern in Relation zur Entwicklungsmannschaft genügend viele Tester benannt bzw. abgestellt werden.
- **Integrationstest**
  Sofern Komponenten zu integrieren und zu testen sind, die vom selben Team entwickelt wurden, kann ein Vorgehen analog zum Komponententest organisiert werden (Modelle 1, 2). Müssen Komponenten zusammengeführt werden, die von verschiedenen Teams stammen, sollte ein gemischtes Integrationsteam mit Vertretern aus den betroffenen Entwicklungsgruppen oder ein unabhängiges Integrationsteam tätig werden. Je nach Grösse des Entwicklungsvorhabens und Anzahl der Komponenten kommen hier die Modelle 3–5 infrage.

6.1.1 Testorganisation – Wann welches Modell? (2/4)

- **Systemtest**
  Das Gesamtprodukt muss aus dem Blickwinkel des Kunden und des späteren Anwenders betrachtet werden. Unabhängigkeit von der Entwicklung ist hier unabdingbar. Damit kommen nur die Modelle 3, 4 und 5 in Betracht.

Natürlich bestimmt auch das im Unternehmen oder Projekt angewendete Softwareentwicklungsmodell, wie unabhängiges Testen realisiert wird.

6.1.1 Testorganisation – Wann welches Modell? (3/4)

- Agile Vorgehensweisen betonen im Gegensatz dazu die enge, funktionsübergreifende («cross functional») Zusammenarbeit in kleinen Teams.
- Dies wird von agilen Teams manchmal so interpretiert, dass unabhängig organisiertes Testen «nicht agil ist» oder «die Agilität behindert».
- Solche Teams verwenden dann nur Modell 1 oder Modell 2, in der Annahme, nur diese wären «agil».
- Damit verzichten diese Projekte auf die Vorteile von unabhängig organisiertem Testen nach den Modellen 3 – 5.
- Es ist aber durchaus möglich, die Modelle 3 - 5 auch in agilen Projekten einzusetzen, ohne Agilität zu verlieren! Im Gegenteil: richtig eingesetzt gewinnt das Team Agilität.

6.1.1 Testorganisation – Wann welches Modell? (4/4)

- Die in agilen Projekten geforderten kurzen Iterationen benötigen automatisiertes Testen. Teammitglieder oder Dienstleister, die sich auf Testautomatisierung spezialisiert haben, können das professioneller und effizienter leisten. Für Aufbau und Pflege einer Toolunterstützung für «Continuous Integration» gilt dies analog. Eine Organisation nach Modell 2 oder die Zusammenarbeit mit externen Dienstleistern nach Modell 5 unterstützt und ermöglicht das am besten.
- Grössere Projekte können Effizienz gewinnen, indem sie Testautomatisierungs- und Integrationsaufgaben für mehrere Komponenten bündeln und dazu an ein Serviceteam nach Modell 3 oder 5 übertragen.
- Das agile Team kann und sollte interne oder externe Testspezialisten (Modell 4 und 5) als Coach nutzen, um seine Testmethoden und das Testdesign regelmässig zu überprüfen und zu verbessern. Viele nicht funktionale Tests werden nur in manchen Iterationen vorgesehen und durchgeführt.
- Die Product Owner, ggf. unterstützt durch Mitarbeiter aus betroffenen Fachbereichen (Modell 4), führen am Ende einer Iteration (explorative) Abnahmetests durch und validieren das entstandene Inkrement gegen die ursprünglichen User Stories oder Use Cases.

6.1.2 Rollen, Aufgaben und Qualifikationen

- **Testmanager (Testleiter)**
  - Gestaltet und leitet die Testaktivitäten eines oder mehrerer Entwicklungsprojekte (Kompetenz: Qualitätsmanagement, Projektmanagement, Personalführung)
- **Tester**
  - Testdurchführung und Erstellung der Fehlerberichte (IT-Grundlagen, Testgrundlagenwissen, Bedienung der eingesetzten Testwerkzeuge, Verständnis des Testobjekts)
- **Testdesigner (Testanalyst)**
  - Erstellung der Testfälle und der Testspezifikation, Priorisierung (Domänenwissen, Anwenden der Testmethoden zur Herleitung der Testfälle, Dokumentation)
- **Testautomatisierer**
  - Testautomatisierung (Testgrundlagenwissen, Programmiererfahrung und sehr gute Kenntnisse der eingesetzten Testwerkzeuge)
- **Testadministrator**
  - Installation und Betrieb der Testumgebung (Systemadministration)

6.2 Teststrategie

- Eine so umfangreiche Aufgabe wie das Testen erfordert eine sorgfältige Planung auf operativer Ebene, aber auch auf strategischer Ebene.
- Als Ausgangspunkt der strategischen Planung dienen (sofern vorhanden) die allgemeine Testpolitik, die Testrichtlinie oder die generische Teststrategie des Unternehmens.
- Diese generischen Vorgaben muss der Testmanager unter Beachtung der spezifischen Ziele, Randbedingungen und Risiken des Projekts, der Art, Kritikalität und Risiken des zu testenden Produkts sowie abgestimmt mit einem evtl. vorliegenden Qualitätssicherungsplan in eine für das Projekt angemessene konkrete Teststrategie überführen.

6.2.1 Teststrategie und Testkonzept (1/2)

Diese strategischen Planungsarbeiten des Testmanagers umfassen folgende Aufgaben:

- Testobjekte festlegen
- Testziele formulieren
- Testprozess zuschneiden
- Testmethoden bzw. Testverfahren auswählen
- Testinfrastruktur festlegen
- Testmetriken definieren
- Testberichtswesen definieren
- Kosten und Aufwand planen

6.2.1 Teststrategie und Testkonzept (2/2)

- Das Ergebnis dieser strategischen Testplanung ist die Teststrategie.
- Der Testmanager legt damit die Ziele und Rahmenbedingungen der Testarbeiten im Projekt fest und dokumentiert diese im «Testkonzept» des Projekts.
- Ein gutes Testkonzept zeichnet sich dadurch aus, dass getroffene Entscheidungen begründet werden.
- Teil 3 der ISO-Norm 29119 stellt eine Referenzgliederung bereit.
- Die Teststrategie kann übergreifend formuliert sein und für mehrere Produkte einer Produktfamilie oder für alle Teststufen innerhalb eines Projekts («Master Test Plan») gelten.
- Es kann aber auch zusätzliche, spezifische Testkonzepte für einzelne Teststufen («Level Test Plan»), für einzelne Testobjekte oder auch Testart-spezifische Konzepte bzw. Strategien (z.B. für Gebrauchstauglichkeitstests oder Performanztests) geben.

6.2.2 Testkonzept (Gliederung angelehnt an ISO 29119)

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering

1. Testkonzeptbezeichnung
2. Einführung
3. Testobjekte — was
4. zu testende Leistungsmerkmale / nicht zu testende L.M.
5. Teststrategie — wie
6. Abnahmekriterien
7. Kriterien für Testabbruch bzw. Fortsetzung
8. Testdokumentation
9. Testaufgaben
10. Testumgebung
11. Verantwortlichkeit/Zuständigkeiten/Personal/Training — wer
12. Zeitplan — wann
13. Planungsrisiken/Genehmigung / Freigabe

6.2.2 Auswahl der Teststrategie

- Mit der Wahl der Teststrategie trifft der Testmanager eine seiner wichtigsten Entscheidungen.
- Die möglichen Herangehensweisen zur Strategiefindung lassen sich entlang zweier Dimensionen einordnen:
  - Entscheidungs- und Gestaltungsspielraum
  - Verfügbarkeit an Wissen über Projekt und Produkt

6.2.2 Vorbeugender vs. Reaktiver Ansatz

- Grosser Einfluss auf die Strategie hat der Zeitpunkt, ab dem Tester in ein Projekt involviert werden.
  - **Vorbeugender Ansatz:** Testplanung und Testdesign beginnen so früh wie möglich.
  - **Reaktiver Ansatz:** Testplanung und Testentwurf beginnen erst, nachdem das Softwaresystem bereits erstellt ist.
- Wann immer möglich ist ein vorbeugender Ansatz zu wählen (Kostenfaktor)!

6.2.2 Analytischer vs. Heuristischer Ansatz

- Die zweite Dimension betrachtet, welches Wissen und welche Informationsquellen dem Testmanager zur Verfügung stehen und welche er in der gegebenen Situation nutzen kann.
- **Analytischer Ansatz**
  - Die Planung stützt sich auf Daten und deren systematische Analyse.
  - Die für eine Strategieentscheidung relevanten Einflussfaktoren werden (ausschnittsweise) bestimmt und ihr wechselseitiger Zusammenhang mathematisch modelliert.
  - Testumfang und Intensität werden so gewählt, dass einzelne oder mehrere Parameter wie Kosten, Zeitbedarf, Testabdeckung, etc. optimiert werden.
- **Heuristischer Ansatz**
  - Die Strategiefestlegung stützt sich auf Erfahrungswissen (interner oder externer Experten) und/oder auf «Daumenregeln», weil keine Daten verfügbar sind, weil eine Modellbildung zu aufwendig oder komplex ist oder weil das nötige Know-how fehlt.

6.2.3 Verschiedene konkrete Strategien

- Die in der Praxis üblichen Herangehensweisen sind meist pragmatische Kombinationen der oben dargestellten grundsätzlichen Teststrategie-Ansätze.
- Zum Beispiel kann risikobasiertes Testen (als analytische Strategie) mit explorativem Testen (als reaktives Strategieelement) kombiniert werden.
- Folgende weitere «Spielarten» sind anzutreffen:
  - Kostenorientiertes Testen
  - Risikobasiertes Testen
  - Modellbasierte Vorgehensweise
  - Methodische Vorgehensweise
  - wiederverwendungsorientierten Ansatz
  - checklistenorientierter Ansatz
  - Prozess- oder standardkonforme Ansätze
  - expertenorientierter Ansatz
  - Leistungserhaltende Vorgehensweise

6.2.4 Testen und Risiko (1/2)

- Wenn nach Kriterien gesucht wird, anhand derer Testziele, Testverfahren oder Testfälle ausgewählt und priorisiert werden können, dann ist «Risiko» eines der am besten geeigneten Kriterien.
- Das risikobasierte Testen hat zum Ziel, die Fehler im Programm zu finden, von denen eine hohes Risiko auf den Projekterfolg ausgeht.
- Als Risikoindikator wird in der Regel das Produkt von Fehlerauftrittswahrscheinlichkeit und erwartetem Schaden angenommen.

6.2.4 Testen und Risiko (2/2)

- **Projektrisiken**
  - Lieferantenaspekte
  - Organisatorische Faktoren
  - Schwächen im Entwicklungsprozess
  - Technische Aspekte
- **Produktrisiken**
  - Produkt erfüllt seinen Einsatzzweck nicht wie erwartet
  - Leistungsmerkmale (Features) besitzen nicht die erwartete funktionale Eignung oder nicht funktionale Merkmale (z.B. fehlende/mangelhafte Funktionalität, Zuverlässigkeit, Benutzbarkeit und Performanz)
  - Systemarchitektur skaliert nicht mit wachsenden Anforderungen
  - schlechte Datenintegrität und –qualität
  - Einsatz des Produkts führt zu Schäden an Geräten oder sogar zur Gefährdung von Menschenleben

6.2.4 Priorisierung der Tests

- Risikobasiertes Testen ist die geeignete Teststrategie, um Produktrisiken gezielt zu mindern und zu bekämpfen, und zwar von Beginn des Projekts an.
- Eine der Massnahmen zur risikobasierten Teststeuerung ist die Priorisierung von Testzielen und/oder der Testfälle je nach erwartetem Risiko im Fehlerfall.
- Die Risikobasierte Priorisierung der Testfälle soll so erfolgen, dass bei einer vorzeitigen Beendigung der Tests zu einem beliebigen Zeitpunkt das bis dahin bestmögliche Ergebnis erreicht wird.
- Wichtige Testfälle kommen zuerst zur Ausführung, so dass und schwer wiegende Fehlerwirkungen können frühzeitig erkannt werden.

6.2.5 Testaufwand und Testkosten

- Wesentlichen Einfluss auf die Teststrategie haben die Kosten, die aus der Umsetzung der Teststrategie resultieren.
- In allen diesen Fällen sind die Testkosten gegen die gesteckten Testziele und insbesondere gegen die Risiken, die das Testen mindern muss, abzuwägen bzw. auszubalancieren.
- Einflussfaktoren bei der Aufwandschätzung sind:
  - **Reifegrad** des Entwicklungsprozesses (Stabilität der Organisation, Fehlerhäufigkeit, Änderungsrate der SW, Zeitdruck)
  - **Qualität und Testbarkeit der Software** (Anzahl, Schwere und Verteilung der Fehler in der Software, Komplexität)
  - Testinfrastruktur (Verfügbarkeit)
  - **Mitarbeiterqualifikation** (Know-How der Tester)
  - **Qualitätsziele** (Testabdeckung, Restfehlerrate)
  - **Teststrategie** (Testziele, Testmethoden, zeitliche Planung)

6.2.6 Schätzung des Testaufwandes

- Der Testaufwand, der für ein Projekt zu veranschlagen ist, hängt von den im vorigen Abschnitt genannten Faktoren ab.
- Zwei grundsätzliche Herangehensweisen sind möglich:
  - **Expertenbasierte Schätzung:** Der Aufwand aller Testaufgaben wird Aufgabe für Aufgabe geschätzt.
  - **Metrikbasierte Schätzung:** Der Aufwand wird aus bekannten Aufwandsdaten vergangener oder ähnlicher Projekte abgeleitet.

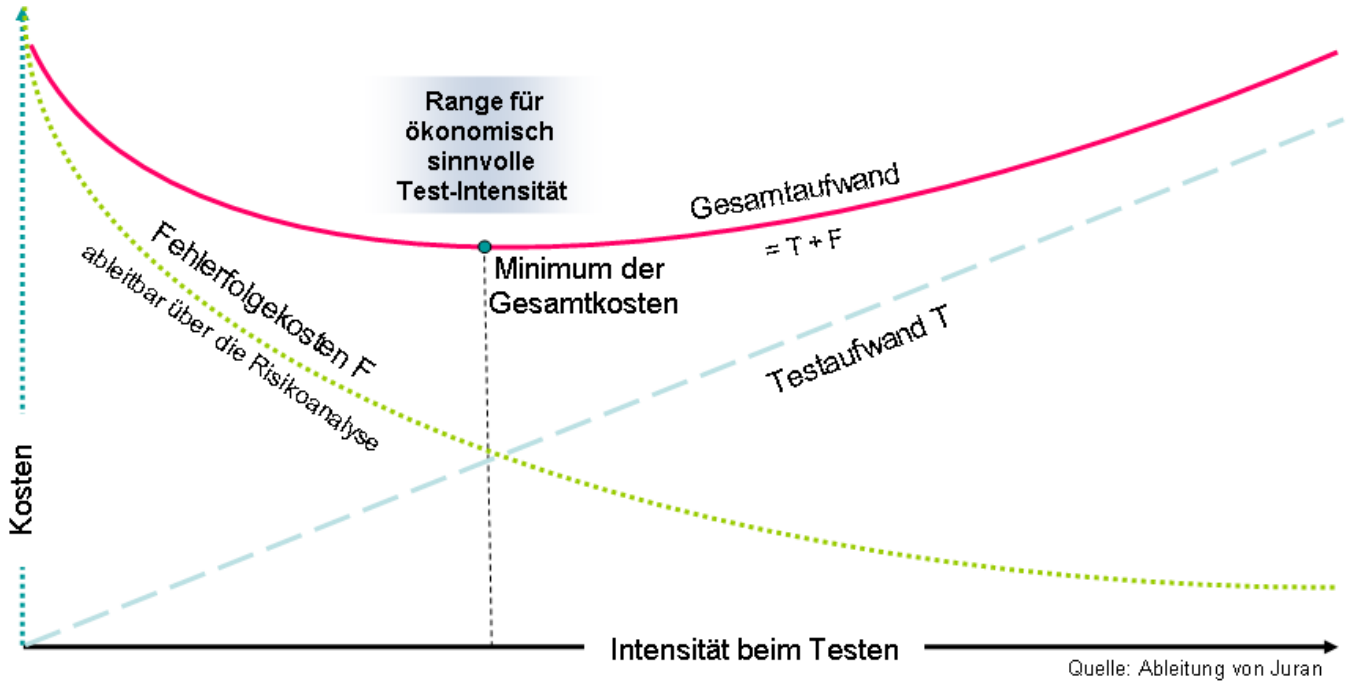
Daumenregel:
Wenn keinerlei Erfahrungswerte verfügbar sind, sollte als initialer Schätzwert für den Testaufwand (über alle Teststufen) von 25% – 50% des Entwicklungsgesamtaufwands ausgegangen werden.

6.2.7 Testkosten vs. Fehlerkosten

- **Direkte Fehlerkosten:** Kosten die dem Kunden durch Fehlerwirkung beim Betrieb des Softwareprodukts entstehen.
  - Z.B. Berechnungsfehler, Fehlbuchungen. Kosten die durch Einspielen neuer Versionen entstehen.
- **Indirekte Fehlerkosten:** Umsatzverlust, weil der Kunde mit dem Produkt unzufrieden ist.
  - Vertragsstrafen oder Minderung
  - Erhöhter Aufwand der Kunden-Hotline
- **Fehlerkorrekturkosten:** Kosten die dem Hersteller durch die Fehlerkorrektur entstehen.
  - Zeit für Fehleranalyse
  - Korrektur
  - Zeit für Regressionstests

6.2.7 Kosten- und Wirtschaftlichkeitsaspekte

- Testen kann aufwändig sein
- Testen kann einen bedeutenden Kostenfaktor darstellen
- Fehlerkosten vs. Testkosten



6.3 Testplanung, Teststeuerung und Testüberwachung

- Die Teststrategie, dokumentiert im Testkonzept, ist das Ergebnis der strategischen Testmanagementarbeit und diese Teststrategie gilt es im Verlauf des Projekts umzusetzen.
- Der Testmanagementprozess gliedert sich in vier Schritte:
  - Festlegung bzw. Justierung der Teststrategie
  - Planung der Testausführung
  - Initiierung und Steuerung der Testausführung
  - Auswertung und Bericht der Testergebnisse
- Der Testprozess wird wiederholt durchlaufen:
  - Für neue, geänderte oder korrigierte Versionen der zu testenden Software,
  - in jeder der vorgesehenen Teststufen (oft parallel zueinander)
  - und bei agilen Projekten in jeder Iteration.

6.3.1 Testausführungsplanung

- Die Teststrategie gibt nur Rahmenbedingungen vor, z.B. welche Testmethoden bzw. Testverfahren einzusetzen sind.
- Sie legt nicht fest, welche Testfälle im Detail zu erstellen und durchzuführen sind.
- Das muss der Testmanager vor oder am Beginn einer Iteration bzw. eines Testzyklus in einer Feinplanung des Testzyklus festlegen.
- Diese Feinplanung bestimmt dann, welche Testfälle zu entwerfen, welche zu automatisieren und welche Testfälle, von welchen Testern, in welcher Reihenfolge auszuführen sind.
- Das Ergebnis dieser Feinplanung ist der «Testausführungsplan» oder «Testzyklusplan» für den anstehenden Testzyklus.
- Faktoren sind zu berücksichtigen: Entwicklungsstand, Testergebnisse, Ressourcen

6.3.1 Kriterien zur Priorisierung (Exkurs)

- **Nutzungshäufigkeit** bzw. **Eintrittswahrscheinlichkeit** einer Fehlerwirkung im Betrieb
  - Funktionen, die viel genutzt werden, sollten intensiver getestet werden
- **Fehlerrisiko**
  - Intensiver Test auf Fehler, die hohen Schaden anrichten
- **(Subjektive) Wahrnehmung**
  - Intensiver Test auf Fehler, die einem Benutzer besonders auffallen und „stören“
- **Priorität der Anforderung**
  - Testfälle zu wichtigen Anforderungen werden bevorzugt getestet
  - Zentrale und architekturbeeinflussende Anforderungen werden bevorzugt getestet

6.3.2 Teststeuerung

- Die Teststeuerung umfasst alle leitenden, initiierenden oder korrigierenden Massnahmen, die unternommen werden, um die (im Testkonzept und im Testausführungsplan) vorgesehenen Testaktivitäten im jeweiligen Testzyklus zu realisieren.
- Die Steuerungsmassnahmen können direkt den Test betreffen, aber auch jede andere Entwicklungsaktivität.
- Folgende Situationen lassen sich unterscheiden:
  - Das Initiieren geplanter Massnahmen: z.B. eingeplante Testaufgaben dem betreffenden Mitarbeiter übertragen und dann im Verlaufsicherstellen, dass die Arbeit startet und Ergebnisse geliefert wer den.
  - Das kurzfristige Reagieren auf Änderungen oder Schwierigkeiten: Beispielsweise kann es notwendig werden, zusätzliche Testressourcen (Personal, Arbeitsplätze, Werkzeuge) anzufordern und einzusetzen, um einen Rückstand der sichtbar wird, aufzuholen.

6.3.3 Testzyklusüberwachung

- Der Zweck der Testzyklusüberwachung ist es, Informationen zu sammeln sowie Feedback und einen Überblick der Testaktivitäten zu liefern.
- Zu überwachende Informationen können manuell oder automatisch gesammelt werden.
- Sie sollten genutzt werden, um den Testfortschritt zu beurteilen und zu messen, ob die Testendekriterien oder die Testaufgaben, die mit einer «Definition of Done» in einem agilen Projekt einhergehen, erfüllt sind.
- Die Überwachung und Erfolgskontrolle der laufenden Testarbeiten geschieht im Idealfall anhand der Testmetriken, die im Testkonzept vereinbart wurden.
- Metriktypen sind: Fehlerbasierte Metriken, Testallbasierte Metriken, Testobjektbasierte Metriken, kostenbasierte Metriken, allgemeiner Arbeitsfortschritt

6.3.4 Testberichte (1/2)

- Der Zweck eines Testberichts ist es, der Projektleitung und ggf. anderen Stakeholdern Informationen über den Verlauf und aktuelle Ergebnisse der Testaktivitäten zusammenzufassend zu kommunizieren.
- Dies geschieht normalerweise routinemässig nach Abschluss einer Iteration bzw. eines Testzyklus oder nach Abschluss einer bestimmten Testaktivität (z.B. Performanztest), soll aber auch spontan beim Auftreten von Problemen erfolgen.
- Ein wichtiges Element des Testabschlussberichts ist die (subjektive) Bewertung des Testmanagers (im Sinne einer Expertenmeinung), ob das Testobjekt freigegeben werden kann.

6.3.4 Testberichte (2/2)

- Die Inhalte eines Testberichtes können variieren, aber die Informationen sind jedoch oft oder meistens enthalten:
  - Liste der Testobjekt(e) was getestet wurde
  - Datum (von ... bis …) wann die Tests durchgeführt wurden
  - Zusammenfassung welche Art Tests auf welcher Teststufe durchgeführt wurden oder wo der Schwerpunkt lag
  - Statistiken zum Testfortschritt in Bezug auf die Endekriterien
  - Statistiken zur Qualität des Testobjekts, insbesondere Fehlerstatus neue/offene/korrigierte Defekte
  - neue/veränderte/bekannte Risiken oder besondere Vorkommnisse Abweichungen vom Plan
  - Tests und Aktivitäten, die für die nächste Berichtsperiode geplant sind
  - (subjektive) Gesamtbewertung des Testobjekts hinsichtlich des erreichten Vertrauens in das Testobjekt

6.4 Fehlermanagement

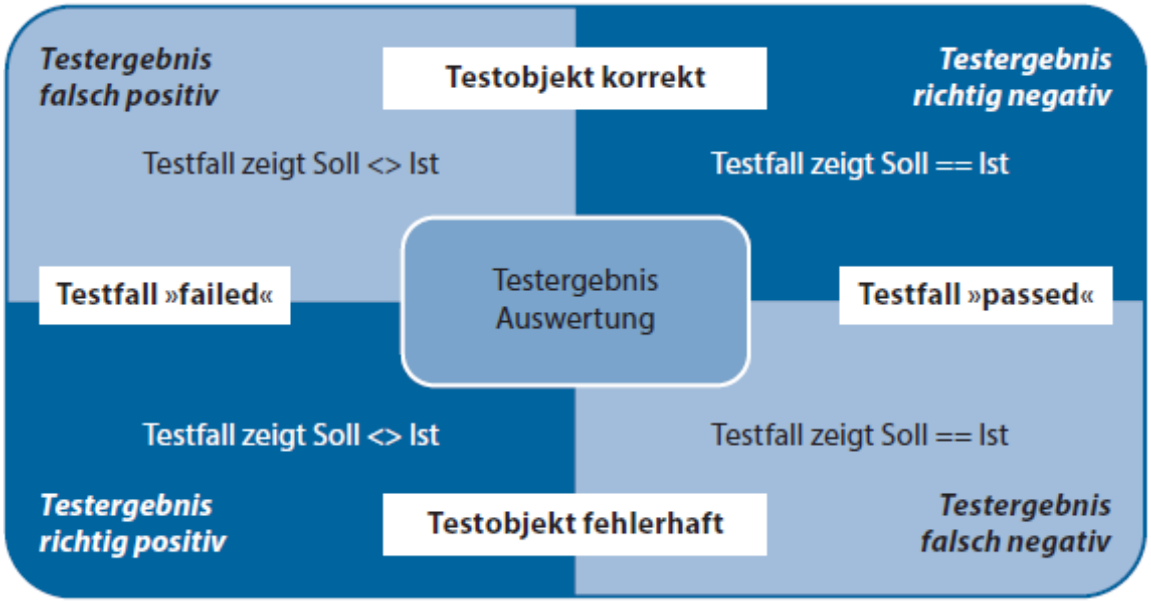
- Damit die durch das Testen aufgedeckten Mängel, Probleme und Fehlerzustände zuverlässig korrigiert werden können, bedarf es eines gut funktionierenden Verfahrens zur Erfassung, Übermittlung und Verwaltung entsprechender Fehlermeldungen.
- Die Gesamtheit der damit verbundenen Tätigkeiten wird als Fehlermanagementprozess oder kurz Fehlermanagement bezeichnet.
- Ein Fehlermanagementprozess besteht aus Vereinbarungen und Regeln, die festlegen, nach welchem Schema jede Fehlermeldung auf gebaut ist, und aus einem Workflow, der definiert, wer die Meldungen wann bearbeiten muss oder darf.
- Inhalt, Umfang und Strenge dieser Prozessfestlegungen kann je nach Unternehmen sehr unterschiedlich sein.
- Eine Fehlermeldung soll grundsätzlich immer dann erstellt werden, sobald eine neue Fehlerwirkung bekannt wird

6.4.1 Testprotokoll auswerten (1/2)

- Ergebnis jeder systematisch durchgeführten Prüfung ist ein Test- bzw. Prüfprotokoll.
- Dieses dokumentiert (je Testfall und/oder je Prüfschritt), welches Istverhalten in der Prüfung jeweils beobachtet wurde, welches Sollverhalten erwartet wurde und ob es zwischen beiden Abweichungen gibt.
- Bei der Auswertung des Protokolls geht es darum, zu analysieren und zu entscheiden, ob und welche der im Protokoll festgehaltenen Abweichungen zwischen «Soll» und «Ist» tatsächlich als Fehlerzustände einzustufen sind.

6.4.1 Testprotokoll auswerten (2/2)

- Diese Einstufung einer Abweichung ist eine Ja-Nein-Klassifikation, bei der folgende Fälle und potenzielle Fehlklassifikationen zu beachten sind:



- Liegt das Problem im Testobjekt (Fall «richtig positiv»), wird über die beobachtete Abweichung eine Fehlermeldung erstellt.

6.4.2 Fehlermeldung erstellen

- Üblicherweise besitzt und führt ein Entwicklungsprojekt eine zentrale Fehlerdatenbank, in der alle Probleme, Mängel oder Fehlerwirkungen, die im Test oder im Betrieb des Produkts entdeckt wurden, erfasst und verwaltet werden.
- Damit diese Kommunikation möglichst reibungsarm funktioniert und die Meldungen auch statistisch sinnvoll ausgewertet werden können, muss jede Meldung nach einem projektweit vorgegebenen einheitlichen Schema aufgebaut sein.

6.4.2 Fehlermeldung (1/3)

- **Kennung**: laufende, eindeutige Meldungsnummer (oft auch als „Ticket“ bezeichnet)
- **Testobjekt**: Bezeichnung des Testobjekts, in welchem System bzw. Subsystem trat der Fehler auf
- **Version**: Identifikation der genauen Version des Testobjekts
- **Testfall**: Beschreibung des Testfalls (Name, Nummer) bzw. der Schritte, die zur Reproduktion der Fehlerwirkung führen
- **Plattform**: Identifikation der HW-/SW-Plattform bzw. der Testumgebung
- **Entdecker**: Identifikation des Testers (ggf. mit Teststufe)
- **Entwickler**: Name des für das Testobjekt verantwortlichen Entwicklers oder Teams

6.4.2 Fehlermeldung (2/3)

- **Erfassung**: Datum, ggf. Uhrzeit, an dem das Problem beobachtet wurde
- **Teststufe**: Komponenten-, Intergations-, System-, Beta- … -Test
- **Problem**: Beschreibung der Fehlerwirkung; erwartete vs. tatsächliche Ergebnisse bzw. Verhalten
- **Kommentar**: Stellungnahmen der Betroffenen zum Meldungsinhalt
- **Status**: Bearbeitungsfortschritt der Meldung; möglichst mit Kommentar und Datum des Eintrags
- **Klasse**: Klassifizierung der Schwere des Problems
- **Priorität**: Klassifizierung der Dringlichkeit einer Korrektur (Auftrittswahrscheinlichkeit und Schadensmass)
  Angabe des fehlerhaften Artefakts („injected in/by“)

6.4.2 Fehlermeldung (3/3)

- **Anforderung**: Verweis auf die (Kunden-) Anforderung, die wegen der Fehlerwirkung nicht erfüllt oder verletzt ist
- **Fehlerquelle**: Soweit feststellbar, die Projektphase, in der die Fehlhandlung begangen wurde (Analyse, Design, Programmierung); nützlich zur Planung prozessverbessernder Massnahmen
- **Verweis**: Querverweise auf andere zugehörige Meldungen
- **Korrektur**: Korrekturmassnahmen des zuständigen Entwicklers mit

6.4.3 Fehlerwirkungen klassifizieren

Klasse	Bedeutung
1	Systemabsturz mit ggf. Datenverlust; das Testobjekt ist in dieser Form nicht einsetzbar.
2	Wesentliche Funktion oder Anforderung nicht beachtet oder falsch umgesetzt; das Testobjekt ist nur mit großen Einschränkungen einsetzbar.
3	Funktionale Abweichung bzw. Einschränkung (»normaler« Fehler); Anforderung fehlerhaft oder nur teilweise umgesetzt; System kann mit Einschränkungen genutzt werden.
4	Geringfügige Abweichung; System kann ohne Einschränkung genutzt werden.
5	Schönheitsfehler (z.B. Rechtschreibfehler oder Mangel im Maskenlayout); System kann ohne Einschränkung genutzt werden.

6.4.3 Fehlerklassifikation - Fehlerpriorität

Priorität	Bedeutung
1 – Patch	Das Problem muss unmittelbar, ggf. provisorisch, behoben werden; ein Patch ist zu erstellen.
2 – nächste Version	Die Fehlerkorrektur erfolgt mit der nächsten regulären Produktversion oder der nächsten Testobjektlieferung.
3 – gelegentlich	Die Fehlerkorrektur erfolgt, sobald die betroffenen Systemteile ohnehin überarbeitet werden.
4 – offen	Die Korrekturplanung ist noch zu treffen.

6.4.4 Fehlerstatus verfolgen (1/3)

- Das Testmanagement muss nicht nur sicherstellen, dass Fehler ordentlich erfasst und verwaltet werden, es muss auch dafür sorgen und über prüfen, dass Korrekturen die gemeldeten Fehlerzustände ausreichend erfolgreich korrigiert haben.
- Hierzu ist eine kontinuierliche Verfolgung des Fehleranalyse- und Korrekturprozesses über alle Bearbeitungsstadien notwendig, von der Entdeckung und Meldung bis hin zur Lösung des Problems
- Diese Fortschrittsverfolgung geschieht anhand des Attributs «Fehlerstatus» und zugehörigen Regeln, die angeben, welche Zustandsübergänge möglich bzw. erlaubt sind.

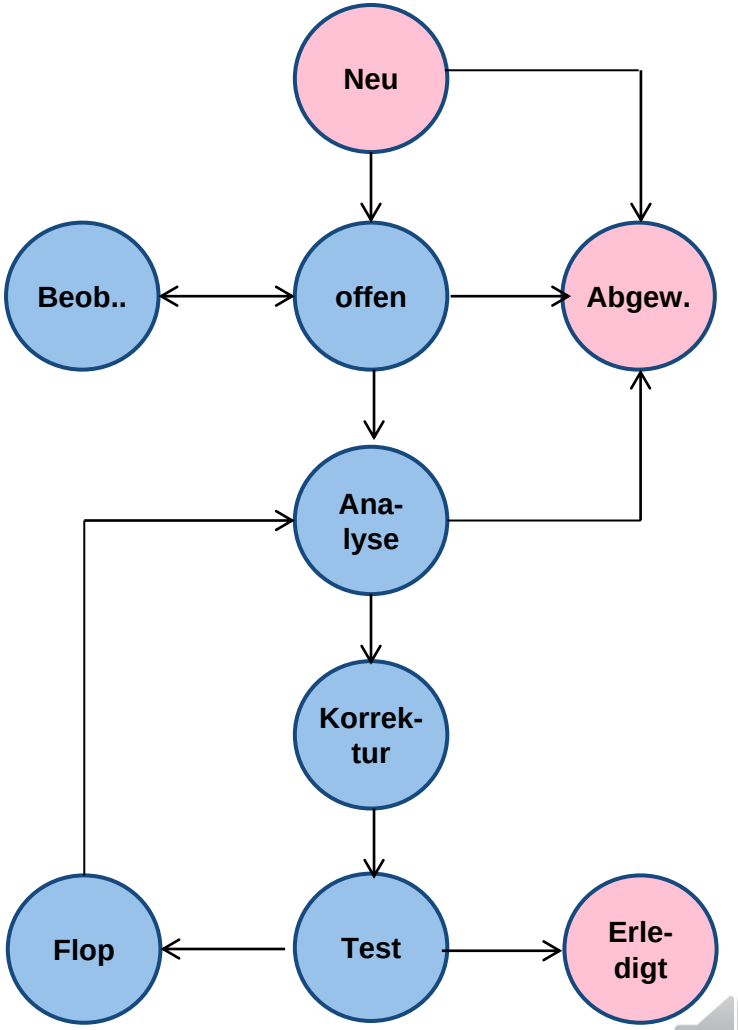
6.4.4 Fehlerstatus verfolgen (2/3)

Status (gesetzt durch)	Bedeutung
Neu (Tester)	Neue Meldung wurde erfasst. Der Verfasser hat eine aus seiner Sicht sinnvolle Beschreibung und Klassifizierung eingetragen.
Offen (Testmanager)	Neue Meldungen werden regelmäßig vom Testmanager gesichtet. Verständlichkeit und vollständige Vergabe der identifizierenden Attribute werden geprüft. Klassifizierende Attribute werden ggf. angepasst, sodass eine projektweit einheitliche Bewertung gegeben ist. Dubletten oder offensichtlich sinnlose oder unberechtigte Meldungen werden »abgewiesen«. Die Meldung wird einem zuständigen Entwickler zugewiesen und auf »Offen« gesetzt.
Abgewiesen (Testmanager)	Meldung ist eindeutig falsch, eine Dublette einer anderen Meldung oder wird als unberechtigt abgewiesen (kein Fehlerzustand im Testobjekt, sondern ein Änderungswunsch, der nicht berücksichtigt wird).
Analyse (Entwickler)	Der zuständige Entwickler setzt die Meldung auf diesen Status, sobald er die Meldung bearbeitet. Das Ergebnis der Problemanalyse (Fehlerzustand und -ursache, Lösungsmöglichkeiten, geschätzter Korrekturaufwand u.Ä.) wird in Kommentaren dokumentiert.
Beobachtung (Entwickler)	Das geschilderte Problem kann weder nachvollzogen noch ausgeschlossen werden. Die Meldung bleibt unerledigt, bis weitere Informationen/Erkenntnisse vorliegen.
Korrektur (Projektmanager)	Aufgrund der Analyse entscheidet der Projektmanager, dass die Korrektur erfolgen soll, und setzt dazu den Status auf »Korrektur«. Der zuständige Entwickler führt die Korrekturen durch. Die Art der Korrektur wird in Kommentaren dokumentiert.
Test (Entwickler)	Der zuständige Entwickler setzt die Meldung auf diesen Status, sobald das Problem aus seiner Sicht behoben ist. Die Softwareversion, in der die Korrektur verfügbar ist, wird angegeben.
Erledigt (Tester)	Problemlösungen, also Meldungen im Status »Test«, werden vom zuständigen Tester im nächstmöglichen Testzyklus verifiziert. Dazu wird mindestens der fehleraufdeckende Test wiederholt. Ergibt dieser Fehlernachtest, dass die Fehlerbeseitigung erfolgreich ist, beendet der Tester die Meldungshistorie im Endzustand »Erledigt«.
Flop (Tester)	Ergibt der Fehlernachtest, dass die Fehlerbeseitigung erfolglos oder ungenügend war, wird »Flop« gesetzt. Eine erneute Analyse ist notwendig.

6.4.4 Fehlerstatus verfolgen(3/3)

- Bsp. Testendekriterien
  - Keine Fehler Klasse 1
  - Alle Fehler der Prio PATCH sind erledigt
  - Neu Meldungen pro Testwoche sinkt

- Viele Änderungen sind keine Fehlerkorrekturen sondern Erweiterungen.
  - Change Control Board (CCB)

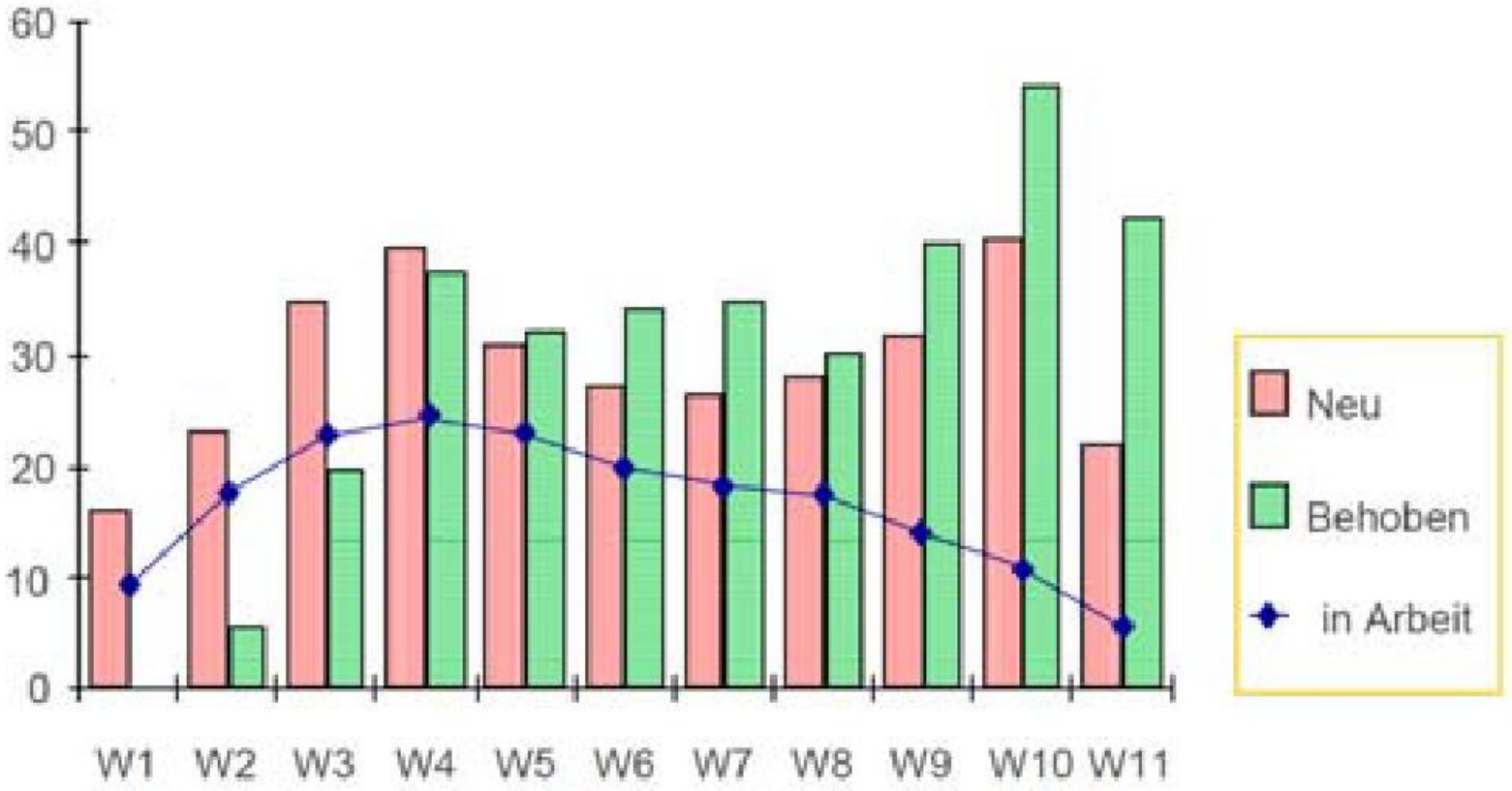


6.4.5 Auswertungen und Berichte

- Test- und Projektmanagement können sich durch Auswertung der Fehlerdatenbank ein Bild über den aktuell erreichten Korrekturfortschritt und die daraus resultierende Produktqualität verschaffen.
- Das Projektmanagement kann anhand der Informationen über den Korrekturfortschritt der Meldungen beurteilen, ob geplante Releasetermine eingehalten werden können oder verschoben werden müssen.
- Die wichtigste Frage in diesem Zusammenhang lautet: «Nehmen die Probleme weiter zu oder ist eine Abflachung der Trendkurve zu erkennen?»

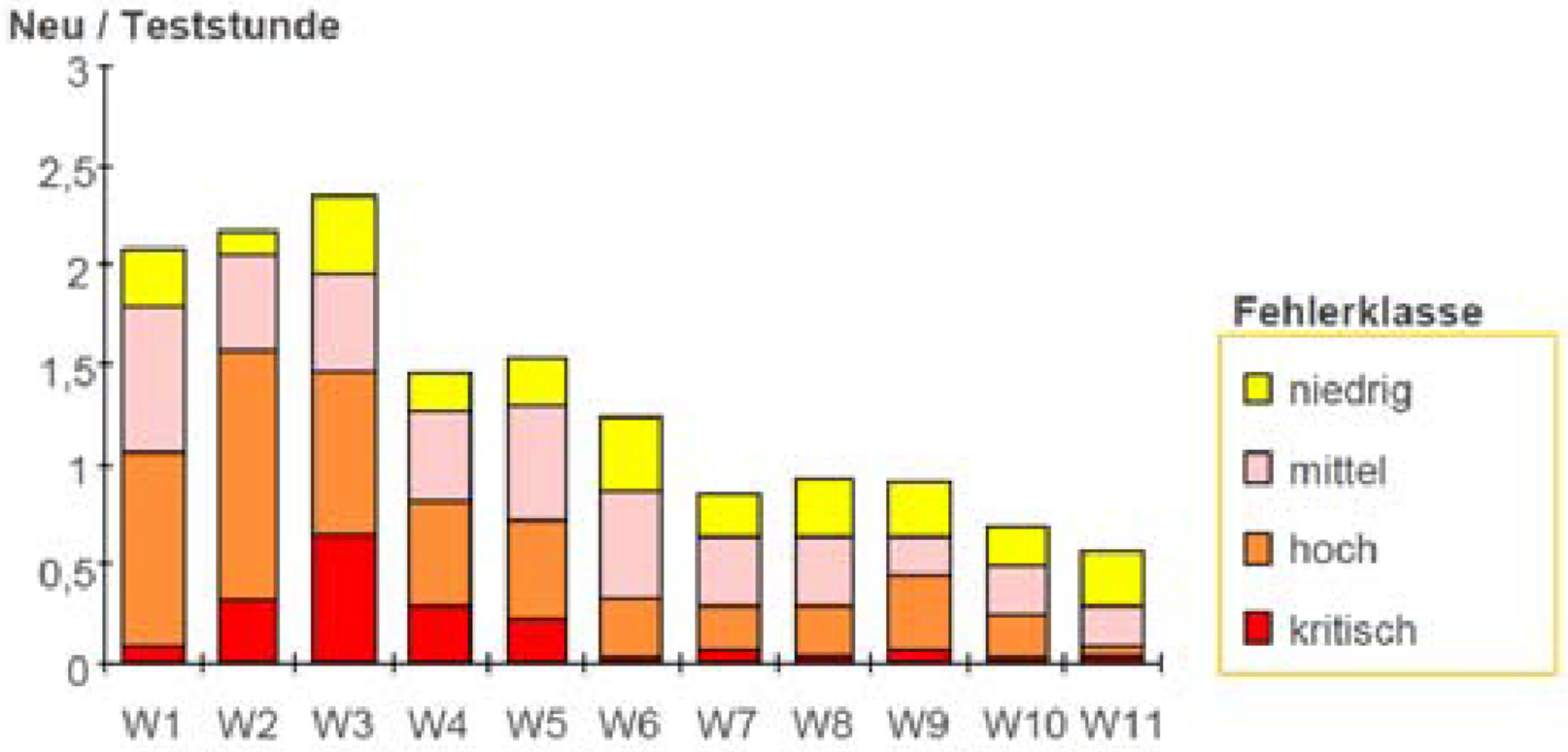
6.4.5 Auswertungen und Berichte – Beispiel 1

- Vergleich neu - Behoben – in Arbeit



6.4.5 Auswertungen und Berichte – Beispiel 2

- Gefundene Fehler nach Fehlerklasse



6.5 Konfigurationsmanagement

- Ein Softwaresystem besteht aus einer Vielzahl von Einzelbausteinen, die zueinander passen müssen, damit das System als Ganzes funktioniert.
- Ziel des Konfigurationsmanagements ist es, die Integrität der Produkte herzustellen und zu erhalten.
- Das betrifft Komponenten, Daten und Dokumentation der Software oder des Systems während des Projekt- und Produktlebenszyklus.
- Für das Testen kann das Konfigurationsmanagement sicherstellen, dass:
  - alle Teile der Testmittel identifiziert und einer Versionskontrolle unterworfen sind sowie Änderungen verfolgt und zueinander und zu den Entwicklungseinheiten (Testobjekten) in Beziehung gesetzt werden, so dass die Rückverfolgbarkeit während des gesamten Testprozesses oder auch des gesamten Produktlebenszyklus erhalten werden kann,
  - alle identifizierten Dokumente und Entwicklungsgegenstände eindeutig in der Testdokumentation referenziert werden.

6.6 Relevante Normen und Standards (Exkurs)

- **Firmenstandards**
  - Qualitätshandbuch, Richtlinien, Rahmentestplan, Programmierrichtlinien
- **Best Practices**
  - Fachlich bewährte Verfahrensweisen
- **Qualitätsmanagementstandards**
  - ISO 9000
- **Branchenstandards**
  - Medizinprodukte DIN EN 60601, Software in Flugzeugen RTC-DO 178B, Bahn-Signalsysteme DIN EN 50128
- **Softwareteststandards**
  - IEEE 29119

Wrap-up (1/2)

- Entwicklungs- und Testaktivitäten sollen organisatorisch getrennt sein. Je klarer diese Trennung erfolgt, umso wirksamer kann getestet werden.
- In agilen Projekten wird diese Trennung zugunsten funktions-übergreifender, enger Zusammenarbeit im Team aufgegeben.
- Die Wirksamkeit des Testens muss hier durch disziplinierte Anwendung geeigneter Testverfahren und Testautomatisierung sichergestellt werden.
- Zu den Aufgaben des Testmanagers gehören die initiale Konzeption und Planung der Tests sowie die anschliessende Planung, Überwachung und Steuerung der einzelnen Testzyklen.
- Seine Teststrategie (Testziele, Testmassnahmen, Werkzeuge usw.) beschreibt der Testmanager im Testkonzept.
- Fehler und Mängel, die im Test übersehen werden, können hohe Fehlerkosten nach sich ziehen.
- Bei der Wahl der Teststrategie wird ein optimales Verhältnis zwischen Testkosten, verfügbaren Ressourcen und drohenden Fehlerkosten angestrebt.

Wrap-up (2/2)

- Um bei Ressourcenmangel schnell entscheiden zu können, welche Tests entfallen können, werden Tests priorisiert.
- «Risiko» ist dabei eines der besten Kriterien zur Priorisierung.
- Risikobasiertes Testen nutzt Informationen über identifizierte Risiken für Planung und Steuerung aller Schritte im Testprozess.
- Messbare Testendekriterien legen objektiv fest, wann das Testen beendet werden kann.
- Ohne klare Testendekriterien besteht die Gefahr, dass die Testarbeiten zufällig abgebrochen werden.
- Fehlermanagement und Konfigurationsmanagement bilden die Basis für einen effizienten Testprozess.
- Fehlermeldungen müssen nach einem projektweit einheitlichen Schema erfasst und über alle Stadien des Fehleranalyse- und Korrekturprozesses verfolgt werden.
- Normen und Standards enthalten Vorgaben und Empfehlungen zur fachgerechten Durchführung von Softwaretests.

Ausblick

- Das Thema der nächsten Vorlesung ist:
  - Testwerkzeuge