Willkommen zur Vorlesung

Methodische Grundlagen des Software-Engineering

im Sommersemester 2012

Prof. Dr. Jan Jürjens

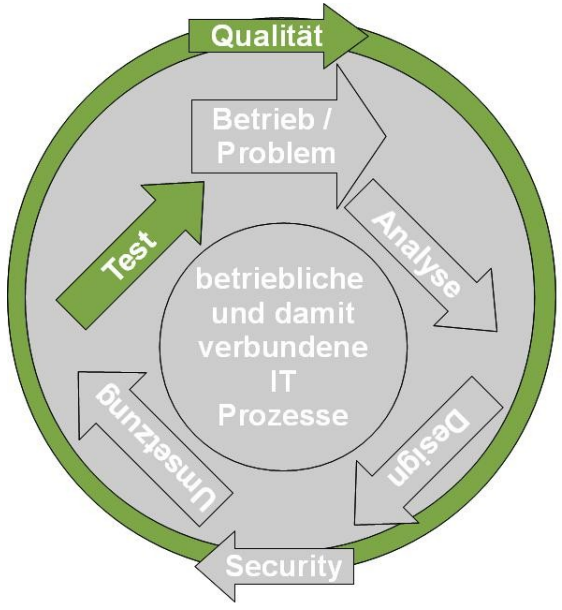
TU Dortmund, Fakultät Informatik, Lehrstuhl XIV

4.6 Test-Management

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
Test-Management

- Geschäfts-Prozesse
- Qualitätsmanagement
- Testen
  - Einführung
  - Grundlagen Softwaretesten
  - Testen im Softwarelebenszyklus
  - Statischer Test
  - Black-Box-Test
  - White-Box-Test
  - Test-Management
  - Testwerkzeuge
  - Fuzzing
- Sicheres Software Design

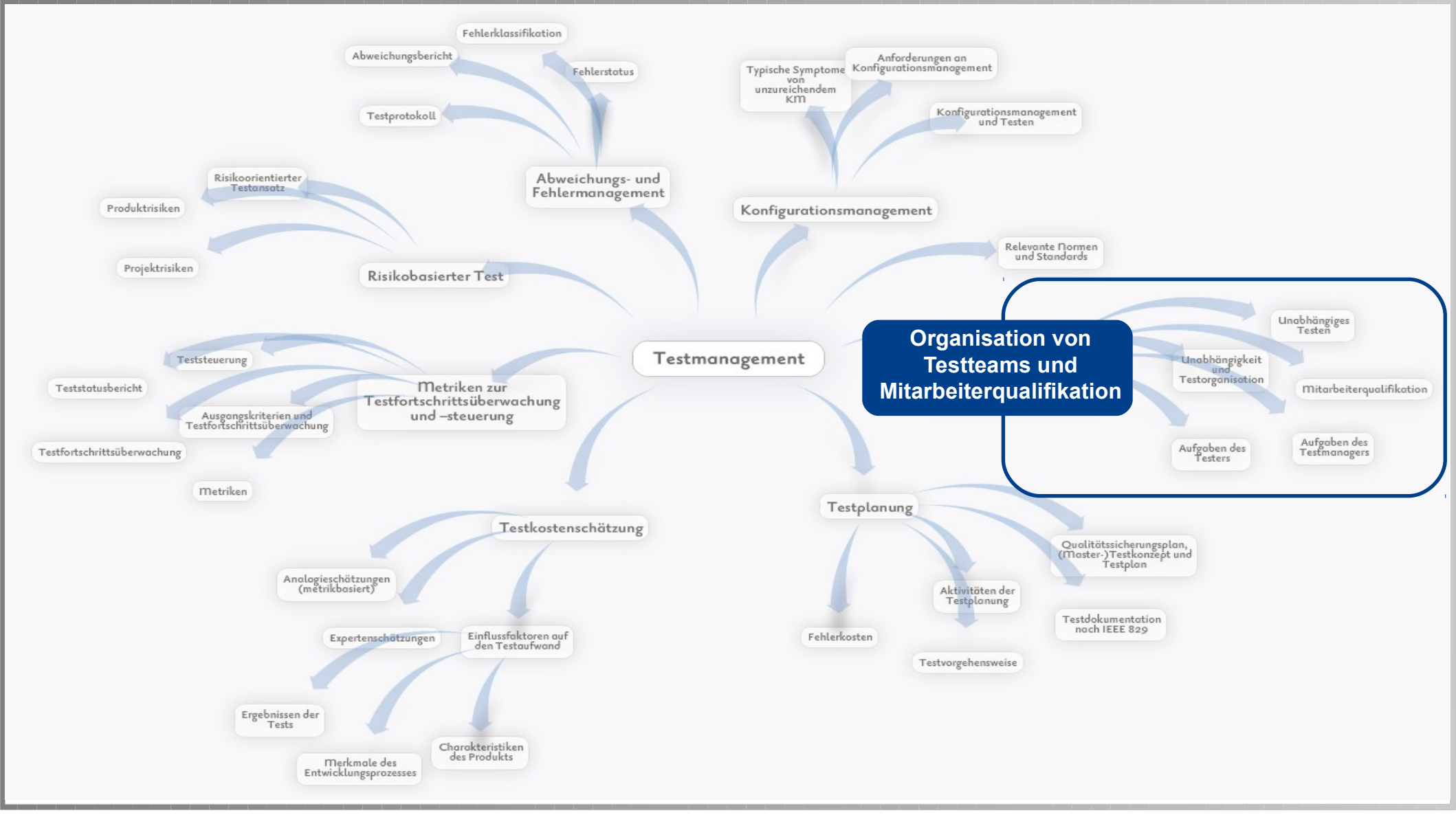


4.6 Testmanagement

4.6 Test-management

- **Organisation von Testteams und Mitarbeiterqualifikation**
- Testplanung und Testkostenschätzung
- Metriken zur Testfortschrittsüberwachung und -steuerung
- Risikobasierter Test
- Abweichungs- und Fehlermanagement
- Anforderungen an das Konfigurationsmanagement
- Relevante Normen und Standards

Organisation von Testteams und Mitarbeiterqualifikation



Unabhängiges Testen

Testaufgaben können übernommen werden von Personen:

- in einer spezifischen Testrolle

  oder

- in einer anderen Rolle
  beispielsweise Projektmanager, Qualitätsmanager, Entwickler, Fach- und Bereichsexperte, Mitarbeiter in Infrastruktur oder IT-Betrieb.

Unabhängige Tester verbessern die Effektivität der Fehlerfindung (4-Augen-Prinzip).

Vor- und Nachteile unabhängigen Testens

Vorteile:

- Unabhängige Tester sind unvoreingenommen und sehen dadurch andere und unterschiedliche Fehlermöglichkeiten, die zu testen sind.
- Unabhängige Tester können (möglicherweise falsche) Annahmen überprüfen, die von den Entwicklern während der Spezifikation und Implementierung des Systems gemacht wurden.

Nachteile:

- Hoher Kommunikationsaufwand durch die Entkopplung der Tester vom Entwicklungsteam (bei vollkommener Unabhängigkeit).
- Ein unabhängiges Testteam kann als letzte Prüfinstanz einen Engpass darstellen (bei schlechter Planung bzw. unzureichender Ausstattung).
- Es besteht die Gefahr, dass die Entwickler die Verantwortung für Qualität nicht mehr ausreichend wahrnehmen und an die unabhängigen Tester delegieren.

Unabhängigkeit und Testorganisation

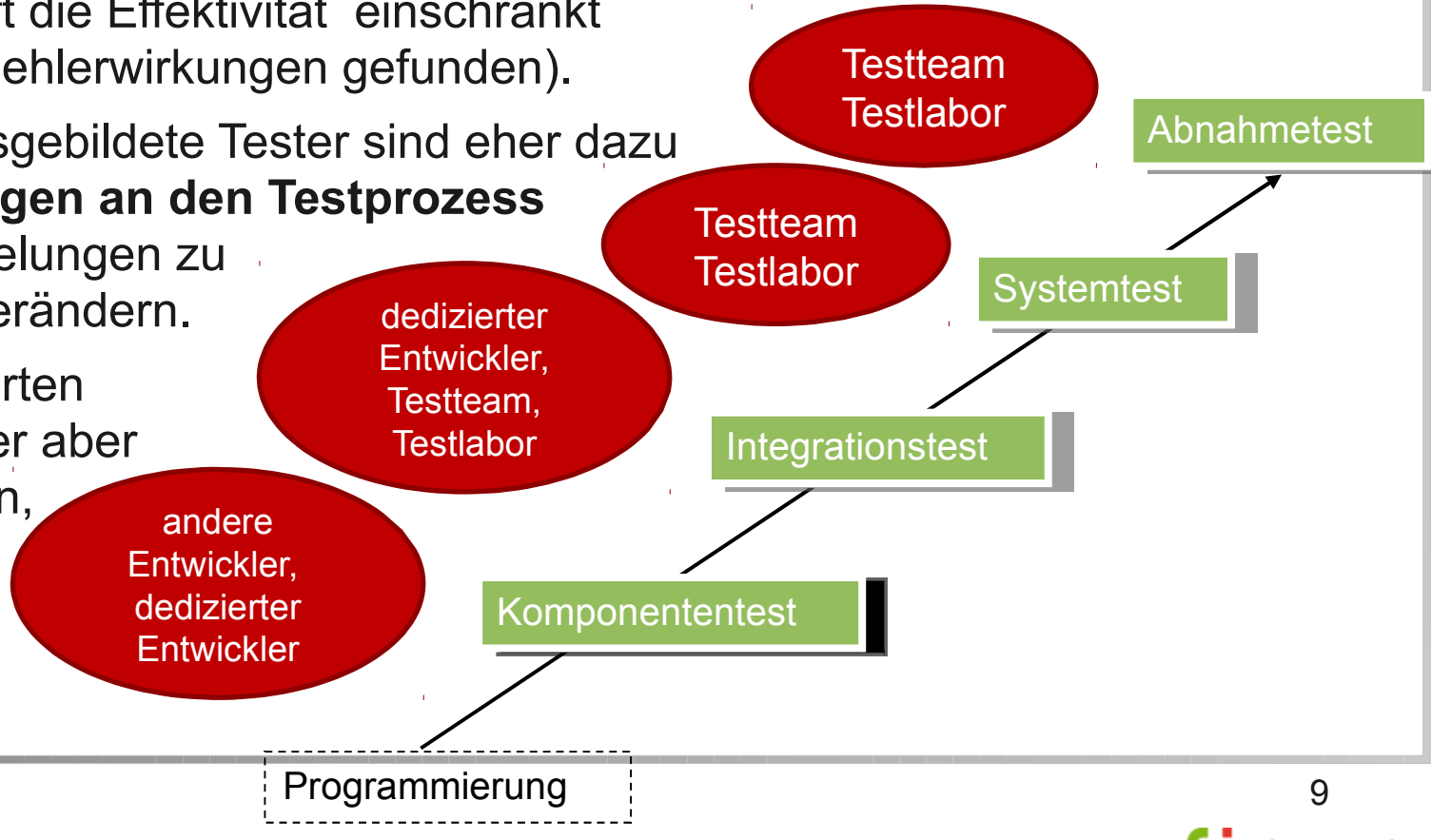
Grad an Unabhängig-keit

Mögliche Organisationsmodelle:

- Jeder **Entwickler** testet den Teil des Programms, den er selbst entwickelt hat.
- Mindestens ein **Mitglied des Entwicklungsteams** ist **ausschließlich** für das Testen verantwortlich („Modultester“).
- **Entwicklungsteam** testet Programmteile **gegenseitig** (Testen wird von einem anderen Entwickler ausgeführt).
- Innerhalb der Organisation gibt es **unabhängige Testteams**, die dem Projektmanagement berichten, aber nicht an der Entwicklung beteiligt sind; d.h. z.B. aus Fach- oder IT-Abteilung oder aus der Gruppe der Anwender werden unabhängige Tester abgestellt.
- Für spezielle Testziele werden **unabhängige Testspezialisten** eingesetzt (Usability-Tester, Sicherheits-Tester), die das Testobjekt gegen gesetzliche Vorschriften und/oder Standards testen.
- **Externe Organisation** (z.B. Testabteilung, externer Testdienstleister, Testlabor, Near-/Offshore) übernimmt das Testen.

Aufgabenteilung zwischen Entwicklung und Test: Tipps

- Je größer, **komplexer** oder sicherheitskritischer das zu testende System ist, je wichtiger ist der Einsatz von **unabhängigen** und ausgebildeten Testern, dabei können Unterschiede bei den einzelnen Teststufen sinnvoll sein.
- **Niedrige Teststufen** (Komponententest, Integrationstest) können unter Beteiligung der Entwickler durchgeführt werden, wobei der Mangel an Objektivität dem eigenen Produkt gegenüber oft die Effektivität einschränkt (es werden weniger Fehlerwirkungen gefunden).
- Unabhängige und ausgebildete Tester sind eher dazu befähigt, **Anforderungen an den Testprozess** aufzustellen und Regelungen zu bestimmen bzw. zu verändern.
- Diese prozessorientierten Aufgaben sollen Tester aber nur dann wahrnehmen, wenn dies vom Management gewollt und unterstützt wird.



Aufgaben und Mitarbeiterqualifikation

Zur Durchführung der Testarbeiten sollen Spezialisten zur Verfügung stehen, deren Know-how alle Aufgabenbereiche im Testprozess abdeckt.

Folgende Rollen sind auszufüllen und im Idealfall mit speziell qualifizierten Mitarbeitern zu besetzen:

- **Testmanager** (Testleiter)
- **Tester** (Testdesigner, Testautomatisierer, Testadministrator, …)

Für Testanalyse, Testentwurf, spezifische Testarten oder Testautomatisierung kann es sinnvoll sein, **Spezialisten** zu haben.

Auch in Abhängigkeit der Teststufe und/oder den Risiken können **verschieden qualifizierte Tester** eingesetzt werden:

- Komponenten- und Integrationsstufe: Entwickler (mit Test Know-how)
- Abnahmeteststufe: Fachexperten und Anwender
- Abnahmetest auf operativer Ebene: Betreiber

Mitarbeiterqualifikation

Neben technischen und test-spezifischen Fähigkeiten benötigen Tester, um erfolgreich zu sein, auch soziale Kompetenz:

- Teamfähigkeit, politisches und diplomatisches Geschick
- Bereitschaft, scheinbare Tatsachen zu hinterfragen
- Durchsetzungskraft
- sicheres Auftreten
- Exaktheit und Kreativität
- Fähigkeit, sich schnell in komplexe Anwendungsgebiete und Applikationen einzuarbeiten

Aufgaben des Testmanagers (1): Umsetzung der Teststrategie

Für die Organisation ist die allgemeine **Testrichtlinie** zu erstellen bzw. zu aktualisieren.

- In der Testrichtlinie werden auf hohem Abstraktionsniveau die Strategie, Vorgehensweisen und wichtigsten projektübergreifenden Ziele einer Organisation in Bezug auf das Testen beschrieben.

Für das jeweilige Projekt ist die **Teststrategie** zu erstellen oder eine vorhandene zu prüfen und ggf. anzupassen.

- Teststrategie ist die abstrakte Beschreibung der Teststufen und der zugehörigen Ein- und Ausgangskriterien. (In der Regel ist eine Teststrategie für mehrere Projekte anwendbar.)

Planung und Umsetzung der Teststrategie ist zu koordinieren (mit Projektmanagern und anderen Entscheidungsträgern).

- Die Aufteilung des Testaufwands über die zu testenden Teile und/oder zu erfüllende Qualitätsmerkmale des Testobjekts.
- Die Auswahl und Reihenfolge (bzw. des Zusammenspiels) von einzelnen Testmethoden, sowie die zu erreichenden Überdeckungsgrade bei den Testmethoden.

Aufgaben des Testmanagers (2)

- Mitwirkung bei anderen Projektaktivitäten, beispielsweise der Integrationsplanung, unter Einbringung der Testanforderungen.
- Planung der Tests:
  - unter Berücksichtigung der Projektgegebenheiten und
  - unter Einbeziehung der Testziele und Risiken,
  - Schätzung der Zeit, des Aufwands und der Kosten des Testens,
  - Beschaffung der benötigten Ressourcen,
  - Festlegung der Teststufen, der Testdurchläufe und der jeweiligen Vorgehensweisen sowie
  - Festlegung der Ziele und Planung des Abweichungsmanagements.
- Einleiten der Analyse und des Entwurfs, der Realisierung und Durchführung von Tests, sowie deren Überwachung und Steuerung bei der Ausführung (Planung und Steuerung des Testprozesses).
- Aktualisierung der Planung auf Grund von Testergebnissen und des Testfortschritts (dokumentiert in den Statusberichten) und Einleitung der erforderlichen Maßnahmen.

Aufgaben des Testmanagers (3)

- Festlegung der konkreten Termine für die Tests
- Erstellung der Testberichte (Teststatus- bzw. Testabschlussberichte) unter Berücksichtigung der aktuellen Informationen (während des Testens gesammelt)
- Einrichtung eines Konfigurationsmanagement der Testmittel
- Auswahl (oder Definition) und Erhebung von Metriken
  - zur Messung des Testfortschritts und
  - zur Bewertung der Qualität des Testens und des Produkts
- Festlegung was, bis zu welchem Grad und wie die Testaufgaben automatisiert werden können und sollen
- Werkzeugauswahl zur Testunterstützung sowie Planung und Organisation notwendiger Schulungen
- Festlegung der Realisierung der Testumgebung

Aufgaben des Testers

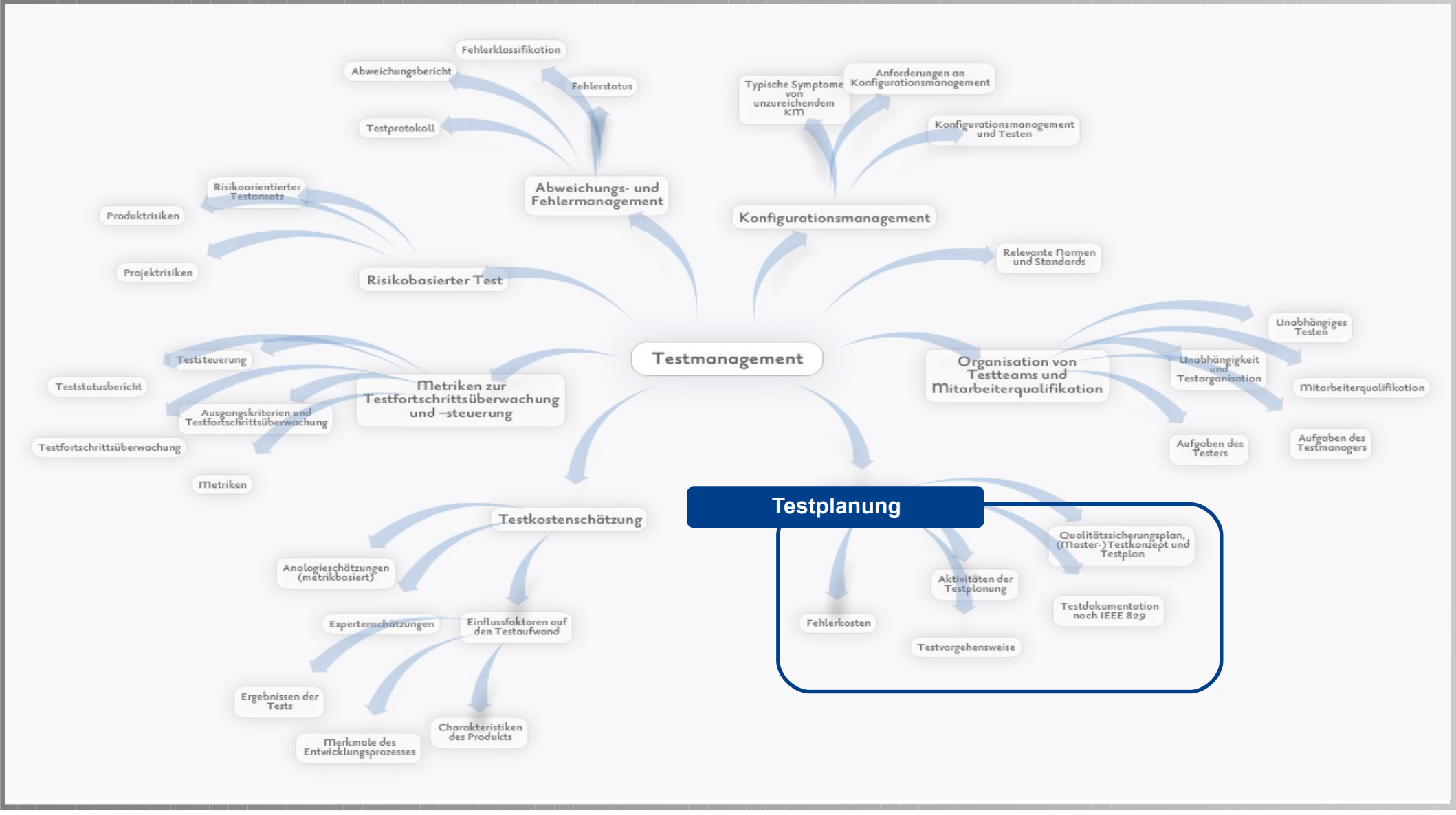
- Anforderungen, Spezifikationen und Modelle des Testobjektes in Hinblick auf Testbarkeit analysieren, prüfen und bewerten
- Mitarbeit an der Erstellung und Prüfung des Testkonzepts
- Erstellen der Testspezifikationen
- Implementierung der Testumgebung (mit Unterstützung der System- und Netzwerkadministration)
- Ausarbeitung (oder Anforderung) von Testdaten
- Umsetzung und Durchführung von Tests auf allen Stufen, sowie Protokollierung, Auswertung der Testergebnisse und Dokumentation der Abweichungen
- Verwendung von Testadministrations-, Testmanagement- und Testüberwachungswerkzeugen (falls gefordert)
- Testautomatisierung (ggf. mit Unterstützung eines Entwicklers oder Testautomatisierungsexperten)
- Performanz von Komponenten und Systemen messen (wenn gefordert)
- Prüfung der Tests, wenn diese von anderen erstellt wurden

4.6 Testmanagement

4.6 Test-management

- Organisation von Testteams und Mitarbeiterqualifikation
- **Testplanung und Testkostenschätzung**
- Metriken zur Testfortschrittsüberwachung und -steuerung
- Risikobasierter Test
- Abweichungs- und Fehlermanagement
- Anforderungen an das Konfigurationsmanagement
- Relevante Normen und Standards

Testplanung



Wann soll mit Testen begonnen werden ?

- So früh im Projekt wie möglich und alle Phasen des Projekts kontinuierlich begleitend.
- Das allgemeine V-Modell sieht deshalb am Ende jeder Entwicklungsphase eine Verifikation des jeweiligen Phasenergebnisses vor.
- Viele Analyse- und Designfehler können so früh gefunden und behoben werden.
- Die Teststufen im rechten Ast des Modells sind als Phasen der Testdurchführung zu verstehen.
- Die zugehörende Testvorbereitung (Testplanung, Testanalyse und Testentwurf sowie ggf. Testrealisierung) startet früher und wird parallel zu den Entwicklungsschritten im linken Ast durchgeführt (s.a. Kap. 4.2, W-Modell).

Qualitätssicherungsplan, Testkonzept und Testplan

- Testen soll nicht als einzige Maßnahme zur Qualitätssicherung (QS) eingesetzt werden, sondern immer im Verbund mit anderen QS-Maßnahmen.
- Die übergreifende Planung der qualitätssichernden Maßnahmen in einem Projekt wird im Qualitätssicherungsplan dokumentiert.
- Die umfassenden Testmaßnahmen sind im **Testkonzept** festzuschreiben: Umfang, Vorgehensweise, Ressourcen und Zeitplanung werden hier festgelegt.
- Im **Testplan** wird die zeitliche Planung der Testdurchführung detaillierter beschrieben (Zuordnung Tester zu Testfällen, Festlegung des Durchführungszeitpunkts).
- Inhalte des Testkonzepts und des Testplans können auch in einem Dokument behandelt werden.
- Bei größeren Projekten kann die Planung in einem **Mastertestkonzept** und in separaten Testkonzepten für Teststufen (z.B. Systemtest oder Abnahmetest) erfolgen.

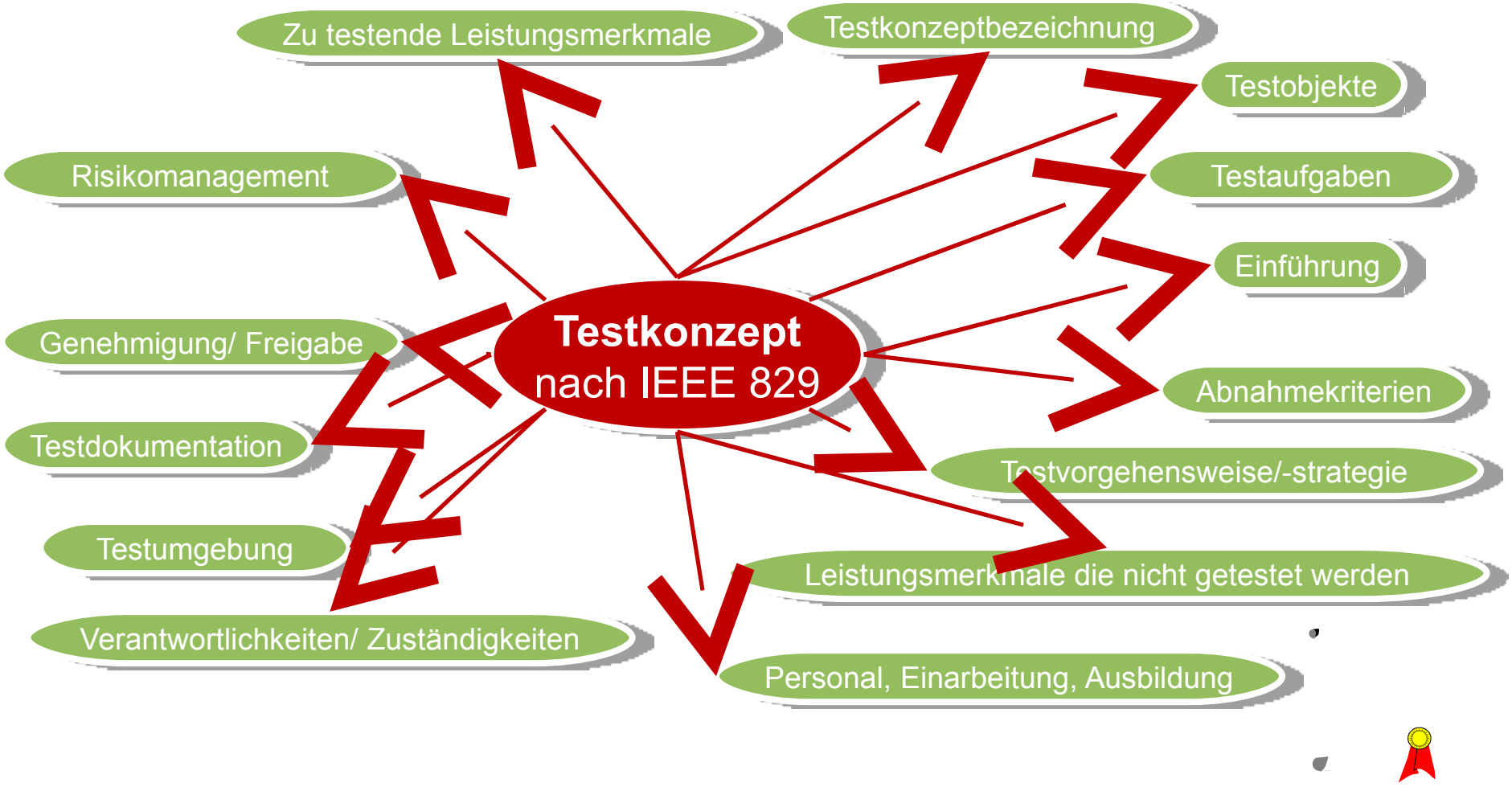
Exkurs: Qualitätssicherungsplan - Gliederung

1. Zweck und Anwendungsbereich
2. Referenzierte Dokumente
3. Projektorganisation und Management
4. Dokumentation
5. Standards, Verfahren, Konventionen, Metriken
6. Software-Reviews
7. Softwaretests
8. Problemmelde- und Korrekturverfahren
9. Werkzeuge, Techniken und Methoden
10. Verwaltung der Medien und Datenträger
11. Lieferantenmanagement
12. Qualitätsaufzeichnungen
13. Trainingsmaßnahmen
14. Risikomanagement
15. Glossar
16. Änderungsverfahren und Änderungsverzeichnis

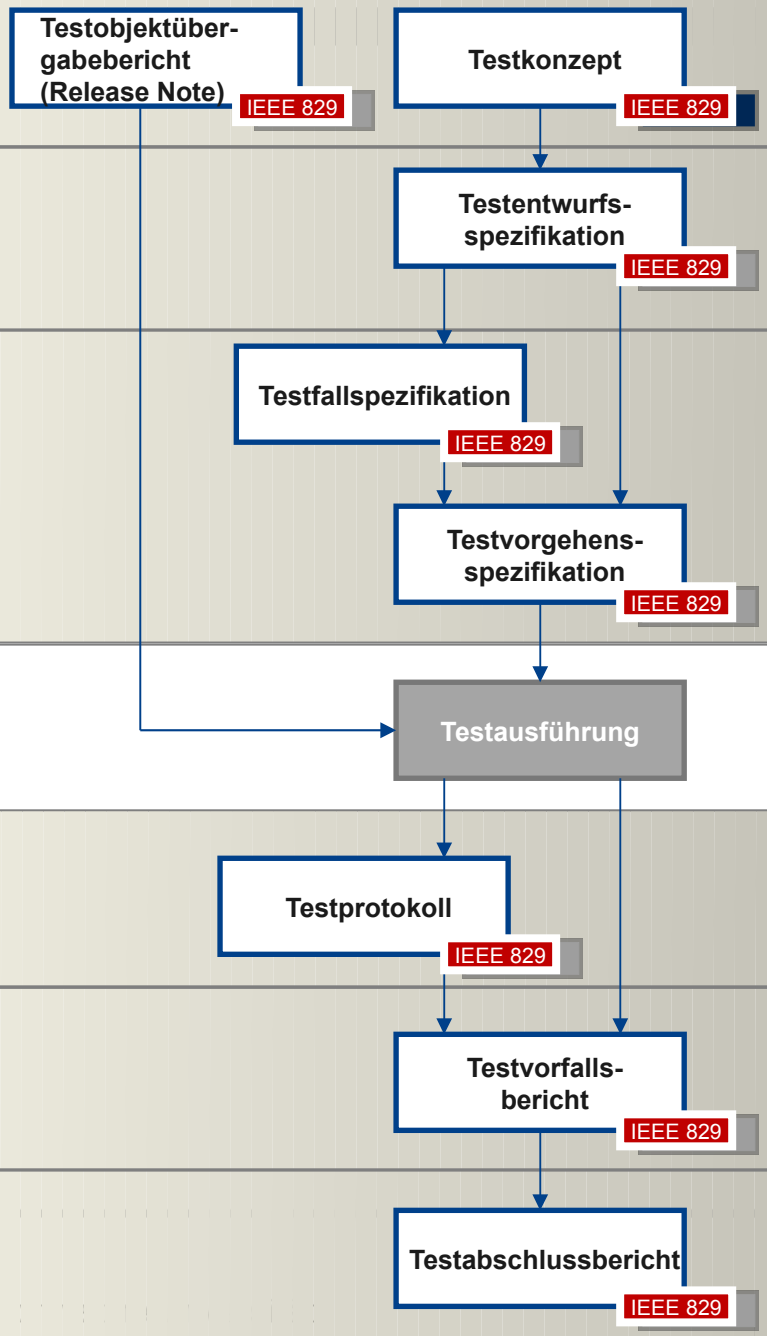
Testkonzept - Gliederung

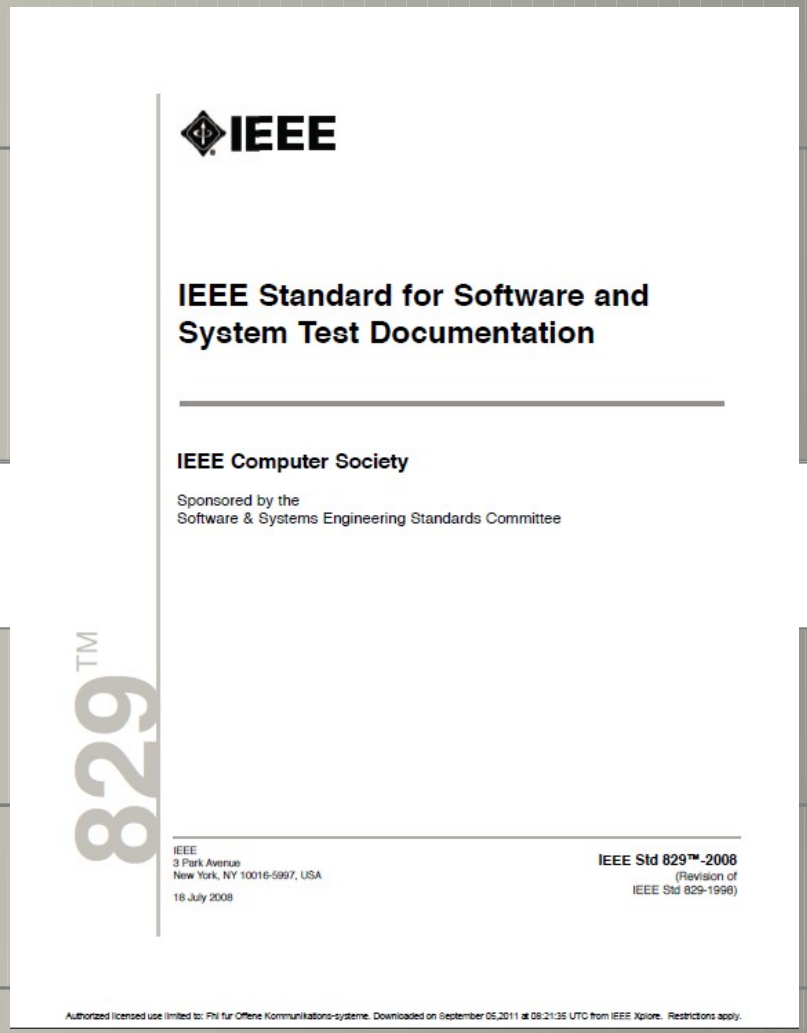
1. Testkonzeptbezeichnung
2. Einführung
3. Testobjekte
4. Zu testende Leistungsmerkmale
5. Leistungsmerkmale, die nicht getestet werden
6. Teststrategie
7. Abnahmekriterien
8. Kriterien für Testabbruch und Testfortsetzung
9. Testdokumentation
10. Testaufgaben
11. Testinfrastruktur
12. Verantwortlichkeiten/Zuständigkeiten
13. Personal, Einarbeitung, Ausbildung
14. Zeitplan/Arbeitsplan
15. Planungsrisiken und Unvorhergesehenes
16. Genehmigung/Freigabe

Testkonzept - Gliederung



Testkonzept - Gliederung





IEEE

IEEE Standard for Software and System Test Documentation

IEEE Computer Society

Sponsored by the
Software & Systems Engineering Standards Committee

829™

IEEE
3 Park Avenue
New York, NY 10016-5997, USA

18 July 2008

IEEE Std 829™-2008
(Revision of
IEEE Std 829-1998)

Aktivitäten der Testplanung

Festlegung der projektspezifischen Teststrategie, der Teststufen, der Eingangskriterien und der Ausgangskriterien (Testendekriterien).

Integration und Koordination der Testaktivitäten mit den Aktivitäten im Softwarelebenszyklus (von der Analysephase bis zum Betrieb und Wartung).

Entscheidung treffen,

- welche Teile wie intensiv zu testen,
- von wem, wann und wie die Testaktivitäten auszuführen,
- wie die Testergebnisse zu bewerten und
- wann die Ausgangskriterien erfüllt sind.

Zuordnung der notwendigen Ressourcen.

Festlegung des Umfangs, Detaillierungsgrads und der Struktur der Testdokumentation (sowie Vorlagen bereitstellen).

Auswahl der Metriken zur Überwachung und Steuerung der Testvorbereitung und -durchführung, Fehlerzustandsbehebung und Risikofaktoren.

Festlegung des Detaillierungsgrades der Beschreibung des Testvorgehens, um eine reproduzierbare Testvorbereitung und -durchführung sicherzustellen.

Testplanung – zu berücksichtigende Punkte

Entwicklungsstand:

- Die zu Beginn des Testzyklus tatsächlich verfügbare Software kann eine gegenüber den ursprünglichen Plänen möglicherweise eingeschränkte oder veränderte Funktionalität aufweisen. Daraus resultieren unter Umständen Anpassungen von Testspezifikationen und Testfällen

Testergebnisse:

- Die in vorangehenden Testzyklen aufgedeckten Probleme machen eventuell eine Änderung der Testpriorisierung notwendig. Korrigierte Fehlerzustände (Defekte) erfordern zusätzliche Fehlernachtests, die ebenfalls neu einzuplanen sind; zusätzliche Tests können auch notwendig sein, weil Probleme noch nicht vollständig reproduziert und analysiert werden konnten

Ressourcen:

- Die Planung des aktuellen Testzyklus muss mit dem aktuellen Projektplan in Einklang stehen; zu beachten sind z.B. Auswirkungen der aktuellen Personaleinsatz- und Urlaubsplanung, die momentane Verfügbarkeit der Testumgebung und spezieller Testwerkzeuge usw.

Zu berücksichtigende Einflussfaktoren (1)

Reifegrad des Entwicklungsprozesses:

- Häufigkeit von Fehlhandlungen der Entwickler
- Rate von Softwareänderungen
- Gültigkeit, Bestand und Aussagekraft von Plänen
- Disziplin bei Konfigurations- und Änderungsmanagement

Testbarkeit der Software:

- Aussagekraft und Aktualität der Dokumentation
- Art der Software und Systemumgebung
- Komplexität der Software

Testinfrastruktur:

- Verfügbarkeit von Testwerkzeugen
- Verfügbarkeit von Testumgebung und Testinfrastruktur
- Verfügbarkeit und Bekanntheit von Testprozess, Standards und Verfahren

Zu berücksichtigende Einflussfaktoren (2)

Mitarbeiterqualifikation:

- Erfahrung und Know-how der Tester bzgl.
  - Testen
  - Testwerkzeugen und Testumgebung
  - Testobjekt
- Zusammenarbeit Tester - Entwickler - Management – Kunde

Qualitätsziele:

- Risikoklassen und Kritikalität
- Testumfang und angestrebte Testabdeckung
- Angestrebte Restfehlerrate bzw. Zuverlässigkeit nach dem Test

Testrichtlinie der Organisation und Teststrategie:

- Anzahl und Umfang der Teststufen (Komponenten-, Integrations-, Systemtest …)
- Wahl der Testmethoden (Black-Box- oder White-Box-Testentwurfsverfahren)
- Zeitliche Planung der Tests (Beginn und Durchführung der Testarbeiten im Projekt bzw. im Softwarelebenszyklus)

Einflussmöglichkeiten des Testmanagers

Reifegrad des Softwareentwicklungsprozesses: Ist eine kurzfristig nicht zu beeinflussende, gegebene Größe; der Reifegrad ist nur langfristig beeinflussbar durch ein Prozessverbesserungsprogramm.

Testbarkeit der Software:

- Hängt stark vom Reifegrad des Softwareentwicklungsprozesses ab.
- Hängt ab vom Ergebnis des Systementwurfs / der Architektur.
- Ein gut strukturierter Prozess mit entsprechenden Reviews führt zu besser strukturierter Software, die einfacher zu testen ist.

Testinfrastruktur: Ist in der Regel vorgegeben, kann aber im Projektverlauf gezielt ausgebaut werden, um punktuell Zeit und Kosten einzusparen.

Mitarbeiterqualifikation: Kurzfristig bedingt beeinflussbar durch Auswahl des Testpersonals, mittelfristig durch Aus- und Weiterbildung.

Qualitätsziele: Werden vom Kunden und anderen Interessengruppen vorgegeben und sind bedingt beeinflussbar (Priorisierung).

Testkonzept / inkl. projektspez. Teststrategie: Liegt im Verantwortungsbereich des Testmanagers und die einzige Stellgröße, die der Testmanager auch kurzfristig beeinflussen und kontrollieren kann.

Testvorgehensweise (1)

Testvorgehensweisen lassen sich basierend auf dem Zeitpunkt, wann mit den Testentwurf begonnen wird, klassifizieren.

- **Präventive Ansätze**
  Tests so früh wie möglich entwerfen
- **Reaktive Ansätze**
  Tests nach der Programmierung entwerfen

Typische Ansätze oder Strategien enthalten:

- **Analytische Ansätze**
  z.B. risikoorientiertes Testen mit der Schwerpunktsetzung auf die Bereiche mit den größten Risiken im Fehlerfall (Analyse des Testobjektes)
- **Modellbasierte Ansätze**
  z.B. stochastisches Testen unter Nutzung von statistischen Informationen über Ausfallraten (wie z.B. Zuverlässigkeitswachstumsmodelle) oder Systembenutzung (wie z.B. Benutzungsprofile)
- **Methodische Ansätze**
  z.B. fehlerbasiertes Testen (Error Guessing), Testen basierend auf Checklisten (erfahrungsbasiert) und Testen der Qualitätsmerkmale

Testvorgehensweisen (2)

Weitere typische Ansätze oder Vorgehensweisen (Strategien):

- **Prozess- oder standardkonforme Ansätze**
  z.B. festgelegt durch Industriestandards oder auch agile Methoden.
- **Dynamische und heuristische Ansätze**
  z.B. exploratives Testen, bei dem keine Planung erfolgt und auf Ergebnisse bzw. Erkenntnisse der bereits durchgeführten Tests reagiert wird.
- **Beratende Ansätze**
  bei denen das Testen durch Hinweise und Beratung von Technologie- und/ oder Geschäftsbereichsexperten (außerhalb des Testteams) gesteuert wird.
- **Wiederverwendungsorientierte Ansätze**
  streben die Nutzung von vorhandenen Tests und Testumgebungen (aus früheren Projekten oder von Standardtestfällen und -umgebungen) an. Umfangreiche Automatisierung der funktionalen Regressionstests.

Eine Kombination der unterschiedliche Ansätze ist sinnvoll, z.B. ein risikoorientierter dynamischer Ansatz.

Auswahl des Testvorgehensweise

Auswahl ist von zentraler Bedeutung und soll die **Rahmenbedingungen** berücksichtigen, einschließlich:

- Risiko des Scheiterns des Projekts
- Risiken für das Produkt und Risiken für Personen, die Umwelt und das Unternehmen bei Produktfehlern bzw. -ausfällen
- Qualifikation und Erfahrung der Personen in den einzusetzenden Techniken, Werkzeugen und Methoden
- Testziel und Auftrag des Testteams
- Für den Entwicklungsprozess einzuhaltende Regularien
- Art des Produkts und des Geschäftsfelds

Mix von Vorgehensweisen auswählen, der unter den gegebenen Randbedingungen ein **optimales Verhältnis** zwischen **Testkosten**, verfügbaren **Ressourcen** und drohenden **Fehlerkosten** darstellt.

Kosten des Testens sollen deutlich niedriger bleiben als Kosten durch nicht beseitigte Defekte / Mängel im Endprodukt. Datenmaterial, das es erlaubt, diese Kosten-Nutzen-Relation zu quantifizieren, besitzt jedoch kaum eine softwareentwickelnde Firma.

Nicht zu testen kann teuer sein

Testen kann sehr aufwändig werden und ein bedeutender Kostenfaktor innerhalb eines Entwicklungsprojekts sein.

Zu fragen ist:

- Wie viel Testaufwand ist für ein bestimmtes Softwareprojekt angemessen ?
- Ab wann übersteigt der Aufwand den möglichen Nutzen ?

Um dies zu beantworten, muss betrachtet werden, welche Fehlerkosten ein Verzicht auf Prüfungen und Tests nach sich zieht.

Zwischen Fehlerkosten und Testkosten gilt es dann abzuwägen.

Fehlerkosten

Werden Prüfungen und Tests im Umfang reduziert oder ganz eingespart, erhöht sich als Folge die Zahl der unentdeckten Fehlerzustände und Mängel. Diese Fehlerzustände und Mängel verbleiben im Produkt und führen ggf. zu folgenden Kosten:

- **Direkte Fehlerkosten:** entstehen dem Kunden durch Fehlerwirkungen beim Betrieb des Softwareprodukts (Hersteller haftet evtl.),
  z.B. Datenverlust, Fehlbuchung, Ausfall oder Schäden an Hardware oder Anlagenteilen, Personenschäden, Kosten durch Einspielen neuer Versionen.
- **Indirekte Fehlerkosten:** sind Kosten bzw. Umsatzverlust für den Hersteller, weil der Kunde mit dem Produkt unzufrieden ist,
  z.B. Vertragsstrafen oder Minderung, erhöhter Aufwand für Kundenhotline und Support, Imageschaden, Verlust des Kunden.
- **Fehlerkorrekturkosten:** entstehen dem Hersteller im Zuge der Fehlerkorrektur,
  z.B. für Fehleranalyse und Korrektur, Regressionstests, erneute Auslieferung und Installation, Nachschulung des Kunden, Verzug bei Neuprodukten (wegen Bindung der Entwicklerkapazität im Wartungsbereich), sinkende Konkurrenzfähigkeit.

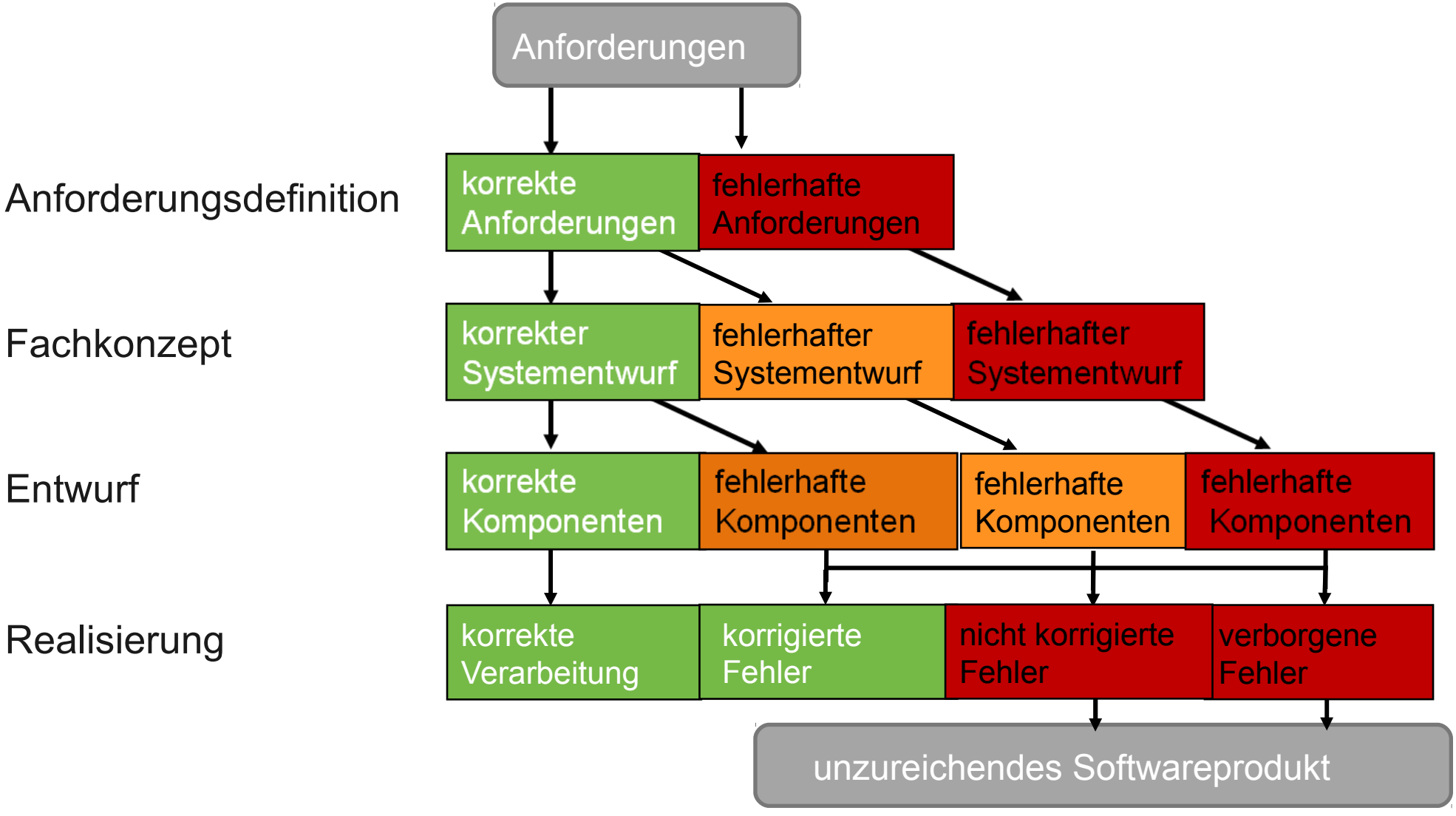
Fehler möglichst früh finden ...

... denn Fehlerkosten steigen rapide über die Entwicklungsphasen an:

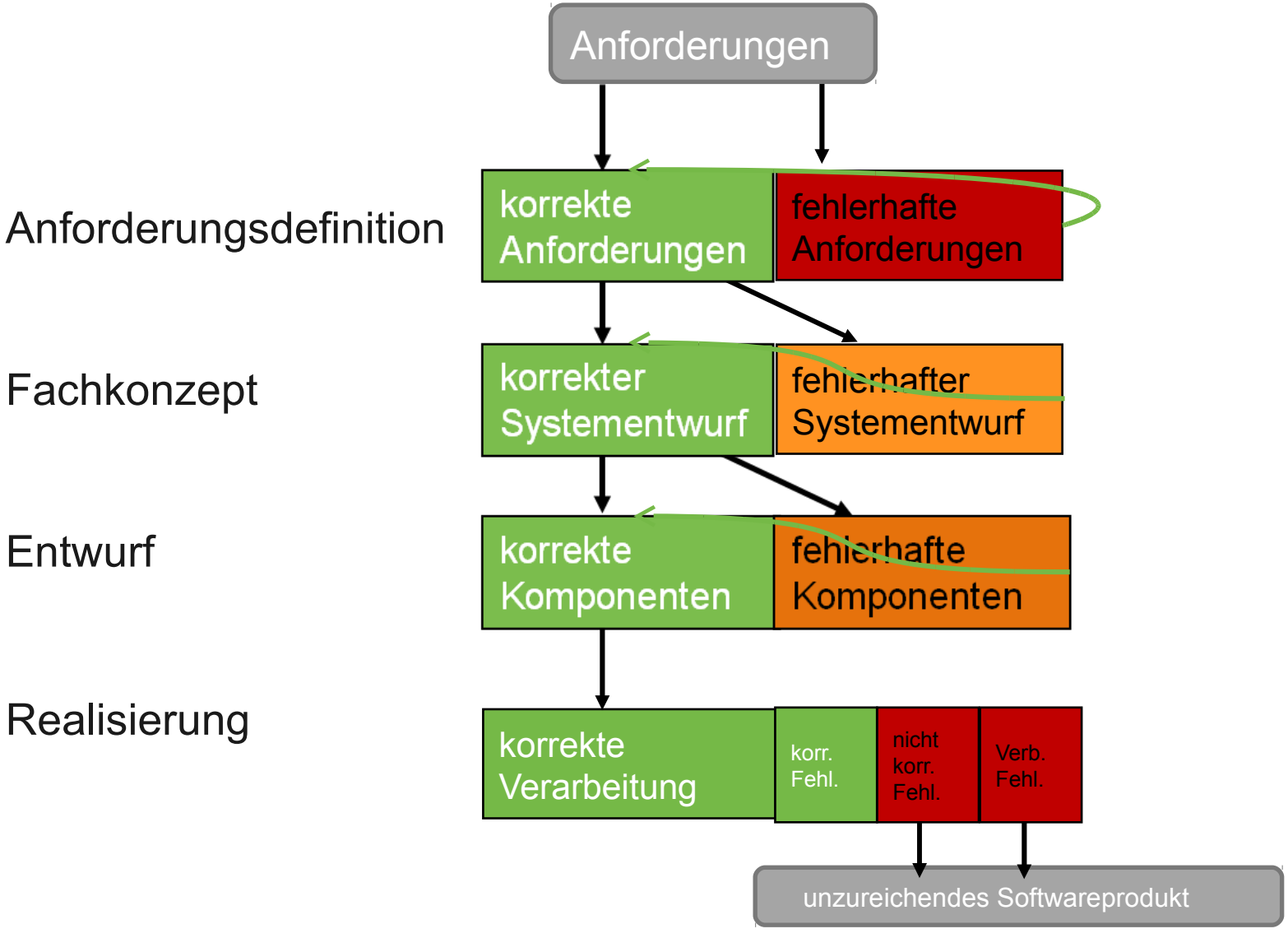
- Ein Fehler, der sehr früh entsteht (z.B. ein Fehler in der Anforderungsdefinition), kann, solange er unentdeckt bleibt, in den anschließenden Entwicklungsphasen viele Folgefehler produzieren (Multiplikation des Effekts des ursprünglichen Fehlers).
- Je später ein Fehler entdeckt wird, umso mehr Korrekturen sind notwendig. Unter Umständen müssen vorangegangene Phasen (Anforderungsdefinition, Design, Programmierung) zumindest teilweise wiederholt werden.

Fehlerrisiko senken oder wenigstens begrenzen durch frühzeitige Prüfungen und Tests.

Fehler und Folgefehler



Fehler und Folgefehler korrigieren und vermeiden

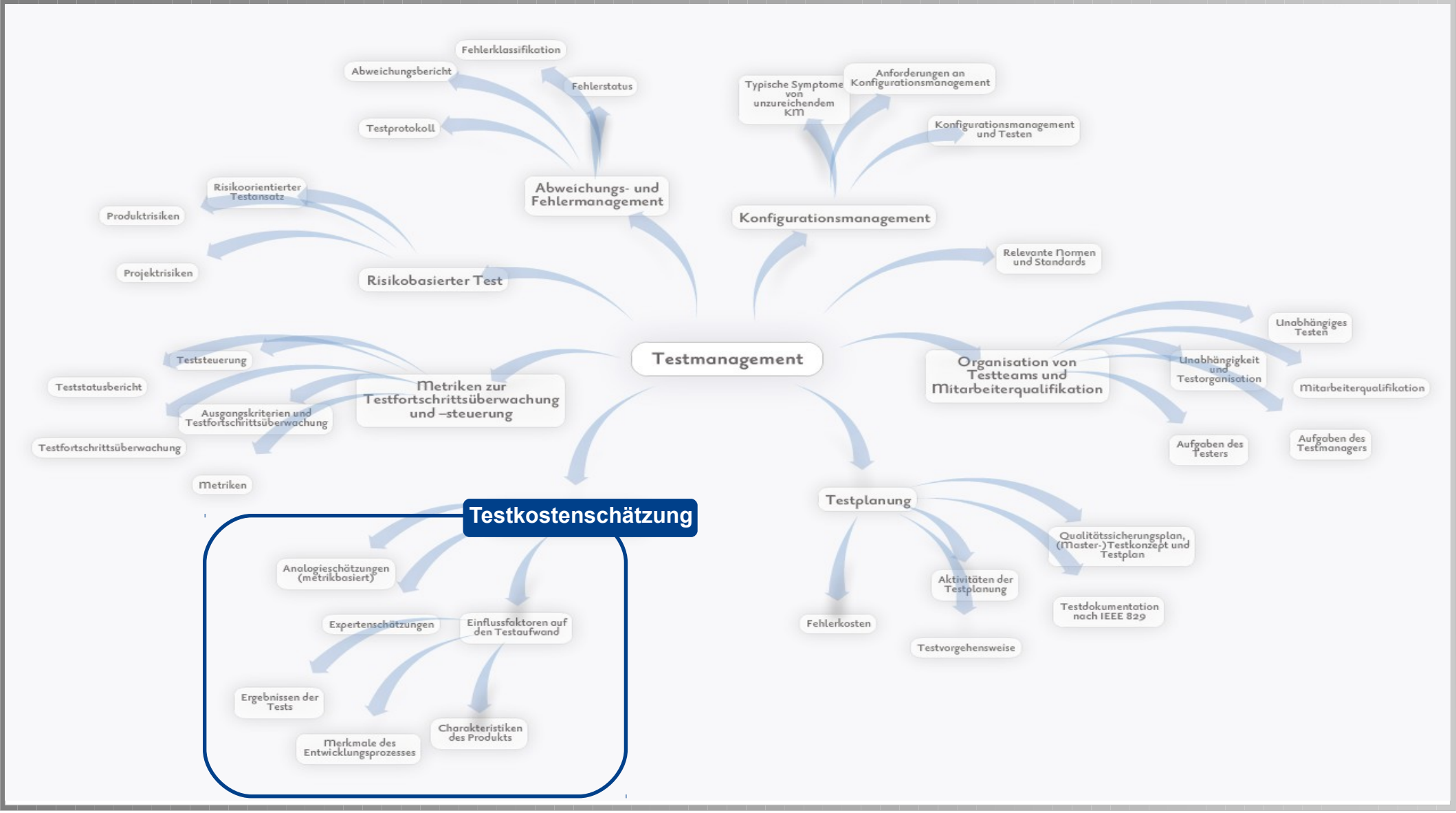


4.6 Testmanagement

4.6 Test-management

- Organisation von Testteams und Mitarbeiterqualifikation
- **Testplanung und Testkostenschätzung**
- Metriken zur Testfortschrittsüberwachung und -steuerung
- Risikobasierter Test
- Abweichungs- und Fehlermanagement
- Anforderungen an das Konfigurationsmanagement
- Relevante Normen und Standards

Testkostenschätzung



Testaufwand

Der Testaufwand hängt von Faktoren ab, u.a.:

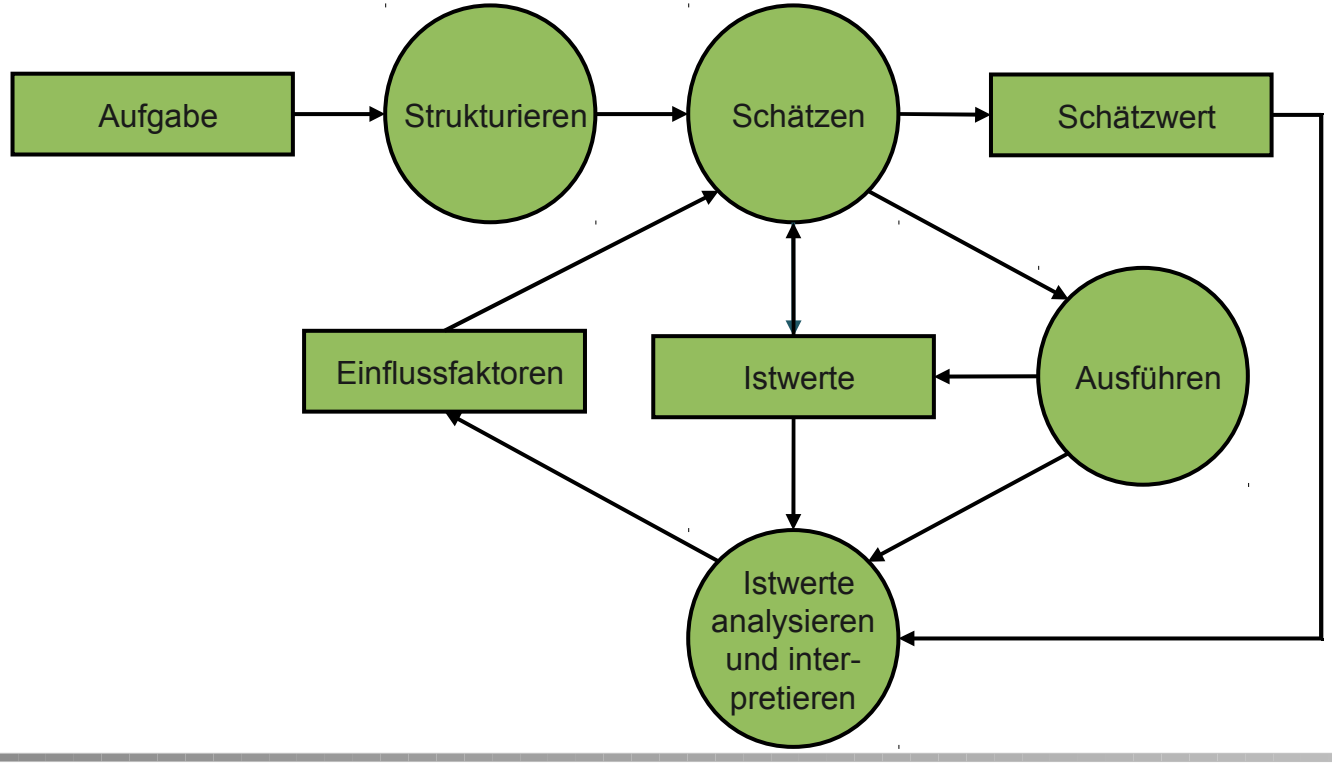
- **von den Merkmale (Charakteristiken) des Produkts:**
  Güte der Testbasis (Spezifikation und weitere Dokumente, die für das Testen herangezogen werden), die Größe und Testbarkeit des Produkts, die Komplexität des Anwendungsbereichs, die Anforderungen an Zuverlässigkeit, Sicherheit und Güte sowie Umfang der Dokumentation.
- **von den Merkmale des Entwicklungsprozesses:**
  Beständigkeit der Organisation und Reifegrad des Entwicklungs- und Testprozesses, Kontinuität bei den benutzten Werkzeugen, der eingesetzten Testinfrastruktur und der jeweiligen Teststrategie, Fähigkeiten der Mitarbeiter und vorhandener Zeitdruck.
- **von den Ergebnissen der Tests:**
  Anzahl der aufgedeckten Fehlerwirkungen und dem damit verbundenem Aufwand für Nacharbeiten (Korrekturen und Fehlernachtest).

Erst wenn der Testaufwand geschätzt ist, können Ressourcen veranschlagt und ein Zeitplan erstellt werden.

Ansätze zur Aufwandsschätzung

Zwei Ansätze für die Schätzung des Testaufwands:

- **Expertenschätzungen:** durch Verantwortliche für diese Aufgaben oder externe Experten
- **Analogieschätzungen:** auf Basis von Metriken früherer oder ähnlicher Projekte oder auf Basis von typischen Werten (metrikbasierter Ansatz).

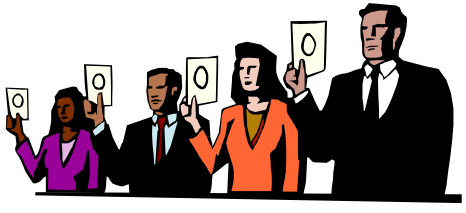


Expertenschätzungen

Einzelschätzung: Experte (oft der Testmanager oder erfahrener Tester) schätzt den Aufwand auf Basis der Aufgabenbeschreibung und seiner Erfahrung

- **Vorteile:** Einfachstes Verfahren. Schnell, kein Rechenaufwand, keine Parameterschätzungen
- **Nachteile:** Hohe Subjektivität, Große Schätzunsicherheit, Keine Transparenz des Schätzprozesses

Mehrfachbefragung: Es werden mehrere Schätzer, möglichst aus unterschiedlichen organisatorischen Bereichen, befragt und ein Mittelwert gebildet oder (bei größeren Unstimmigkeiten) diskutiert und ein Konsens gesucht.



Delphi-Schätztechnik:

- Formalisierte, schriftliche Befragung mehrerer Experten
- Zwei oder mehr Befragungsrunden mit Sitzungen zur Diskussion der Zwischenergebnisse der vorausgegangenen Schätzrunde

Schätzsitzung: Übertragung des Delphi-Prinzips auf eine einzige Gruppensitzung.

Analogieschätzung

Vergleich des zu schätzenden Testprojekts mit einem oder mehreren bereits abgeschlossenen, ähnlichen Projekten:

- Gleiches oder ähnliches Anwendungsgebiet bzw. Aufgabenstellung
- Gleiche oder ähnliche Produktgröße
- Gleiche oder ähnliche Randbedingungen (z.B. Projektteam, SEU u.ä.)

Unterschiede in Aufgabenstellung und Realisierungsbedingungen werden vom Schätzer aufgrund seiner Erfahrung intuitiv berücksichtigt.

Vorteile:

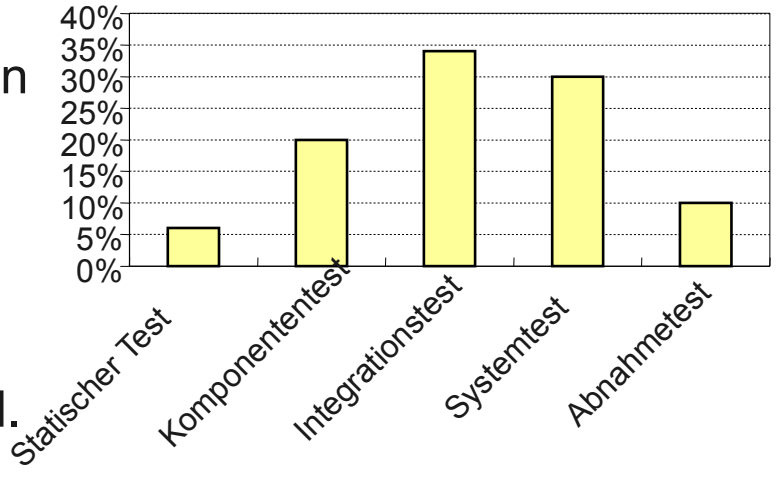
- Schätzungen bereits in sehr frühen Projektstadien möglich.
- Aufwand aus tatsächlichen Projekt-Istwerten abgeleitet.
- Schätzungen sowohl auf Gesamtprojekt- als auch auf Subsystemebene.

Nachteile:

- Schwierige Beurteilung der Repräsentativität der Analogieprojekte.
- Hohe Subjektivität der vorgenommenen Zu- und Abschläge.

Prozentsatzmethode

Aus **früheren Projekten** wird ermittelt, wie sich Aufwand und Projektzeit prozentual auf die einzelnen Testaktivitäten verteilen. Voraussetzung: **Einheitlicher Testprozess.**



Bei Folgeprojekten: **2-stufige Top-Down-Schätzung** des Aufwands.

- 1. Stufe: Schätzung des Testaufwands im Verhältnis zum Gesamtaufwand bzw. zum Entwicklungsaufwand.
- 2. Stufe: Verteilung der Testaufwände auf Teststufen.

Zwei Varianten für **Bottom-Up-Schätzungen**:

- Eine/mehrere Projektphasen sind vollständig abgeschlossen, der Ist-Aufwand wird errechnet und prozentual als Soll für die restlichen Phasen hochgerechnet (insbesondere zur Schätzung der Systemtest- und der Einführungsphase).
- Für eine Phase/Aktivität (z. B. Programmierung) wird eine detaillierte Mikroschätzung durchgeführt und daraus wird auf das Gesamtprojekt geschlossen.

Zu Projektbeginn eher für Kontroll-Checks sinnvoll, da hohes Fehlerrisiko bei Hochrechnung aus niedrigen Anfangswerten besteht.

Prozentsatzmethode: Verteilung des Aufwands (PM)

	Produktgröße			
	2 KDSI	**8 KDSI**	**32 KDSI**	**128 KDSI**
Anforderungsanalyse	6%	6%	6%	6%
Architektur und Grobentwurf	10%	10%	10%	10%
Realisierung	68%	65%	62%	59%
Feinentwurf	26%	25%	24%	23%
Implementierung und Modultest	42%	40%	38%	36%
Integration und Systemtest	16%	19%	22%	25%

KDSI = Kilo Delivered Source Instructions (~KLOC)

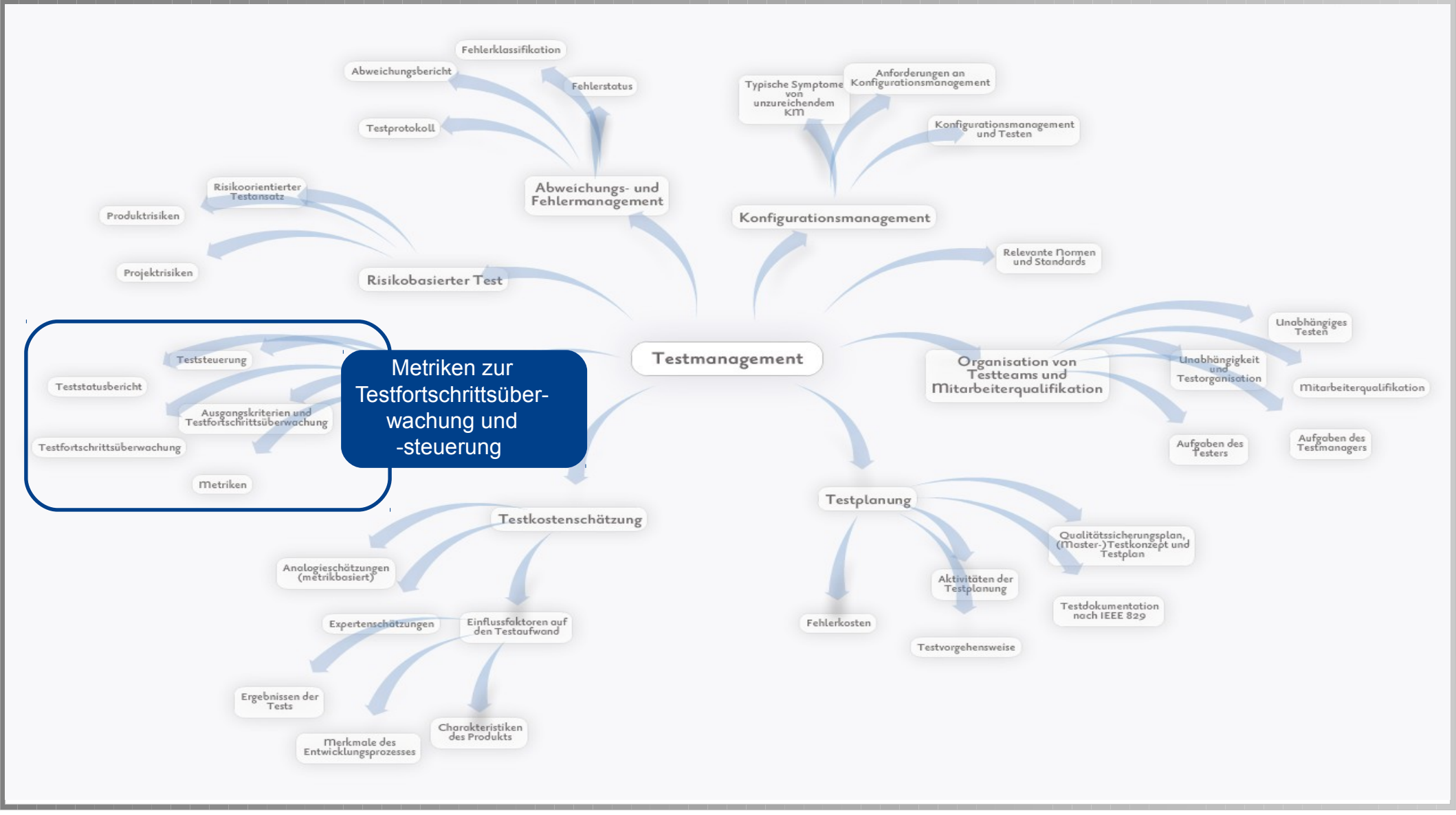
Tabelle in Anlehnung an: Barry Boehm: Software Engineering Economics. Prentice Hall, 1981

4.6 Testmanagement

4.6 Test-management

- Organisation von Testteams und Mitarbeiterqualifikation
- Testplanung und Testkostenschätzung
- **Metriken zur Testfortschrittsüberwachung und -steuerung**
- Risikobasierter Test
- Abweichungs- und Fehlermanagement
- Anforderungen an das Konfigurationsmanagement
- Exkurs: Relevante Normen und Standards

Metriken zur Testfortschritts-überwachung und -steuerung

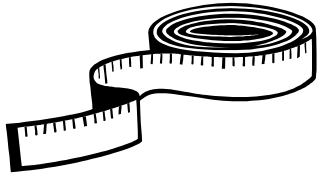


Testfortschrittsüberwachung (1)

Metriken zur Messung und Bewertung von Testfortschritt und Testende

Testmetrik

Messbare Eigenschaft eines Testfalls, Testlaufs oder Testzyklus mit Angabe der zugehörigen Messvorschrift.



Metriken

- Fehlerbasierte Metriken: z.B. Zahl gefundener Fehlerzustände
- Testfallbasierte Metriken: z.B. Anzahl geplanter Tests
- Testobjektbasierte Metriken: z.B. Codeüberdeckung

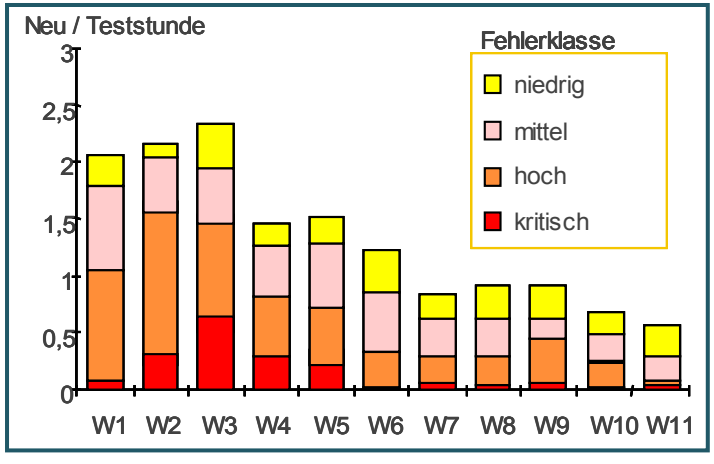
Testfortschrittsüberwachung (2)

Überwachung und Erfolgskontrolle anhand objektiver Testmetriken und darauf aufbauender Ausgangskriterien (Testendekriterien), die im Testkonzept vereinbart wurden.

Dabei ist darauf zu achten, dass nur solche Metriken verwendet werden, die regelmäßig, zuverlässig und einfach zu messen sind, und solche deren Aussagekraft klar sind und wenig Interpretations-spielräume lassen.

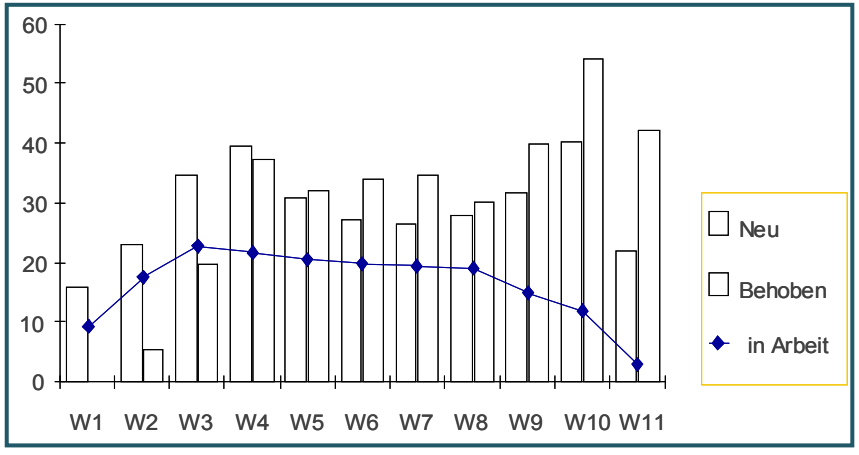
Gebräuchliche Testmetriken sind u.a.:

- **Fehlerbasierte Metriken:** z.B. Fehlerdichte, gefundene und behobene Fehler, Fehleraufdeckungs-rate und Nachtestergebnisse



Testfortschrittsüberwachung (3)

- **Testfallbasierte Metriken**, z.B.
  - Verhältnis der bereits spezifizierten Testfälle zur Gesamtzahl der geplanten Testfälle
  - Anzahl der durchgeführten/nicht durch-geführten Testfälle
  - Anzahl der bestandenen/fehlgeschlagenen Testfälle
  - Verhältnis der durchgeführten Arbeiten zur Bereitstellung der Testumgebung zum Gesamtaufwand der Bereitstellung



- **Testobjektbasierte Metriken**, z.B. Testabdeckung der Anforderungen, der Risiken oder des Codes
- **Kostenbasierte Metriken**, z.B. Vergleich der Kosten für das Auffinden des nächsten Fehlers und den nächsten Testdurchlauf
- **Subjektives Vertrauen der Tester** in das Produkt (nicht messbar !)

Standortbestimmung

Die gemessenen Daten dienen zur Standortbestimmung und zur Beantwortung der Fragen:

- Wie weit ist der Test vorangekommen ?
- Kann der Test beendet und das Produkt ausgeliefert werden ?

Welche Kriterien zur Bestimmung des Testendes sinnvoll und angemessen sind, hängt von den zu erfüllenden Qualitätsanforderungen (Kritikalität der Software) ab, aber auch von den verfügbaren Testressourcen (Zeit, Personal, Werkzeuge).

Die im Projekt geltenden Ausgangskriterien werden ebenfalls im Testkonzept festgelegt.

Jedes Testendekriterium muss so gewählt sein, dass es sich aus den laufend erhobenen Testmetriken berechnen lässt.

Ausgangskriterien und Testfortschrittsüberwachung

Das Ziel von Ausgangskriterien ist die Festlegung, wann mit dem Testen aufgehört wird, z.B. am Ende einer Teststufe oder wenn eine Reihe von Tests ein spezifisches Ergebnis erzielt haben.

Typische Ausgangskriterien sind z.B.:

- Überdeckungsmaße, z.B. Abdeckung von Code, Funktionalität oder Risiko
- Schätzung der (verbleibenden) Fehlerdichte oder Zuverlässigkeitsschätzung
- Umfang der verbleibende Risiken, wie z.B. erkannte aber nicht behobene Fehler oder fehlende Testüberdeckung in einzelnen Softwareteilen
- Kosten
- Zeit, z.B. Termin der Auslieferung oder Markteinführung

Testfortschrittsüberwachung liefert Rückmeldung und Übersicht über Testaktivitäten. Die zur Messung der Ausgangskriterien benötigte Information kann manuell oder automatisiert gesammelt werden. Metriken können ermittelt werden, um den Fortschritt im Zeitplan und die Einhaltung des Budgets zu beurteilen.

Teststatusbericht

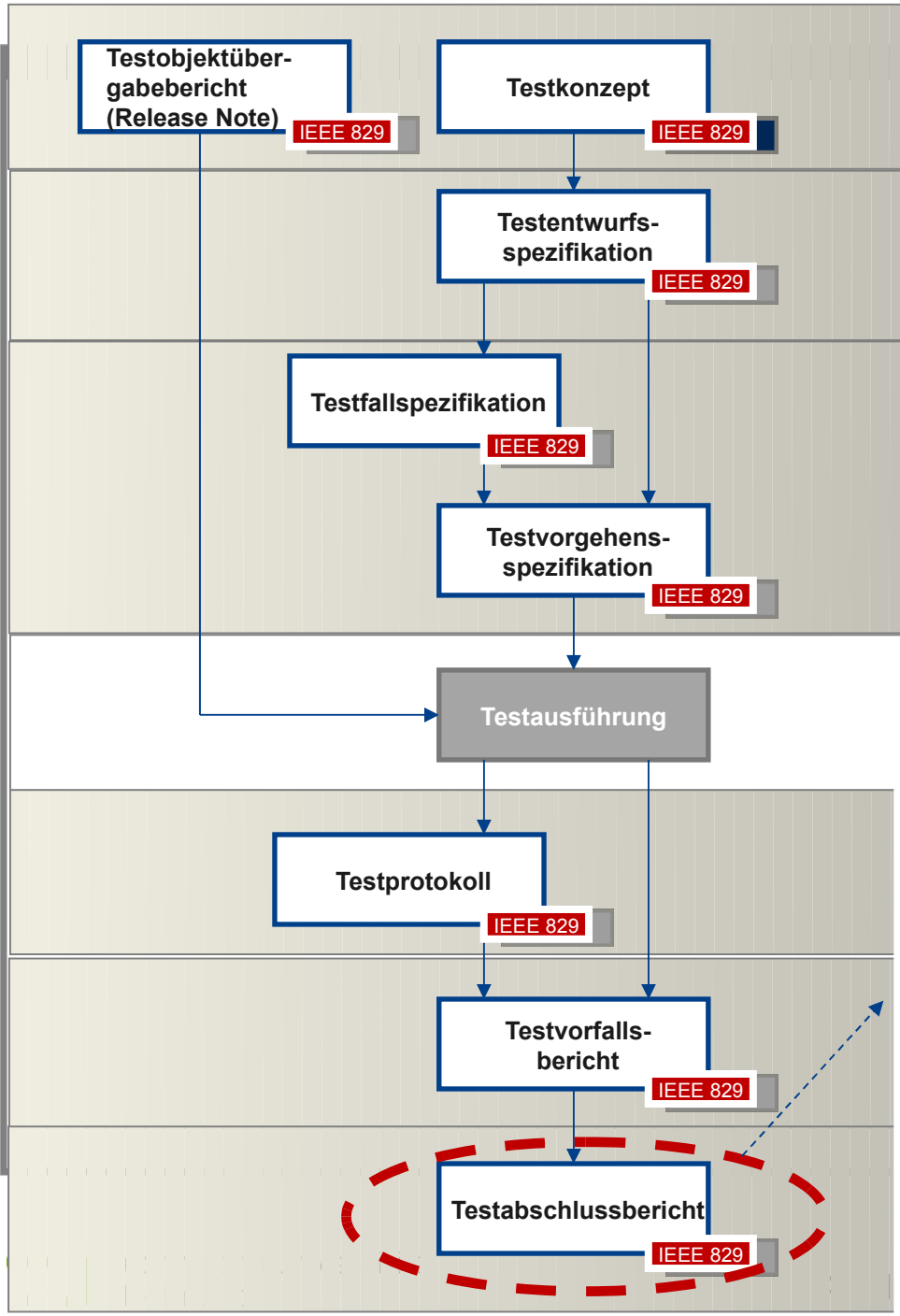
Nach jedem Testzyklus erstellt der Testmanager einen Teststatusbericht, aus dem folgende Informationen über den Stand der Testarbeiten hervorgehen:

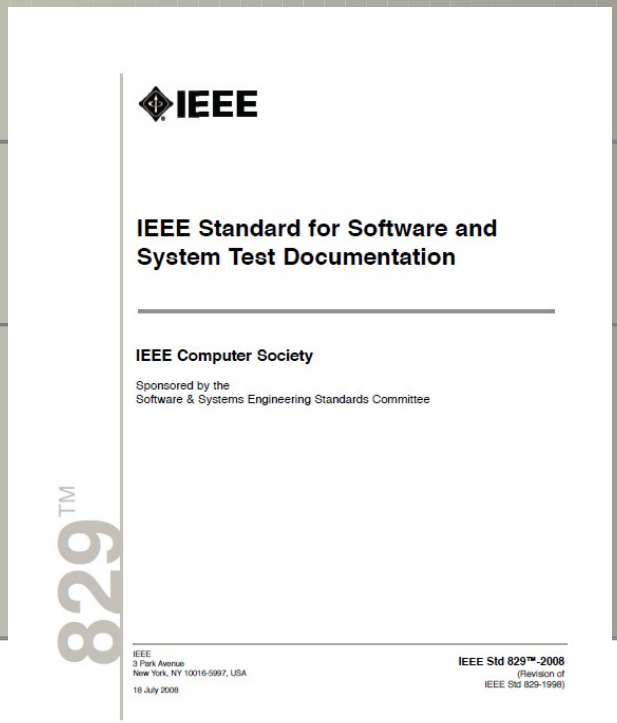
- Testobjekt(e), Teststufe, Testzyklus-Datum (von ... bis …)
- Testfortschritt: Was passierte während des Testzeitraums (geplante / gelaufene / blockierte Tests) ?
- Fehlerstatus: neue / offene / korrigierte Fehler
- Risiken: neue / veränderte / bekannte
- Ausblick: Planung des nächsten Testzyklus
- Gesamtbewertung
  - Beurteilung der verbleibenden Fehler
  - Fortsetzung des Testens ökonomisch sinnvoll
  - Beurteilung der Freigabereife des Testobjekts
  - Grad des Vertrauens in das Testobjekts
  - Entscheidung über weitere Aktivitäten

Zur Beurteilung und Entscheidungsfindung sollen Metriken herangezogen und ausgewertet werden. Folgende Aspekte sind zu berücksichtigen

- Eignung der Testziele für die Teststufe
- Eignung der gewählten Teststrategie
- Effektivität der Tests zur Erreichung der festgelegten Ziele

Teststatusbericht (2)




IEEE

IEEE Standard for Software and System Test Documentation

IEEE Computer Society

Sponsored by the
Software & Systems Engineering Standards Committee

829™

IEEE
3 Park Avenue
New York, NY 10016-5997, USA
18 July 2008

IEEE Std 829™-2008
(Revision of
IEEE Std 829-1998)

Testbericht (engl. test summary report): Ein Dokument, welches die Testaktivitäten und -ergebnisse zusammenfasst. Es enthält eine Bewertung der durchgeführten Tests gegen definierte Ausgangsbedingungen (Testendekriterien) [nach IEEE 829]

Teststeuerung (1)

Bei Verzögerungen gegenüber der Projekt- oder Testplanung geeignete Gegenmaßnahmen einleiten. Beispiele:

- Bei (neu) erkannten Risiken Änderung der Priorisierung der Tests (z.B. verspätete Lieferung von Softwareteilen).
- Anpassung der zeitlichen Planung, wenn sich die Verfügbarkeit der Testumgebung verzögert.
- Korrigierte Fehlerzustände sind durch einen Entwickler nachzutesten, bevor die Software weiter verwendet wird.
- Ggf. zusätzliche Testressourcen (Personal, Arbeitsplätze, Werkzeuge) anzufordern und einzusetzen, um einen Rückstand gegenüber dem Testplan in den verbleibenden Zyklen aufzuholen

Stehen keine zusätzlichen Ressourcen zur Verfügung, muss der Testplan angepasst werden. Mit niedriger Priorität eingestufte Testfälle werden gestrichen

Eine weitere Möglichkeit ist, Testfälle, die in verschiedenen Varianten geplant sind, nur in einer Variante durchzuführen (z.B. werden Tests nur unter einem Betriebssystem ausgeführt, statt wie geplant unter verschiedenen).

Teststeuerung (2)

Durch solche Anpassungen werden zwar einige interessante Tests nicht absolviert, aber die gesparten Ressourcen ermöglichen es, wenigstens die Testfälle hoher Priorität durchzuführen.

Je nachdem wie gravierend die aufgedeckten Mängel und Probleme sind, ist die Testdauer zu verlängern, weil zusätzliche Testzyklen notwendig sind.

- Nach jedem Korrekturzyklus muss die geänderte Software erneut getestet werden.
- Kann zur Verschiebung der Markteinführung bzw. der Auslieferung des Softwareprodukts führen.

Wichtig ist, dass der Testmanager jede solche Planänderung dokumentiert und kommuniziert.

- Änderung am ursprünglichen Testplan bedeutet in aller Regel eine Erhöhung des Freigaberisikos.

Es ist Aufgabe des Testmanagers, dieses Risiko den Projektverantwortlichen zu jedem Zeitpunkt klar und offen darzulegen.

Teststeuerung (Zusammenfassung)

Teststeuerung beschreibt sämtliche Führungs- oder Korrekturmaßnahmen, die auf Grund gesammelter oder berichteter Informationen und Metriken ergriffen werden.

Maßnahmen können jede Testaktivität betreffen und können jede andere Softwarelebenszyklusaktivität oder -aufgabe beeinflussen.

Beispiele von Maßnahmen zur Teststeuerung sind:

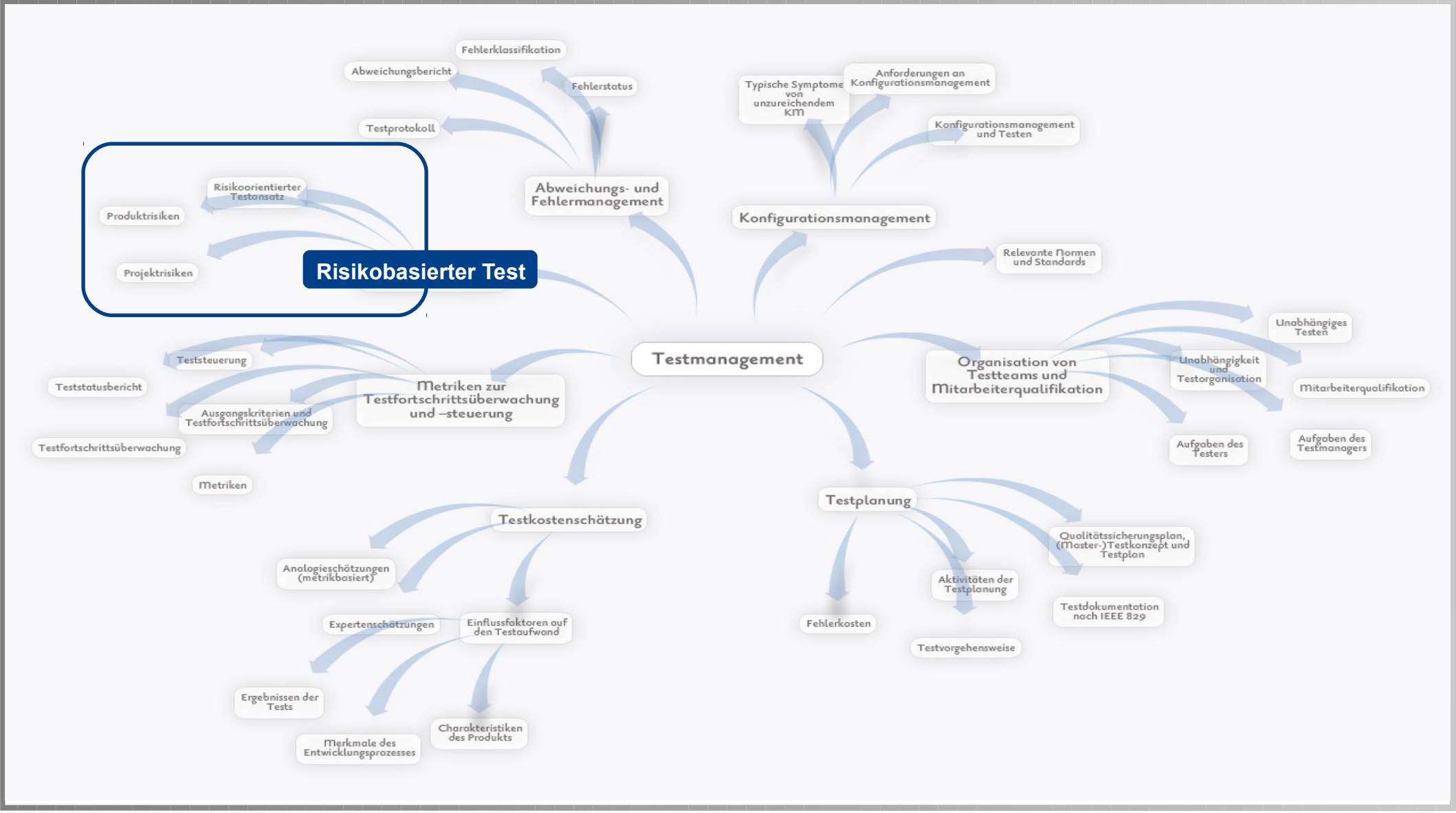
- Repriorisierung von Tests, wenn identifizierte Risiken auftreten.
- Änderung des Testzeitplans aufgrund der Verfügbarkeit der Testumgebung.
- Setzen einer Eingangskriterium mit der Maßgabe, dass Fehlerbehebungen durch einen Entwickler nachzutesten sind, bevor die Software an die nächste Teststufe übergeben wird.

4.6 Testmanagement

4.6 Test-management

- Organisation von Testteams und Mitarbeiterqualifikation
- Testplanung und Testkostenschätzung
- Metriken zur Testfortschrittsüberwachung und -steuerung
- **Risikobasierter Test**
- Abweichungs- und Fehlermanagement
- Anforderungen an das Konfigurationsmanagement
- Relevante Normen und Standards

Risikobasierter Test



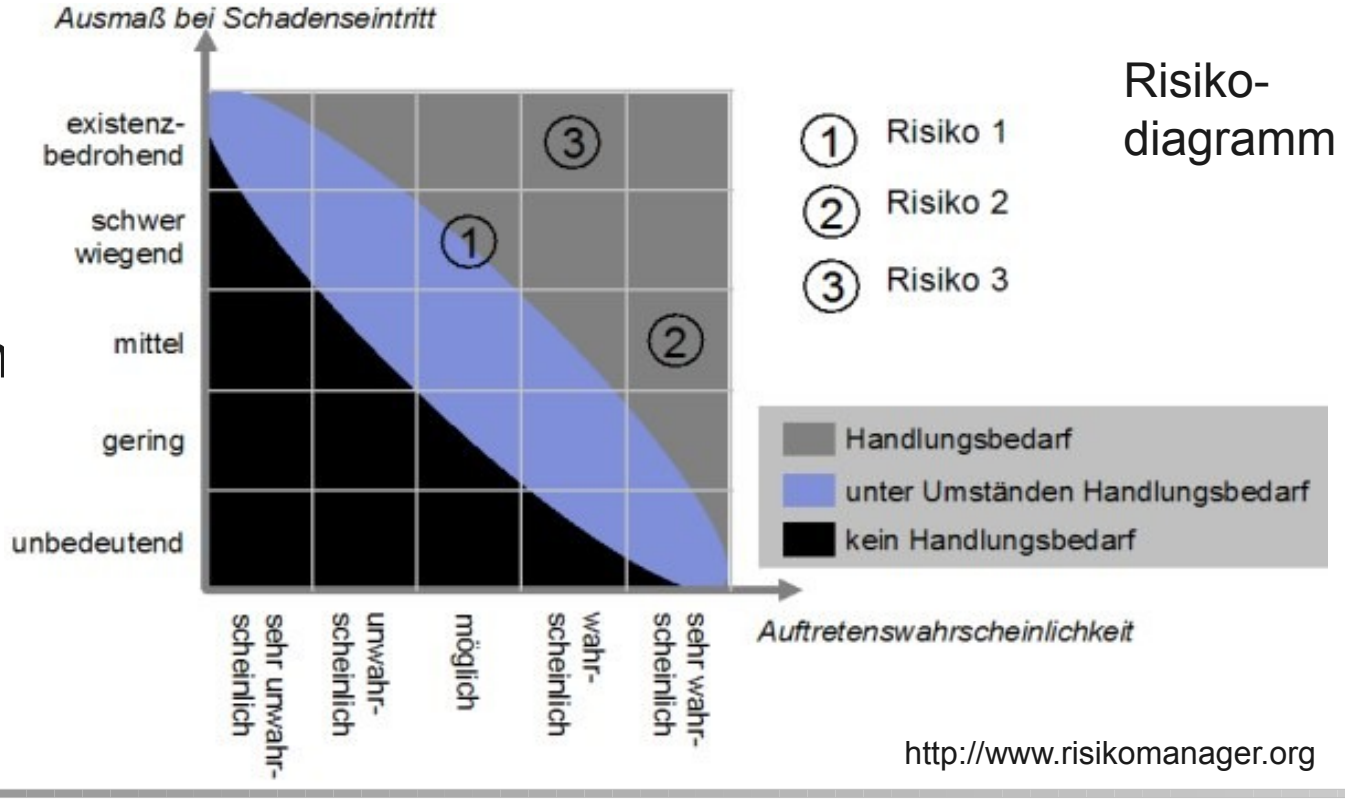
Was ist ein Risiko ?

Risiko: Problem, das in der Zukunft eintreten könnte und unerwünschte Folgen hat; ein Faktor, der zu negativen Konsequenzen in der Zukunft führen könnte. Die Höhe eines Risikos ergibt sich aus der Wahrscheinlichkeit des Problemeintritts (W) und dem damit verbundenen Schaden (S):

$$R = W \cdot S$$

Risiken unterteilen sich in

- Prozessrisiken
- Projektrisiken
- Produktrisiken



Risiko-diagramm

http://www.risikomanager.org

Projektrisiken

Risiken bezüglich der Erreichung der Projektziele:

Lieferantenaspekte: Versagen einer dritten Partei, Vertragsaspekte.

Organisatorische Aspekte: Qualifikation, Mitarbeiterengpässe, Personal- und Trainingsaspekte.

Politische Aspekte:

- Kommunikationsprobleme mit Testern bzgl. ihrer Erfordernisse und Ergebnisse.
- Nicht Nutzung von Erkenntnissen, die beim Testen oder in Reviews gefunden werden (z.B. keine Umsetzung von erkannten Verbesserungen der Entwicklungs- und Testpraktiken).
- Nicht angemessene Haltung gegenüber dem / Erwartungen an das Testen (z.B. Unterschätzung des Nutzens der Fehleraufdeckung durch das Testen).

Technische Aspekte:

- Beschreibung der Anforderungen ist zu ungenau.
- Erfüllung der Anforderungen (unter den gegebenen Randbedingungen) hat zu viel "Spielraum" (zu viele Auslegungsmöglichkeiten).
- Unzureichende Qualität des Designs, des Codes, der Konfigurationsdaten, Testdaten und der Tests.
- Nicht rechtzeitig verfügbare Testumgebung.
- Verspätete Lieferung von Werkzeugen (z.B. Datenkonvertierungswerkzeuge).

Produktrisiken

Sicherheits-Risiken bezüglich des Testobjekts (und seines Einsatzes):

- Risiken gegen Unfallsicherheit (Safety): Produkt könnte Geräten, Menschen oder Organisationen **Schaden** zufügen.
- Risiken gegen Angriffsicherheit (Security): Produkt ermöglicht bzw. unterstützt kriminelle Handlungen (Viren, phishing, …).

Risiken bezüglich der **Qualität** des (gelieferten) Produkts:

- Auslieferung von fehlerhafter Software.
- Unzureichende Eigenschaften der Software (z.B. mangelnde Sicherheit, ungenügende Zuverlässigkeit, schlechte Benutzbarkeit und nicht ausreichende Leistungsfähigkeit/Performanz).
- Nichterfüllung der geforderten / erwarteten Funktionen durch die Software.
- Schlechte Datenintegrität und –qualität (z.B. Datenmigrationsprobleme, Datenkonvertierungsprobleme, Datenüberführungsprobleme, Verletzung von Datenstandards).

Risikoliste

Risikoliste		Projekt: CarConfig					Version: 1.8e			Vom: 15. Juni 2006
	Risikobewertung						Risikostatus			
ID	Bezeichnung / Eigentümer	W	S	RPZ	Indikator	Trig ger	Akt. wert	Maßnahmen	Verant-wortl.	Einge-leitet am
R7	Verlust der Konfig-Management-Datenbank / Sc	9	500	4500	Antwortzeit	5 s	1	Neuer Server	Mü	-
					Recovery	2/T	0,1			
R2	Verzögerung in der Entwicklung / Me	9	100	900	Meilenstein-verzug	14 T	7	Testplan angleichen	Sc	10.06.06
					Krankenstand	3	3	Testpersonal abordern	Wy	15.06.06
R4	Unzureichende Eingangsqualität der Testobjekte / Sc	9	10	90	Fail-Eing.Test	3/T	2	Code-Reviews intensivieren	Me	-
					Fehlerrate	15/T	12			
...	...				...			...		

Legende
W - Wahrscheinlichkeit des Auftretens (1 = vernachlässigbar, ... , 9 = sehr hoch)
S - Schaden (nichtlineare Abbildung auf (Geld-)Werte: z.B. 1, 10, 50, 100, 500, 1000)
RPZ - Risikoprioritätszahl (RPZ = S * W)
Indikator - Messgröße zur Ankündigung eines bevorstehenden Risikoeintritts
Trigger - Schwellwert mit Einleitung der Maßnahmen zur Risikobekämpfung (z.B. Fail-Eing.Test - bei mehr als 3 fehlgeschlagenen Eingangstest pro Tag sind als Maßnahme vermehrt Code-Reviews durchzuführen.)

Tabelle aus: Spillner, Roßner, Winter, Linz: Praxiswissen Softwaretest – Testmanagement. dpunkt verlag, Heidelberg, 2006, S. 191

Risikomanagement

Risikomanagement ist die systematische Anwendung von Praktiken für die Aufgaben der Risikoidentifizierung, Risikoanalyse, Risikopriorisierung und Risikosteuerung zur Verhinderung von Fehlschlägen.

Aktivitäten:

- Ermittlung des Risikokontextes
- Risikoidentifikation
- Risikoanalyse und -priorisierung
- Risikosteuerung und -bewältigung
- Risikoüberprüfung und -überwachung

Risikoorientierter Testansatz

Ziel: Test-Aktivitäten priorisieren auf Basis des verbleibenden Risikos. Pro-aktive Möglichkeit zur Reduzierung des Produktrisikos, beginnend mit Projektstart.

Identifikation von Produktrisiken und ihre Berücksichtigung bei der Steuerung der Testplanung, sowie der Erstellung und Durchführung von Tests. Identifizierte Risiken bestimmen:

- die einzusetzenden Testverfahren und den Umfang der Tests,
- die Priorisierung der Tests mit dem Ziel, kritische Fehlerzustände so früh wie möglich zu finden,
- ob zusätzlich zum Testen weitere Tätigkeiten zur Risikoreduktion notwendig sind (z.B. die Bereitstellung von Schulungsmaßnahmen für unerfahrene Designer).

Risikoorientiertes Testen bietet Auskunft über verbleibende Risiken, durch den Nachweis der Effektivität der Behebung von kritischen Fehlern.

Beim risikoorientierten Testen werden das Wissen und die Kenntnisse der Stakeholder genutzt, um Risiken zu identifizieren und die erforderlichen Tests zum Behandeln der Risken zu bestimmen.

Risikomanagement hilft die **Wahrscheinlichkeit von Produktfehlern zu minimieren** und bietet einen systematischen Ansatz für:

- die Beurteilung (und regelmäßiger Neubewertung) von Fehlerwirkungen, die auftreten können (Risiken),
- die Entscheidung, welche Risiken behandelt werden müssen,
- die Umsetzung von Maßnahmen zur Behandlung bzw. Behebung dieser Risiken.

Zusätzlich kann Testen:

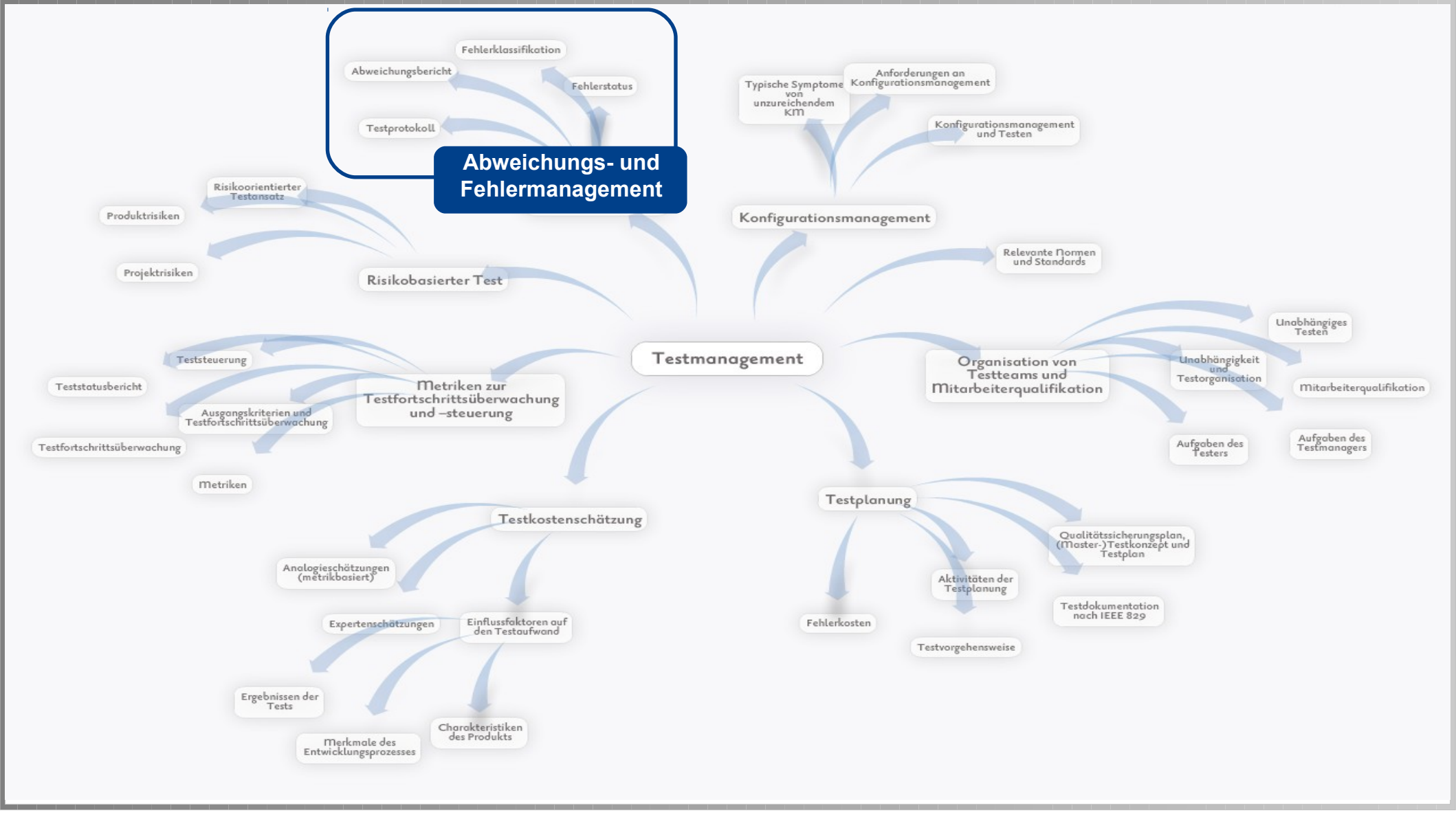
- die Identifikation neuer Risiken unterstützen,
- helfen festzulegen, welche Risiken reduziert werden sollen,
- die Unsicherheit bezüglich der Risiken verringern.

4.6 Testmanagement

4.6 Test-management

- Organisation von Testteams und Mitarbeiterqualifikation
- Testplanung und Testkostenschätzung
- Metriken zur Testfortschrittsüberwachung und -steuerung
- Risikobasierter Test
- **Abweichungs- und Fehlermanagement**
- Anforderungen an das Konfigurationsmanagement
- Relevante Normen und Standards

Abweichungs- und Fehlermanagement



Fehler- und Abweichungsmanagement

Damit die in den verschiedenen Teststufen gefundenen Mängel, Probleme und Fehlerwirkungen zuverlässig und schnell korrigiert werden können, bedarf es eines gut funktionierenden Verfahrens zur Übermittlung und Verwaltung entsprechender Fehlermeldungen.

Ein solches Verfahren wird als **Fehlermanagement oder Abweichungsmanagement** bezeichnet.

Dieses Verfahren beginnt nach Ende eines Testlaufs und umfasst:

- Testprotokoll
- Fehlermeldung (allgemein: Abweichungsbericht)
- Fehlerklassifikation
- Fehlerstatus

Testprotokoll

Auswertung der angefallenen Testprotokolle:

- Abweichungen zwischen erwartetem Soll-Verhalten und beobachtetem Ist-Verhalten.
- Die Tester vergewissern sich, ob tatsächlich eine Abweichung zum Sollverhalten vorliegt oder ob die Ursache ein falsch entworfener Testfall, eine fehlerhafte Testautomatisierung oder eine falsche Testausführung war (auch einem Tester kann eine Fehlhandlung unterlaufen).

Fehler dokumentieren:

- Liegt das Problem im Testobjekt, wird über die Beobachtung eine Problemmeldung oder Fehlermeldung erstellt.
- Für jede im Testprotokoll enthaltene und vom Sollwert bzw. Sollverhalten abweichende Beobachtung (Duplikate von Fehlermeldungen vermeiden).

Ursachenanalyse ist Aufgabe der Entwickler:

- Die Tester müssen nicht die Ursache des gemeldeten Problems herausfinden.

Fehlermeldung - Fehlerdatenbank

Üblicherweise ist eine zentrale **Fehlerdatenbank** vorhanden, in der alle Probleme, Mängel oder Fehlerwirkungen, die im Test (oder auch später im Betrieb) entdeckt werden, erfasst und verwaltet werden.

Fehlermeldungen können von allen an der Entwicklung beteiligten Personen, aber auch von Kunden- und Anwenderseite eingebracht werden.

Sie können sich auf Probleme in den getesteten Programm-(teil)en beziehen, aber auch auf Fehler oder Mängel in Spezifikationen, Benutzerhandbüchern oder anderen Dokumenten.

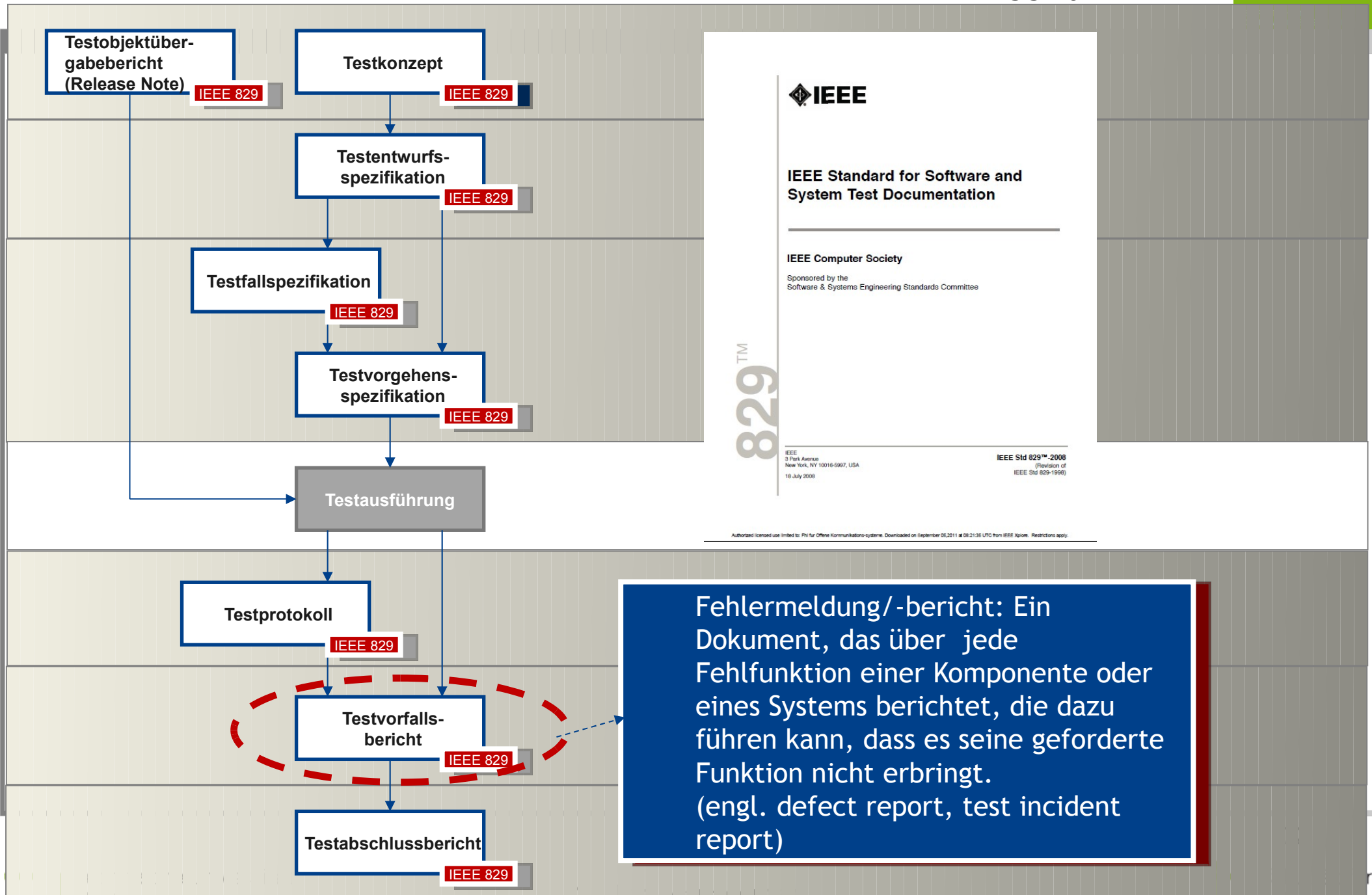
Fehlermeldeverfahren - oft auch **Problemmeldeverfahren** genannt.

- Denn gemeldet werden alle offenen Probleme; aber nicht jedes Problem muss ein Fehler (der Entwickler) sein. Problemmeldung klingt auch weniger nach »Schuldzuweisung«.
- Es ist keine Einbahnstraße, sondern jeder Entwickler kann betreffende Meldungen kommentieren, etwa um Rückfragen an den Tester zu richten oder eine unberechtigte Meldung zurückzuweisen.

Nimmt ein Entwickler aufgrund einer Fehlermeldung eine Korrektur am Testobjekt vor, so wird er dies ebenfalls in der Fehlerdatenbank dokumentieren, damit der zuständige Tester diese Korrektur im nächsten Testzyklus nachvollziehen und erneut testen kann.

Testmanager und Projektmanager können sich mittels der Fehlerdatenbank zu jedem Zeitpunkt ein aktuelles und vollständiges Bild über Anzahl und Status der Fehler und über den Korrekturfortschritt verschaffen.

Fehlermeldung/ -bericht



IEEE

IEEE Standard for Software and System Test Documentation

IEEE Computer Society

Sponsored by the
Software & Systems Engineering Standards Committee

829™

IEEE
3 Park Avenue
New York, NY 10016-5997, USA
18 July 2008

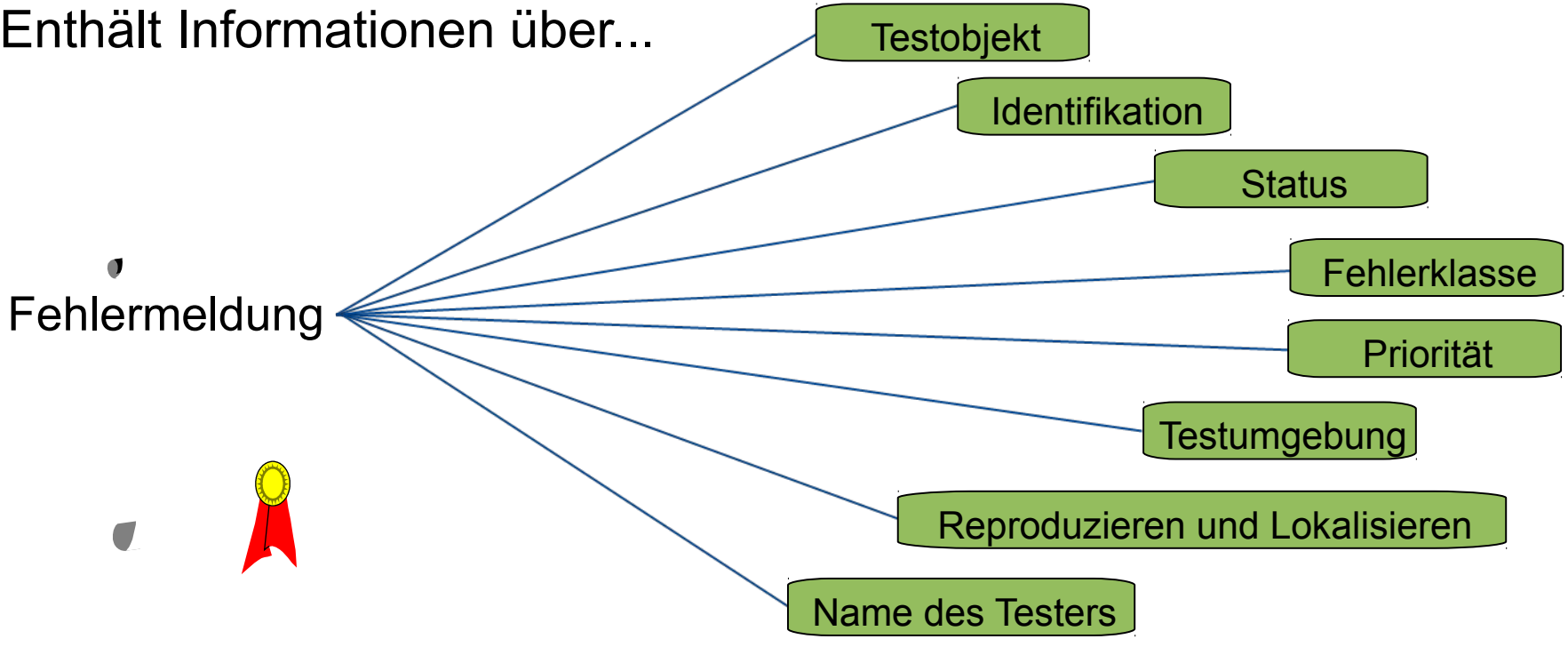
IEEE Std 829™-2008
(Revision of
IEEE Std 829-1998)

Fehlermeldung/-bericht: Ein Dokument, das über jede Fehlfunktion einer Komponente oder eines Systems berichtet, die dazu führen kann, dass es seine geforderte Funktion nicht erbringt.
(engl. defect report, test incident report)

Fehlermeldung - einheitliches Schema (1)

Typischerweise beinhaltet eine Fehlermeldung neben der eigentlichen Problembeschreibung Informationen zur Identifikation der getesteten Software, zur Testumgebung, den Namen des Testers, die Fehlerklasse, Status und Priorisierung sowie andere Informationen mit Relevanz für das Reproduzieren und Lokalisieren des potenziellen Fehlers.

Enthält Informationen über...



Fehlermeldung - einheitliches Schema (2)

Damit die Kommunikation reibungslos funktioniert und die Meldungen statistisch ausgewertet werden können, muss jede Meldung nach einem projektweit vorgegebenen einheitlichen Schema aufgebaut sein. Dieses Schema legt der Testmanager (z.B. im Testkonzept) fest. Typischerweise beinhaltet eine **Fehlermeldung nach IEEE 829:**

- Datum der Entdeckung der Abweichung / des Problems sowie
- Identifikation (oder Bezeichnung der Konfiguration) der Software / des Dokuments
- Aktivität, in der die Abweichung beobachtet wurde
- ggf. Nummer des Testfalls, Ist- und vorausgesagte Ergebnisse (Soll-Ergebnisse) sowie genaue Beschreibung der Abweichung (zur besseren Ermöglichung der Behebung)
- Schwere der Auswirkung auf das System und auf Stakeholder-Interessen (Fehlerklasse)
- Dringlichkeit / Priorität der Behebung (Fehlerpriorität)
- Status (z.B. offen, zurückgestellt, abgewiesen, Duplikat, Korrektur, Nachtest oder erledigt)
- Schlussfolgerungen und Empfehlungen (aus dem gefundenen Fehler, z.B. Vermeidung von Mehrfachvererbung)
- Globale Probleme, z.B. wenn andere Bereiche durch die Korrektur der Abweichung beeinflusst werden könnten
- Änderungshistorie, welche Tätigkeiten von welchen Projektteammitgliedern zur Korrektur der Abweichung durchgeführt wurde (Isolation der Fehlerursache, Änderung der betreffenden Code-Stellen, Fehlernachtest)

Fehlermeldung

Wichtig ist, dass alle Informationen erfasst werden, die für das **Reproduzieren und Lokalisieren des potenziellen Fehlers** notwendig sind, sowie Informationen, die eine Auswertung der Produktqualität und des Korrekturfortschritts ermöglichen.

Unabhängig vom vereinbarten Schema gilt:

- Eine Meldung muss objektiv verfasst werden.
- Eine Meldung muss so abgefasst sein, dass der zuständige Entwickler mit minimalem Aufwand das Problem nachvollziehen und dessen Ursache finden kann.
- Die Reproduktion, Lokalisierung und Behebung eines Problems bedeuten für den Entwickler in der Regel nicht eingeplante, zusätzliche Arbeit; in dieser Situation neigen Entwickler dazu, jede unklare Fehlermeldung ohne weitere Analyse abzulehnen oder zurückzustellen.

Fehlerklasse – Auswirkung des Fehlers im Betrieb

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

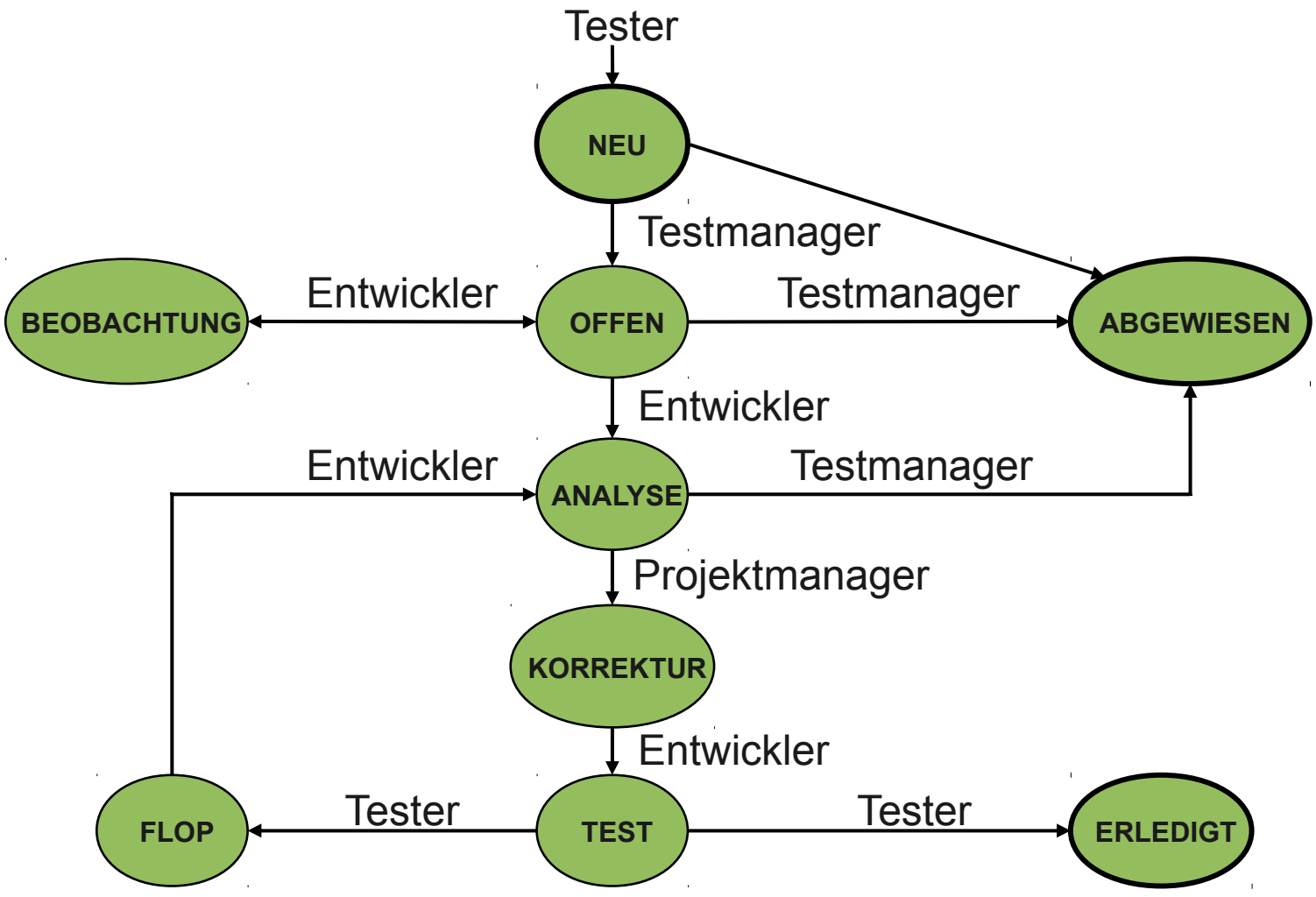
Fehlerpriorität – Dringlichkeit für die Behebung

Priorität	Bedeutung
1 – PATCH	Der Arbeitsablauf beim Anwender ist blockiert oder die laufenden Tests können nicht fortgesetzt werden. 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	Korrekturplanung ist noch zu treffen

Fehlerstatus

- Das Testmanagement muss nicht nur sicherstellen, dass Fehler ordentlich erfasst und verwaltet werden, es muss in Kooperation mit dem Projektmanagement dafür sorgen, dass Fehlerzustände zügig korrigiert werden und dass diese Korrekturen mit den neuen Versionen des Testobjekts verfügbar sind.
- Hierzu ist eine kontinuierliche Verfolgung des **Fehleranalyse- und Korrekturprozesses** über alle Stadien notwendig.
- Diese Verfolgung geschieht anhand des Fehlerstatus.
- Dabei durchläuft jede Fehlermeldung eine Reihe festgelegter Stati, welche die erstmalige Erfassung bis zur erfolgreichen Fehlerkorrektur beinhalten.

Fehlerstatusmodell



Fehlerkorrektur - Erweiterungen

Die von den Entwicklern auszuführenden Änderungen sind in vielen Fällen keine »Fehlerkorrekturen«, sondern echte (funktionale) Erweiterungen.

Da die Grenze zwischen »Fehlermeldung« und »Erweiterungswunsch« und die Beurteilung als »berechtigt« oder »unberechtigt« oft Ermessenssache sind, wird im Fehlermanagement eine Instanz benötigt, die **Änderungsanforderungen genehmigt oder ablehnt**.

Diese Instanz heißt Änderungskontrollausschuss (*change control board*) und wird üblicherweise von Vertretern folgender Interessengruppen gebildet:

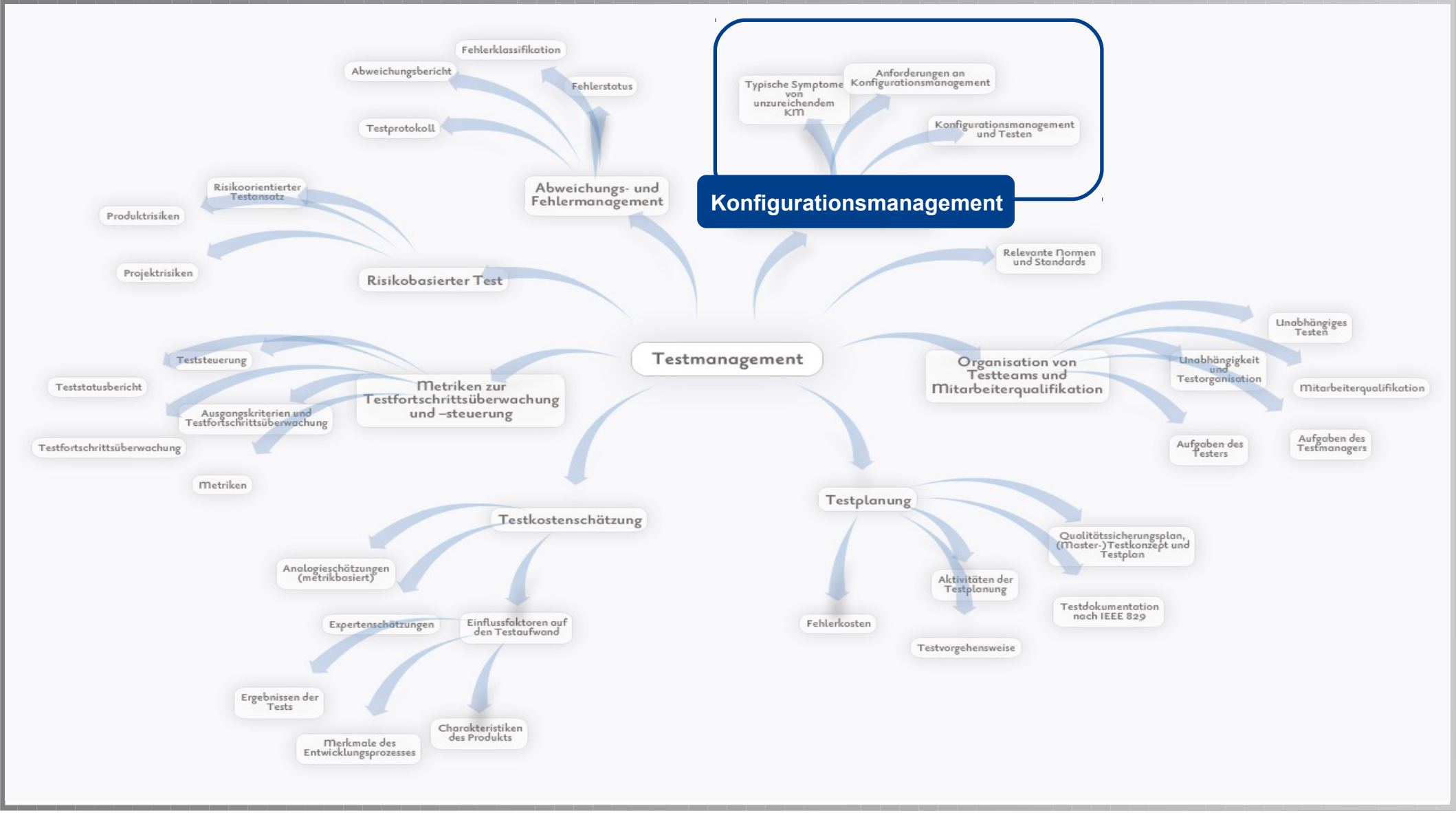
- Produktmanagement
- Projektmanagement
- Testmanagement
- Kundenvertreter

4.6 Testmanagement

4.6 Test-management

- Organisation von Testteams und Mitarbeiterqualifikation
- Testplanung und Testkostenschätzung
- Metriken zur Testfortschrittsüberwachung und -steuerung
- Risikobasierter Test
- Abweichungs- und Fehlermanagement
- **Anforderungen an das Konfigurationsmanagement**
- Relevante Normen und Standards

Anforderungen an das Konfigurationsmanagement



Ziel: Schaffung und Erhalt der Integrität (Aktualität, Vollständigkeit) der Software oder des Systems (Komponenten, Daten und Dokumentation, ...) während des Projekt- und Produktlebenszyklus.

Ein Softwaresystem besteht aus einer Vielzahl von Einzelbausteinen, die zueinander passen müssen, damit das System als Ganzes funktioniert.

Von jedem dieser Bausteine entstehen im Laufe der Entwicklung des Systems neue, korrigierte oder verbesserte Versionen oder Varianten.

Da an diesem Prozess mehrere Entwickler und Tester parallel beteiligt sind, ist es alles andere als einfach, den Überblick zu behalten, welche Bausteine aktuell sind und zusammengehören, dies ist Aufgabe des **Konfigurationsmanagement.**

Typische Symptome von unzureichendem KM

Entwickler **überschreiben** gegenseitig ihre am Code oder anderen Dokumenten vorgenommenen **Modifikationen**, da der gleichzeitige Zugriff auf gemeinsame Codedateien nicht verhindert wird. Die Integrationsarbeiten sind behindert,

- weil unklar ist, welche Versionen des Codes einer bestimmten Komponente im Entwicklungsteam existieren und welche davon aktuell sind,
- weil unklar ist, welche Versionen verschiedener Komponenten zusammengehören und zu einem größeren Teilsystem integriert werden können und
- weil Compiler und andere Entwicklungswerkzeuge in unterschiedlichen Versionen eingesetzt werden.

Fehleranalyse, Fehlerkorrektur und Regressionstest sind erschwert,

- weil unbekannt ist, wo und warum der Code einer Komponente gegenüber der Vorversion geändert wurde und
- weil unbekannt ist, aus welchen Codedateien ein bestimmtes integriertes Teilsystem (Objektcode) hervorgegangen ist.

Tests und Testauswertung sind behindert,

- weil unklar ist, welche Testfälle zu welchem Versionsstand eines Testobjekts gehören und
- weil unklar ist, welche Testergebnisse ein Testlauf an einem Testobjekt einer bestimmten Version erbracht hat.

Anforderungen an Konfigurationsmanagement

Versionsverwaltung:

- Katalogisieren, Speichern und Wiederabrufen von unterschiedlichen Versionen eines Konfigurationsobjekts (z.B. Version 1.0 und 1.1 einer Datei).
- Hierzu gehört auch das Mitführen von Kommentaren, aus denen der jeweilige Änderungsgrund hervorgeht.

Konfigurationsverwaltung:

- Bestimmung und Verwaltung aller Dateien (Konfigurationsobjekte) in der jeweils passenden Version, die zusammen ein Teilsystem bilden (Konfiguration).
- Voraussetzung hierfür ist eine Versionsverwaltung.

Statusverfolgung von Fehlern und Änderungen:

- Aufzeichnung von Problemberichten und Änderungsanforderungen und die Möglichkeit, deren Umsetzung an den Konfigurations-objekten nachzuvollziehen.

Konfigurationsaudits:

- Um die Wirksamkeit des Konfigurationsmanagements zu prüfen, ist es sinnvoll, Konfigurationsaudits durchzuführen.
- In einem solchen Audit kann z.B. geprüft werden, ob alle Softwarebausteine vom Konfigurationsmanagement erfasst werden, ob die Konfigurationen korrekt identifiziert werden können usw..

Konfigurationsmanagement kann für das **Testen** sicherstellen, dass:

- alle Elemente der Testmittel sich genau bestimmen lassen, einer Versionskontrolle unterworfen sind, Änderungen verfolgt werden und zueinander und zu den Entwicklungseinheiten (Testobjekten) in Beziehung gesetzt werden können,
- die Rückverfolgbarkeit während des Testprozesses oder des Produktlebenszykluses gewährleistet ist und
- alle Dokumente und Software(teile) eine eindeutige Zuordnung in der Testdokumentation haben.

Konfigurationsmanagement unterstützt den Tester die Testobjekte, Testdokumente, Testfälle und Testrahmen genau zu bestimmen (und ggf. zu reproduzieren).

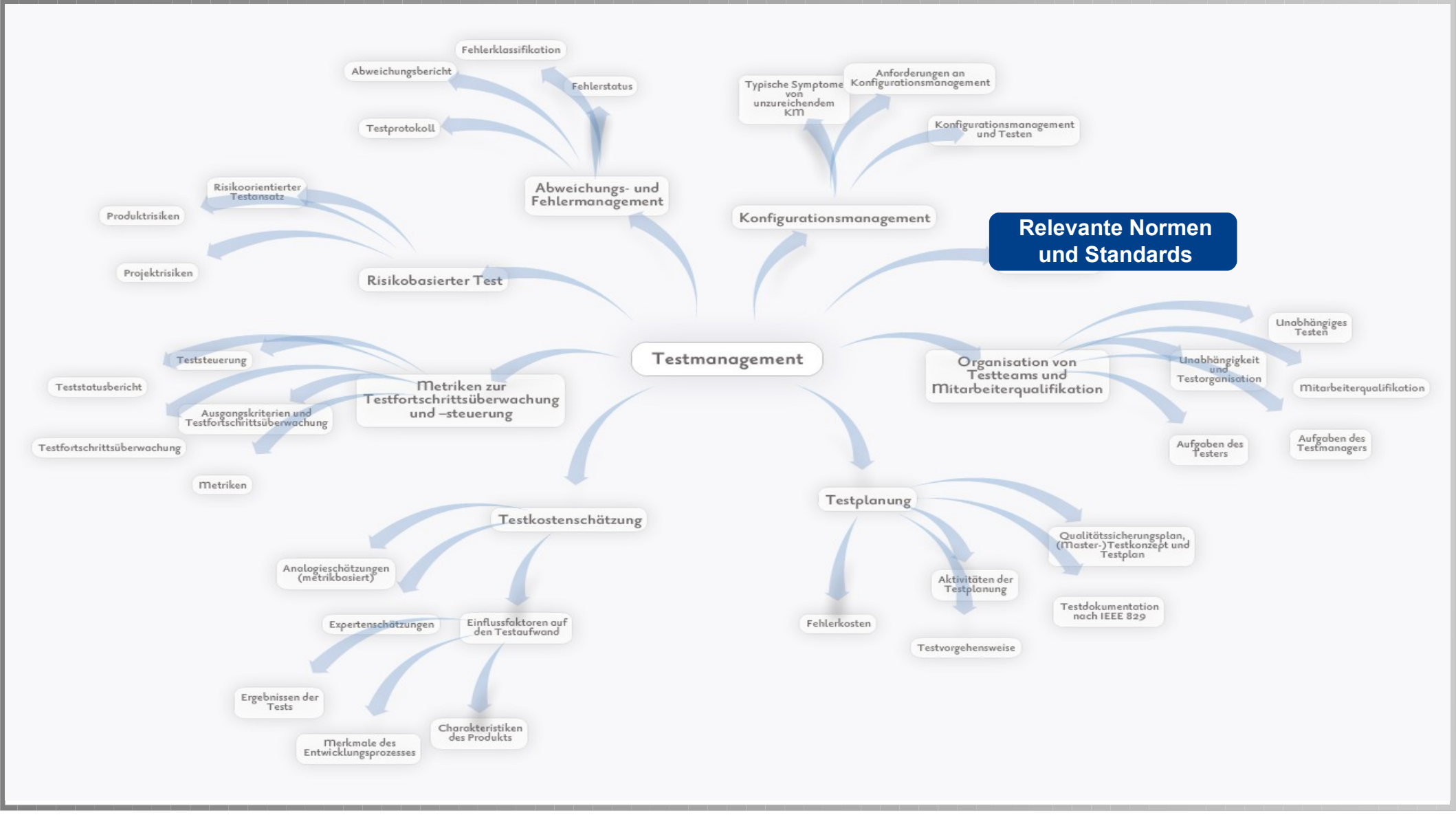
Während der Testplanung sollen die Vorgehensweisen (Abläufe) des Konfigurationsmanagements und die zu verwendende Infrastruktur (Konfigurationsmanagementwerkzeuge) ausgewählt, beschrieben und realisiert werden.

4.6 Testmanagement

4.6 Test-management

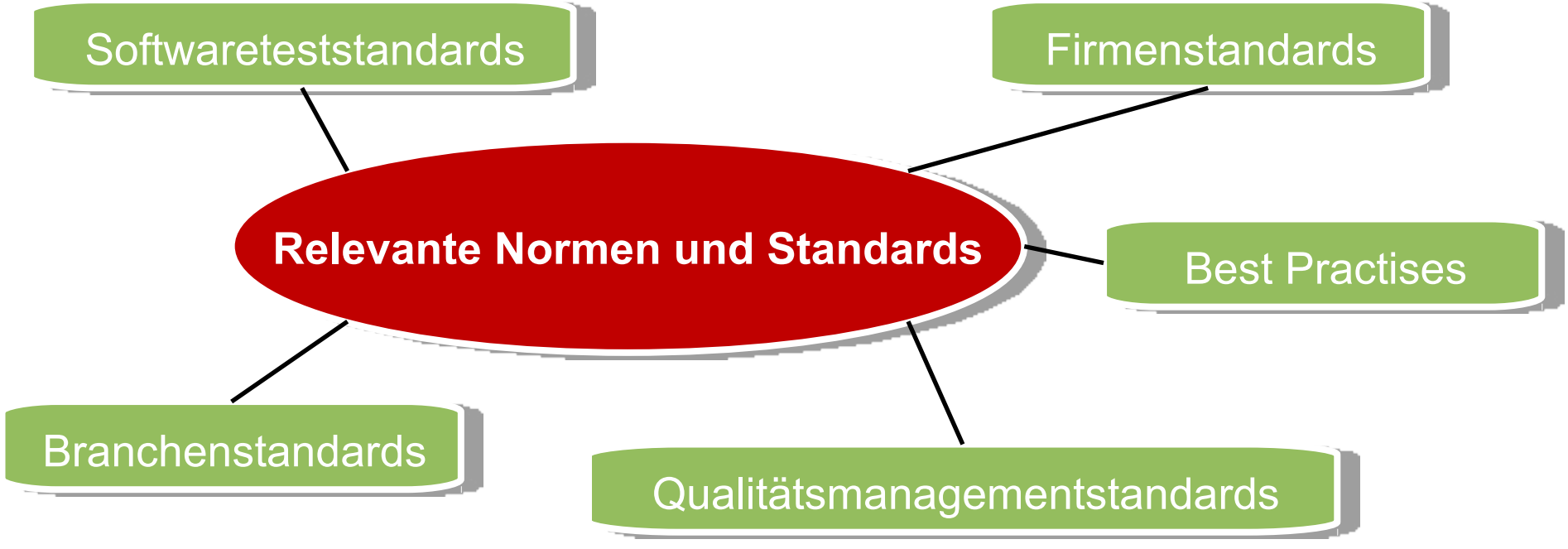
- Organisation von Testteams und Mitarbeiterqualifikation
- Testplanung und Testkostenschätzung
- Metriken zur Testfortschrittsüberwachung und –steuerung
- Risikobasierter Test
- Abweichungs- und Fehlermanagement
- Anforderungen an das Konfigurationsmanagement
- **Relevante Normen und Standards**

Relevante Normen und Standards



Relevante Normen und Standards (1)

Zu den Aufgaben eines Qualitäts- oder Testmanagers gehört es festzustellen, welche Normen, Standards oder evtl. gesetzliche Richtlinien für das zu testende Produkt (Produktnormen) oder auch für das Projekt (Prozessnormen) relevant sind, und deren Einhaltung sicherzustellen.
Normen und Standards, die Randbedingungen vorgeben und den Stand der Technik definieren:



Firmenstandards: Firmeninterne Richtlinien und Verfahrensanweisungen (des Herstellers und ggf. des Kunden), wie z.B. Qualitätsmanagement-handbuch, Rahmentestplan, Programmierrichtlinien.

Best Practises: Nicht standardisierte, aber fachlich bewährte Vorgehensweisen und Verfahren, die den Stand der Technik in einem Anwendungsgebiet darstellen.

Qualitätsmanagementstandards:

- Branchenübergreifende Standards, die Mindestanforderungen an Prozesse spezifizieren, ohne konkrete Anforderungen bezüglich der Umsetzung zu formulieren.
- Bekanntes Beispiel ist [ISO 9000], die z.B. fordert, dass im Produktionsprozess (also auch im Spezialfall Software-entwicklungsprozess) geeignete (Zwischen-)Prüfungen vorzusehen sind, ohne anzugeben, wann und wie diese zu erledigen sind.

[ISO 9000] ISO 9000:2000. Qualitätsmanagementsysteme – Grundlagen und Begriffe.

Exkurs: Relevante Normen und Standards (3)

Branchenstandards: Branchenspezifische Standards, die für eine bestimmte Produktkategorie oder ein Einsatzgebiet festlegen, in welchem Mindestumfang Tests durchzuführen oder nachzuweisen sind. Beispielsweise:

- [DIN 60601-1-4] für Medizinprodukte,
- [RTC-DO 178B] für Software in Flugzeugen,
- [EN 50128] für Bahn-Signalsysteme,

Softwareteststandards: Prozessstandards, die produktunabhängig festlegen, wie Softwaretests fachgerecht durchzuführen sind. Beispiele: [ISO 9126] [BS 7925-2], [IEEE 829], [IEEE 1028].

Technologiestandards für das Software Testen: Standards, die Technologien für die Automatisierung des Software Testens definieren. Beispiele: [ETSI TTCN-3] [OMG UTP], [IEEE 1671]

[DIN 60601-1-4] DIN EN 60601-1-4, Ausgabe: 2001-04. Medizinische elektrische Geräte – Teil 1-4: Allgemeine Festlegungen für die Sicherheit; Ergänzungsnorm: Programmierbare elektrische medizinische Systeme
[RTC-DO 178B] RTC-DO Std 178B, Software Considerations in Airborne Systems and Equipment: Certification, RTCA, Inc., 1992
[EN 50128] EN 50128:2001. Railway applications – Communication, signalling and processing systems – Software for railway control and protection systems, European Committee for Electrotechnical Standardization

[ISO 9126] ISO/IEC 9126 Teile 1-4: Software Engineering - Product Quality, 2001
Part 1: Quality model; Part 2: External Metrics; Part 3: Internal Metrics; Part 4: Quality in Use Metrics
[BS 7925-2] British Standard 7925-2: Software testing, Part 2: Software component testing, 1998
[IEEE 829] IEEE Std 829-1998: IEEE Standard for Software Test Documentation
[IEEE 1028] IEEE Std 1028-1997: IEEE Standard for Software Reviews

[ETSI TTCN-3] ETSI Testing and Test Control Notation, v4.3.1, Juli 2011. ebenso als ITU-T Z.140 ff.
[OMG UTP] OMG UML Testing Profile, v.1.1, Juli 2011
[IEEE 1671] Standard for Automatic Test Markup Language (ATML) for Exchanging Automatic Test Information via XML, 2010

Zusammenfassung

Entwicklungs- und Testaktivitäten sollen organisatorisch getrennt sein. Je klarer diese Trennung erfolgt, umso wirksamer kann getestet werden !

Je nach Aufgabenstellung innerhalb des Testprozesses werden Mitarbeiter mit rollenspezifischen Testkenntnissen benötigt (neben fachlichen auch soziale Kompetenz).

Zu den Aufgaben des Testmanagers gehören Planung, Überwachung und Steuerung der einzelnen Testzyklen. In dem mit dem übergeordneten Qualitätssicherungsplan abgestimmten Testkonzept beschreibt und begründet der Testmanager, wie er die anstehenden Testaufgaben lösen will, welchen Testumfang er plant, welche Werkzeuge eingesetzt werden sollen usw.

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.

Risikomanagement und risikoorientierter Test helfen, die kritischen Fehler frühzeitig zu finden und den Testaufwand zu optimieren.

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. Eine Orientierung an solchen Standards macht auch dort Sinn, wo deren Einhaltung nicht verpflichtend vorgeschrieben ist.

Nach diesem Abschnitt sollten Sie...

- Wissen über die Organisation von Testteams haben und die wichtigsten Aufgaben von Testmanager und Tester kennen,
- Ziel und Inhalt eines Testkonzepts und einer Testspezifikation nach IEEE 829 zusammenfassen können,
- die Testplanung durchführen und dabei zwei Verfahren zur Schätzung der Testkosten kennen und Kosten- und Wirtschaftlichkeitsaspekte berücksichtigen können,
- Metriken zur Testfortschrittsüberwachung und -steuerung kennen,
- Zweck und Inhalt eines Testabschlussberichts auf der Basis der IEEE 829 zusammenfassen können,
- den Zusammenhang zwischen Risiko und Testen kennen sowie das Risikomanagement für die Testplanung einsetzen können,
- ein Fehlermanagement etablieren und Anforderungen an ein Konfigurationsmanagement benennen können,
- die Anforderungen an ein Konfigurationsmanagement benennen können,
- den Inhalt eines Abweichungsberichts kennen und einen Abweichungsbericht erstellen können.

Folgende Fragen sollten Sie jetzt beantworten können

- Welche grundsätzlichen Modelle einer Aufgabenteilung zwischen Entwicklung und Test lassen sich unterscheiden ?
- Welches sind die typischen Aufgaben der Rollen Testmanager und Tester ?
- Nennen Sie zwei Verfahren zur Schätzung des Testaufwands.
- Welche Arten von Metriken zur Überwachung des Testfortschritts lassen sich unterscheiden ?
- Welche Informationen soll ein Teststatusbericht enthalten ?
- Welches sind die Aktivitäten des Risikomanagements ?
- Wie berechnet man ein Risiko ?
- Was sind typische Produkt- und Projektrisiken ?
- Welche Daten soll eine Fehlermeldung enthalten ?
- Was ist der Unterschied zwischen Fehlerpriorität und Fehlerklasse ?
- Wozu wird ein Fehlerstatusmodell benötigt ?
- Welche Aufgabe hat ein Änderungskontrollausschuss ?
- Welche Anforderungen aus Sicht des Tests stellen sich an das Konfigurationsmanagement ?
- Welche grundsätzlichen Arten von Normen und Standards lassen sich unterscheiden ?

Diese Begriffe sollten Sie kennen ...