



GTB

German Testing Board

Software. Testing. Excellence.



Basiswissen Softwaretest Certified Tester Statischer Test

HS@GTB
2019
Version 3.1

Lernziele für den Abschnitt Statischer Test

(nach Certified Tester Foundation Level Syllabus,
deutschsprachige Ausgabe, Version 2018)



- **3.1 Grundlagen des statischen Tests**

- FL-3.1.1 (K1) Arten von Softwarearbeitsergebnissen erkennen können, die durch die verschiedenen statischen Testverfahren geprüft werden können
- FL-3.1.2 (K2) Beispiele nennen können, um den Wert des statischen Tests zu beschreiben
- FL-3.1.3 (K2) Den Unterschied zwischen statischen und dynamischen Verfahren unter Berücksichtigung der Ziele, der zu identifizierenden Fehlerzustände und der Rolle dieser Verfahren innerhalb des Softwarelebenszyklus erklären können

Lernziele für den Abschnitt Statischer Test

(nach Certified Tester Foundation Level Syllabus,
deutschsprachige Ausgabe, Version 2018)



• 3.2 Reviewprozess

- FL-3.2.1 (K2) Die Aktivitäten des Reviewprozesses für Arbeitsergebnisse zusammenfassen können
- FL-3.2.2 (K1) Die unterschiedlichen Rollen und Verantwortlichkeiten in einem formalen Review erkennen können
- FL-3.2.3 (K2) Die Unterschiede zwischen den unterschiedlichen Reviewarten erklären können: informelles Review, Walkthrough, technisches Review und Inspektion
- FL-3.2.4 (K3) Ein Reviewverfahren auf ein Arbeitsergebnis anwenden können, um Fehlerzustände zu finden
- FL-3.2.5 (K2) Die Faktoren erklären können, die zu einem erfolgreichen Review beitragen

Kapitel 3

Statischer Test



Grundlagen

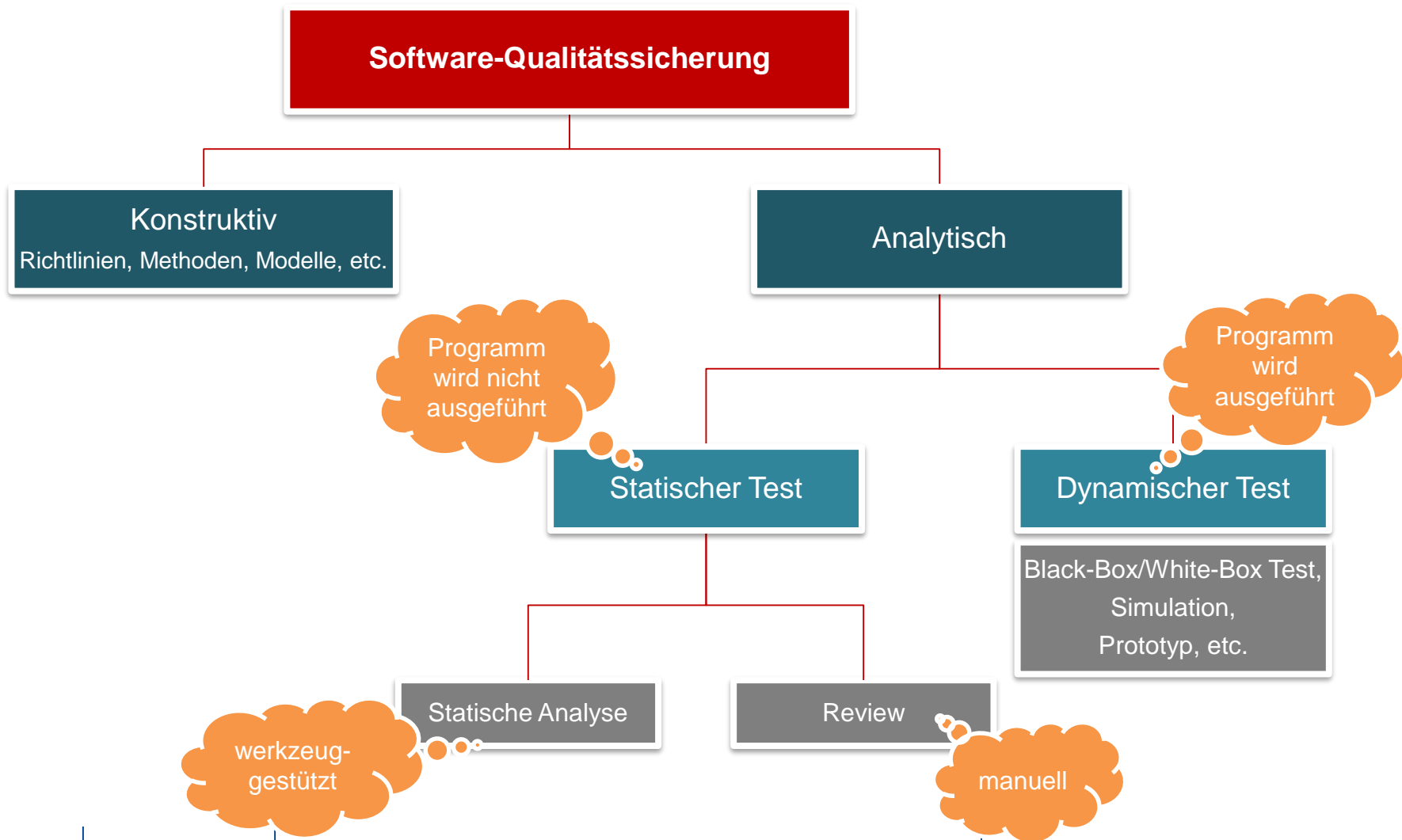
Reviews

Exkurs: Statische Analysen

Exkurs: Kontroll- und Datenflussanalyse

Exkurs: Metriken

Software-Qualitätssicherung und der statische Test



Statischer Test vs. dynamischer Test

- Verschiedene Artefakte im Entwicklungsprozess erstellt
 - Anforderungsspezifikation (Fließtext)
 - Modelle (UML)
 - Programm Quellcode
 - ...
- Programme sind statische Beschreibungen von dynamischen Abläufen (Berechnungen etc.)
- Dynamische Tests: Ausführung des Quellcodes prüfen
 - finden Fehlerwirkungen (Symptome)
- Statische Tests: Prüfung ohne Ausführung
 - finden Fehlerzustände (Ursachen)
- Fokus auf unterschiedliche Fehlerzustände, z.B. ungenutzte Variable vs. falsche Ausgabe



Statischer Test

- Analyse des Testobjekts durch intensive Betrachtung
- Kurzfristiges Ziel: **Finden von Fehlerzuständen**
 - Verstöße gegen Spezifikationen oder Standards, Fehler in Anforderungen oder Design, unzureichende Wartbarkeit
 - Verletzung von Projektplänen
- Mittelfristiges Ziel: Entwicklungsprozess optimieren
- Grundidee: **Prävention!**
 - Fehlerzustände so früh wie möglich beheben
 - Qualität erhöhen, Zeit und Kosten sparen
 - statischen Test vor dynamischem Test durchführen

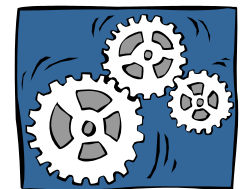


Typische Testobjekte für den statischen Test

- Spezifikationen, z. B. fachliche Anforderungen, technische Anforderungen und Sicherheitsanforderungen
- Epics, User-Stories und Abnahmekriterien
- Architektur und Entwurfsspezifikationen
- Benutzeranleitungen, Handbücher
- Verträge, Projektpläne, Zeitpläne und Budgetpläne
- Modelle wie Aktivitätsdiagramme oder Zustandsdiagramme
- Programm Quellcode
- Testkonzepte, Testspezifikationen, Testskripte, Testberichte
- Webseiten

Manuelle vs. automatisierte Prüfung

- Betrachtung des Prüfobjekts durch Mensch oder Werkzeug
- **Reviews**
 - Manuelle Prüfungen
 - Menschliche Denkfähigkeit zur Prüfung komplexer Sachverhalte nutzen
 - bei allen Arten von Dokumenten durchführbar
- **Statische Analyse**
 - Automatisierte Prüfungen gegen Regeln durch geeignete Werkzeuge
 - Nur bei Dokumenten mit formaler Struktur möglich (z. B. Programmcode, UML-Diagramme)



Manuelle Prüftechniken

- nicht werkzeuggestützt
- menschliche Denkfähigkeit nutzen, um komplexe Sachverhalte zu prüfen
- intensives Lesen und Nachvollziehen der untersuchten Dokumente
- verschiedene Vorgehensweisen zur Überprüfung der Dokumente möglich
- Unterscheidung durch Intensität und benötigte Ressourcen





Zunächst ein Selbsttest (1 von 2)

- Wie viele "F" kommen im folgenden Text vor?

FINISHED FILES ARE THE RESULT OF YEARS OF SCIENTIFIC STUDY COMBINED WITH THE EXPERIENCE OF YEARS



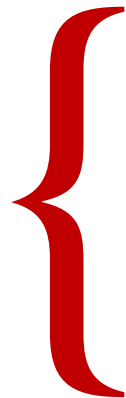
Zunächst ein Selbsttest (2 von 2)

- Wie viele hatten Sie gezählt?
- Es sind sechs! Drei zu finden ist normal - sechs wirklich gut!

FINISHED **F**ILES ARE THE RE-
SULT OF **F** YEARS OF **F** SCIENTI**F**-
IC STUDY COMBINED WITH THE
EXPERIENCE OF **F** YEARS

Kapitel 3

Statischer Test



Grundlagen

Reviews

Exkurs: Statische Analysen

Exkurs: Kontroll- und Datenflussanalyse

Exkurs: Metriken

Reviews

Review (survey)

- Duden: Übersicht, Überblick, Rückschau
- **ISTQB: Eine Art des statischen Tests, während der ein Arbeitsprodukt oder ein Prozess von einer oder mehreren Personen beurteilt werden, um Befunde zu erheben und um Verbesserungspotentiale zu identifizieren.**
- Bezeichnung für Vorgehen bei Prüfung von Dokumenteninhalten
 - oft die einzige Möglichkeit
- Oberbegriff für von Personen durchgeführte statische Prüfverfahren
- ISO 20246:2017 ist der Standard für das Review von Arbeitsergebnissen

ISO/IEC 20246 - Work Product Reviews - Contents

- Foreword
- Introduction
- 1 Scope
- 2 Conformance
- 3 Normative References
- 4 Terms and Definitions
- 5 Work Product Reviews - Introduction
- 6 Software Review Process
- 7 Review Techniques
- Annex A (informative) Review Documentation
- Annex B (informative) Review Roles
- Annex C (informative) Review Attributes
- Annex D (informative) Review Types (and attribute mapping)
- Annex E (informative) Mapping to IEEE 1028
- Annex F (informative) Reviews - Life Cycle Mapping
- Annex G (informative) Review Measurement & Improvement
- Annex H (informative) Tool Support
- Annex I (informative) Bibliography



Reviews („Was wird explizit einem Review unterzogen“?)

Zu reviewende Dokumente („documents under review“ - DuRs)



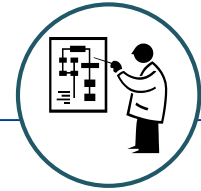
Reviews: Grundlegende Arbeitsschritte

5 Arbeitsschritte bei
(formalem) Review

Hinweis:
Die Darstellung
der Arbeitsschritte
orientiert sich
an der
ISO/IEC 20246!



Reviews: Planung (und Vorbereitung)



- **Definition des Reviewumfangs („review scope“)**

- Was wird einem Review unterzogen?
- Zweck des Reviews?
- Unterstützenden Informationen (z.B. Standards) verfügbar?
- Qualitätsmerkmale bewerten? Welche?
- Eingangs- und Endekriterien (für formale Reviewarten)
- Schätzung von Aufwand und Zeitbedarf

- **Identifiziere die Reviewmerkmale und –Charakteristiken**

- Reviewarten (Informelles Review, Walkthrough, Inspektion)
- Reviewrollen (Autor, Reviewleiter, ...)
- Reviewaktivitäten (Planung, ..., Fehlerbehebung und Bericht)

- **Auswahl und Festlegung geeigneter Reviewer**

- Auswahl der Teilnehmer des Reviews
- Klärung Verfügbarkeit der Teilnehmer
- Klärung Reviewrollen und Fokus (mit jedem Reviewer)



Reviews: Reviewbeginn (Kick-Off)



- **Verteilung der Reviewdokumente (so früh wie möglich)**

- Arbeitsprodukt, welches einem Review unterzogen werden soll
- Checklisten oder Listen von Prüfkriterien
- Dokumentvorlage für Reviewbefunde
- Verteilung physisch oder elektronisch

- **KICK-OFF / Überblicksmeeting**

- **Reviewleiter**

- präsentiert Umfang und Schwerpunkte des Reviews
 - erläutert Umfang, Ziele, Prozess, Rollen und Arbeitsergebnisse
 - beantwortet Fragen der Teilnehmer

- **Reviewteilnehmer**

- notwendiger Aufwand für Review erbringbar?
 - geben formales Commitment ab



Reviews: Individuelles Review (Ind. Vorbereitung)



- Teilnehmer des Reviewteams bereiten sich individuell vor
- Reviewer: intensiv mit zur Verfügung gestellten Dokumenten auseinandersetzen
 - Beschränkung auf bestimmte Aspekte ratsam (z.B. Testbarkeit einer Anforderung)
 - Einteilung des Prüfobjekts ratsam (z.B. gezielte Auswahl verwendeter Checklisten)
- Notieren von erkannten potenziellen Fehlerzuständen, Empfehlungen, Fragen oder sonstigen Kommentaren



Reviews: Individuelles Review – Reviewverfahren



- unterschiedliche Reviewverfahren für alle Reviewarten anwendbar
- Effektivität der Verfahren variiert in Abhängigkeit von genutzter Reviewart
- Reviewverfahren nach ISO 20246:2017:
 - Ad-hoc-Review
 - checklistenbasiertes Review
 - szenariobasiertes Review
 - rollenbasiertes Review
 - perspektivisches Lesen

Ad-hoc-Review



- wenige / gar keine Vorgaben für Durchführung
- oft genutztes Verfahren
- Reviewer
 - Prüfobjekt von vorne nach hinten lesen
 - identifizieren und dokumentieren Befunde asap
- Erwartung: Reviewer erfassen möglichst viele Befunde & Fehlerzustände
- Vorteile:
 - Erfordert wenig Vorbereitung
- Nachteile:
 - Erfolg hängt stark von den Fähigkeiten der Reviewer ab
 - Es kann zu vielen doppelt berichteten Befunden kommen

Checklistenbasiertes Review



- systematische Vorgehensweise
- Reviewer nutzt Checklisten
 - Reviewbeginn: Verteilung und Zuordnung der Checklisten zu Reviewern
- Fragen der Checkliste
 - Fokus auf potenzielle Fehlerzustände
 - aus Erfahrungen abgeleitet
 - auf Art des Prüfobjekts zugeschnitten
 - identifizierte und bekannte Risiken abdecken
 - regelmäßig überarbeiten (neue Erfahrungen einfließen lassen)
- Vorteile:
 - Systematische Abdeckung typischer Fehlerarten
- Nachteile:
 - Fehlerzustände außerhalb der Checkliste können übersehen werden
 - zeitintensiver als ein informelles Ad-hoc Review

Szenariobasiertes Review



- Reviewer bekommen strukturierte Richtlinien für Review
- Bewertung des Prüfobjekts bzgl. der Eignung für vorgegebene Szenarien
 - Verwendung des Prüfobjekts durch Endkunden
 - Weitere Entwicklung auf Basis des Prüfobjekts
 - Ableiten von Testfällen aus dem Prüfobjekt
 - ...
- Probeläufe mit dem Prüfobjekt auf Basis der erwarteten Nutzung
- Vorteile:
 - Systematische Abdeckung der vorgegebenen Szenarien
- Nachteile:
 - Fehlerzustände außerhalb der Szenarien können übersehen werden

Rollenbasiertes Review



- Reviewer bewerten das Prüfobjekt aus der Perspektive individueller Stakeholder-Rollen, z.B.
 - spezifische Arten von Endanwendern (erfahren, unerfahren, Senioren, Kinder usw.)
 - spezifische Rollen innerhalb einer Organisation (Benutzeradministrator, Systemadministrator, Performanztester usw.)
- perspektivisches Lesen ist ein Spezialfall des rollenbasierten Reviews (→ nächste Folie)
- Vorteile:
 - Stakeholder-spezifische Fehlerzustände können gefunden werden
- Nachteile:
 - Fehlerzustände außerhalb der Rollen können übersehen werden
 - Manche Reviewer haben Schwierigkeiten, andere Perspektiven einzunehmen

Perspektivisches Lesen

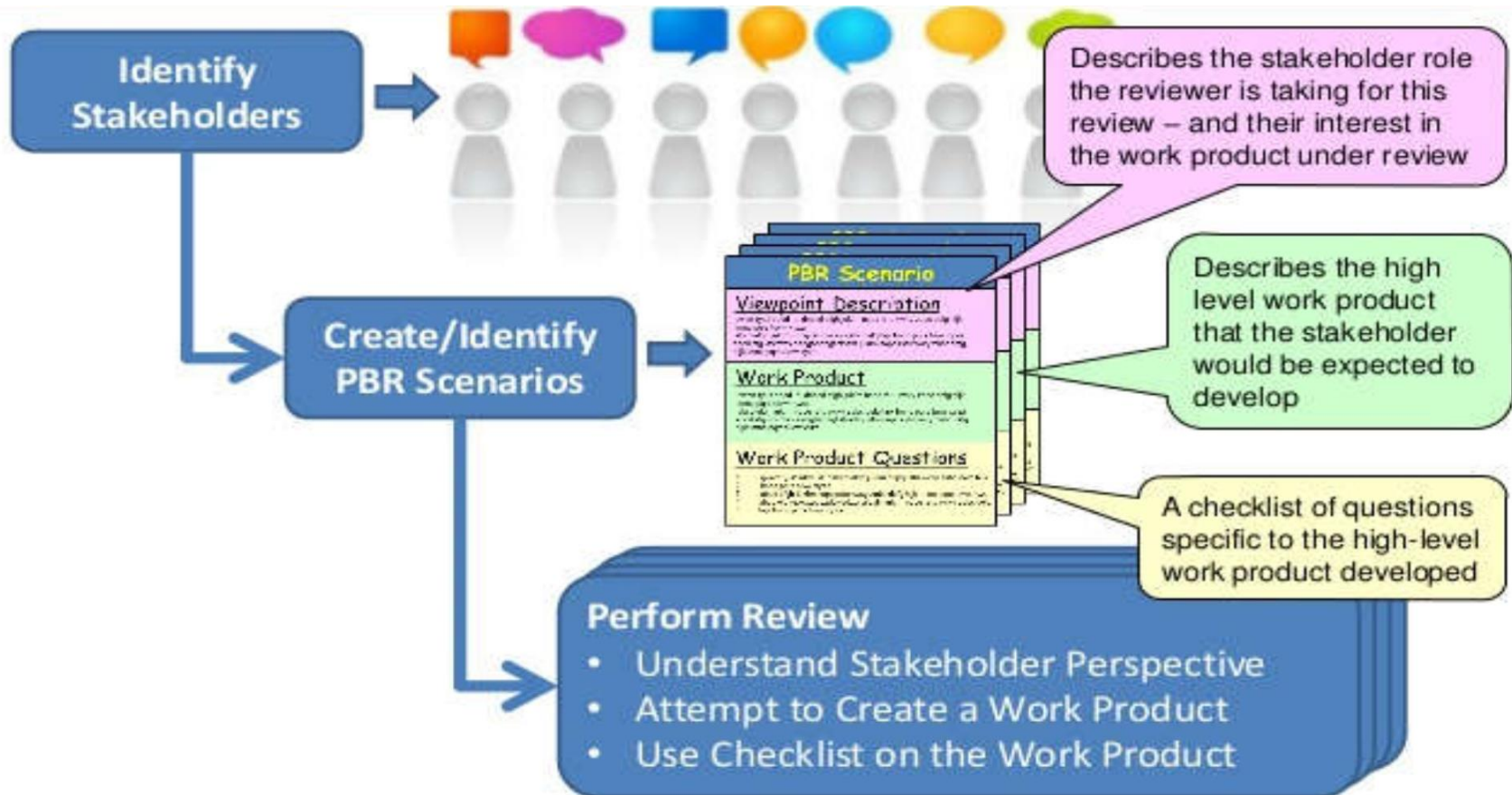


Perspektivisches Lesen



- effektivstes Reviewverfahren für technische Dokumente
- Reviewer übernehmen Standpunkte unterschiedlicher Stakeholder
 - Endanwender, Marketing, Designer, Tester, Betrieb, Regulierer
- Prüfobjekt so **nutzen**, wie es der Stakeholder tun würde
 - Tester leitet Testfälle ab
 - Endnutzer bedienen die entworfene Benutzungsschnittstelle
- Wichtig: angemessenes Einbeziehen der unterschiedlichen Standpunkte, basierend auf Risiken
- Vorteile:
 - mehr Tiefe im individuellen Review
 - weniger Doppelungen von Befunden
- Nachteile:
 - Annahme anderer Perspektiven schwierig (passende Checklisten??)

Perspektivisches Lesen (Perspective-based reading - PBR)



Reviewverfahren – Zusammenfassung



Ad hoc

- Weitverbreitet
- Personenabhängig, führt oft zu doppelten Befunden

Checklistenbasiert

- Deckt systematisch typische Fehler ab

Szenarien und Probeläufe


- Orientieren sich an der erwarteten Benutzung

Rollenbasiert

- Aus der Perspektive individueller Rollen

Perspektivisch

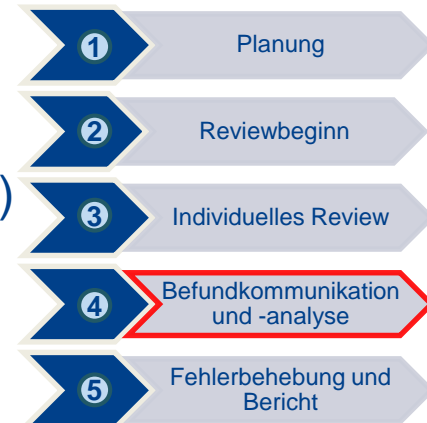
- Nachgewiesen effektiv

 Basis: ISO 20246:2017 Software and systems engineering - Work product reviews

Reviews: Befundkommunikation und -analyse



- Kommunikation identifizierter potenzieller Fehlerzustände (z.B. in einer Reviewsitzung)
- Analyse potenzieller Fehlerzustände, Zuweisung von Zuständigkeit (z.B. Autor, ...) und Status (z.B. abgelehnt, ...)
- Bewertung und Dokumentation von Qualitätsmerkmalen
- Bewertung der Reviewbefunde gegenüber den Endekriterien, um **Reviewentscheidung bzgl. Reviewobjekt** zu treffen
 - annehmen, ablehnen, umfangreiche Änderungen notwendig, geringfügige Änderungen notwendig



Quellen zur Reviewsitzung:

Frühauf, K.; Ludewig, J.; Sandmayr, H.:
Software-Prüfung: eine Fibel.
vdf, Verlag der Fachvereine, Zürich, 4. Aufl. 2000

Rösler, P.; Schlich, M.; Kneuper, R.:
Reviews in der System- und Softwareentwicklung, dpunkt.verlag, 2013

Reviews: Reviewsitzung (1 von 4)



- Vom Reviewmoderator (kurz Moderator) geführt
 - kann Sitzung absagen oder abubrechen, wenn (min. ein) Reviewer nicht erscheinen oder ungenügend vorbereitet sind
 - Moderator ist kein Reviewer
- In der Regel auf festen Zeitrahmen beschränkt (max. 2 Stunden)
 - falls nötig weitere Sitzung einberufen
 - frühestens am nächsten Tag
- Das Prüfobjekt steht zur Diskussion, nicht der Autor !
 - Reviewer achten auf ihre Ausdrucksweise
 - Autor darf weder sich noch das Prüfobjekt verteidigen (d.h. der Autor sollte keinen Angriffen ausgesetzt werden, die ihn zur Verteidigung zwingen)
 - Erläuterung seiner Entscheidungen wird teilweise als legitim und hilfreich angesehen
- Keine allgemeinen Stilfragen (außerhalb der Richtlinien) diskutieren

Reviews: Reviewsitzung (2 von 4)



- Entwicklung von Lösungen ist nicht Aufgabe des Reviewteams
- Jeder Reviewer muss Gelegenheit haben, seine Befunde zu präsentieren
- Befunde (z.B. Fehlerzustände) nicht in Form von Anweisungen an den Autor protokollieren
 - Anweisungen in Form konkreter Korrekturvorschläge allerdings teilweise durchaus sinnvoll und hilfreich für Qualitätsverbesserung
- Die Bewertung der einzelnen Befunde und die Gesamtbeurteilung sollte von allen Reviewern getragen werden
- Der Konsens der Reviewer zu einem Befund ist zu protokollieren

Reviews: Reviewsitzung (3 von 4)



Einzelne Befunde gewichten, z.B. als

- **Kritischer Fehler(zustand)**
Prüfling oder Prüfobjekt ist für den vorgesehenen Zweck unbrauchbar, muss vor der Freigabe behoben werden
- **Hauptfehler**
Nutzbarkeit des Prüflings beeinträchtigt, Fehlerzustand sollte vor der Freigabe behoben werden
- **Nebenfehler**
geringfügige Abweichung, z.B. Rechtschreibfehler oder Mangel im Ausdruck, beeinträchtigt den Nutzen kaum
- **kein Fehler**
fehlerfrei, bei Überarbeitung nicht ändern



Reviews: Reviewsitzung (4 von 4)



- Reviewteam gibt Empfehlung über die Annahme des Prüflings ab:
 - Akzeptieren → ohne Änderungen
 - Akzeptieren mit geringen Änderungen → kein weiteres Review
 - Nicht akzeptieren → weiteres Review oder andere Prüfmaßnahme erforderlich
- Protokoll enthält Liste aller Befunde (z.B. Fehlerzustände), die in der Sitzung diskutiert wurden, und zusätzlich
 - Informationen zum Prüfobjekt und den verwendeten Dokumenten
 - Beteiligte Personen und ihre Rollen
 - Kurze Zusammenfassung der wichtigen Punkte
 - Ergebnis der Reviewsitzung mit der Empfehlung der Reviewer
- Am Schluss geben alle Sitzungsteilnehmer das Protokoll frei

Reviews: Protokoll

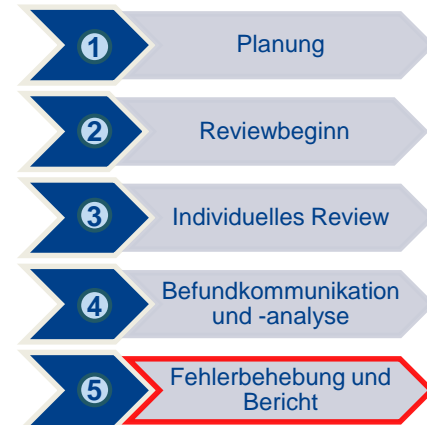
Beispiel einer einseitigen Zusammenfassung (Deckblatt des Protokolls)

Review					
Laufzettel und Ergebniszusammenfassung					
Projekt					
Autor:					
Dokument:			Version:		
Review-Art					
Planung					
Verantwortlich für die Durchführung des Review					(1)
Unterlagen geprüft:	Ja <input type="checkbox"/>	Datum:		Nein <input type="checkbox"/>	
Unterlagen verteilt:	Ja <input type="checkbox"/>	Datum:		Nein <input type="checkbox"/>	
Vorbereitung					
Inspektor 1			Dauer:		h (2)
Inspektor 2			Dauer:		h (2)
Inspektor 3			Dauer:		h (2)
Inspektor 4			Dauer:		h (2)
Inspektor 5			Dauer:		h (2)
Inspektor 6			Dauer:		h (2)
Review-Sitzung					
Moderator:					
Datum:		Teilnehmer:		Dauer:	h
			Summe Aufwand:		h (3)
Ergebnis					
Schwere Fehler:		Reinspektion erforderlich:	Ja <input type="checkbox"/>	Nein <input type="checkbox"/>	
Leichte Fehler:					
Unklarheiten:		Nacharbeit erledigt bis:			
Vorschläge:					
Dokument-Fehler:		Gesamtaufwand:			=(2)+(3)
Abschluss					
Arbeitsergebnis Ok, Nacharbeit fertig.					
Datum:		Unterschrift (1):			

Reviews: Fehlerbehebung und Bericht



- Beheben von im geprüften Arbeitsergebnis gefundenen Fehlerzuständen (üblicherweise durch den Autor)
- Kommunikation von Fehlerzuständen an zuständige Personen, wenn sie in einem Dokument gefunden wurden, das zu dem Prüfobjekt in Beziehung steht
- Aufzeichnung des aktualisierten Status der Fehlerzustände (in formalen Reviews)
- Sammeln von Metriken (formalere Reviewarten)
- Prüfen, ob Endekriterien erfüllt sind (formalere Reviewarten)
 - z.B. Fehlerzustände vollständig und korrekt behoben?
- Abnahme von Arbeitsergebnis, wenn Endekriterien erreicht
- ggf. Folgereview ansetzen, das aber oft kürzer ausfällt





Generic Review - Roles





Reviews: Rollen



Management

entscheidet über Durchführung von Reviews, stellt Zeit im Projektplan zur Verfügung und überprüft, ob Reviewziele erfüllt sind

Reviewleiter

verantwortet das Review und entscheidet über Teilnehmer, Ort, Zeitpunkt des Reviews.

Moderator

leitet Reviewsitzungen; falls nötig, vermittelt zwischen verschiedenen Standpunkten; Erfolg hängt häufig von ihm ab.

Autor

hauptverantwortlich für das zu prüfende Dokument (bzw. die zu prüfenden Dokumente); überarbeitet das Prüfobjekt nach dem Review.

Reviewer

identifizieren im Prüfobjekt Befunde (z.B. Fehlerzustände); haben spezifischen technischen oder fachlichen Hintergrund (auch Prüfer, Gutachter oder Inspektoren genannt)

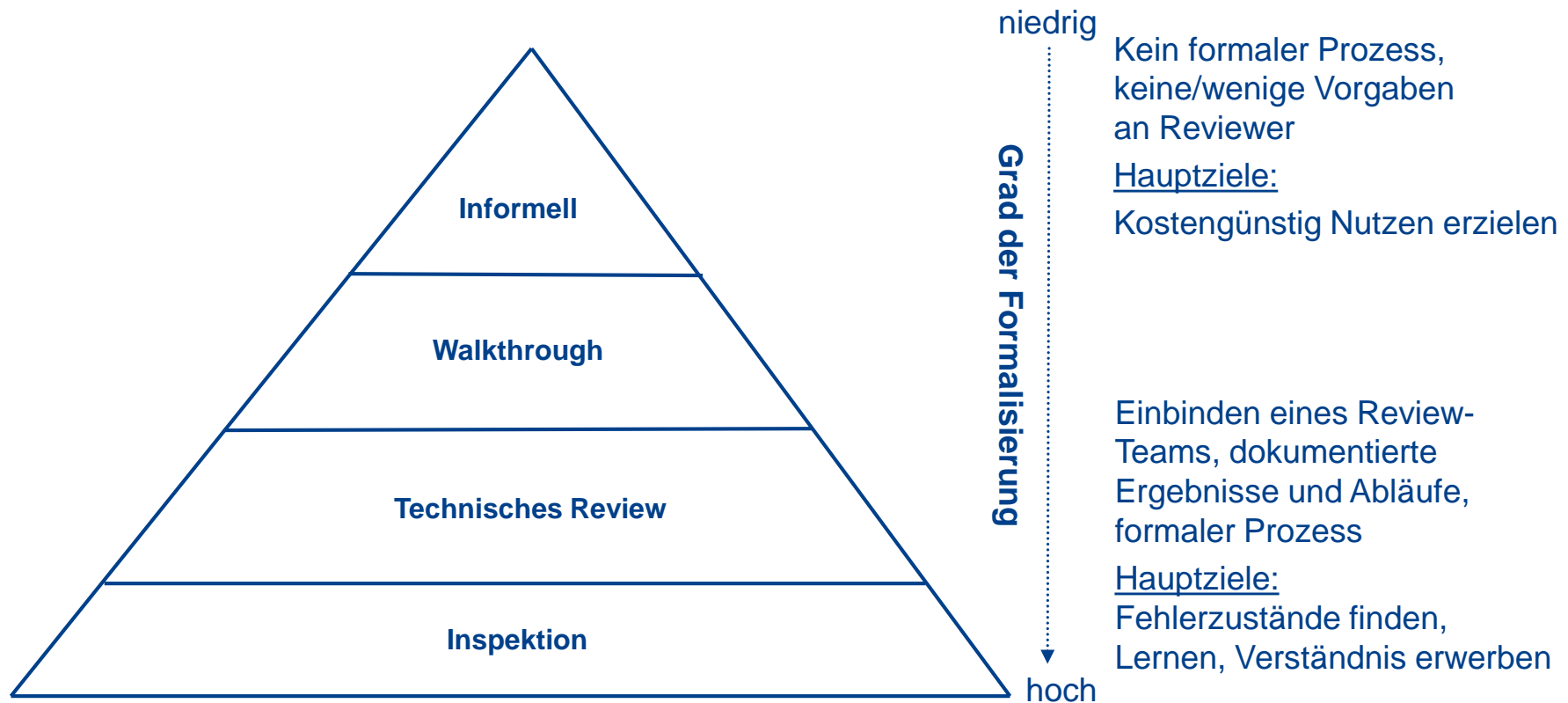
Protokollant

dokumentiert alle Ergebnisse, Probleme und offene Punkte, die im Verlauf der Sitzung identifiziert werden.



Reviews: Arten (1 von 2)

Reviews können in unterschiedlichen Formen durchgeführt werden



Reviews: Arten (2 von 2)

- Reviews variieren: sehr informell bis sehr formal (d.h. strukturiert)
- anzuwendender Reviewprozesses ist abhängig von Faktoren wie
 - Softwareentwicklungslebenszyklus-Modell
 - Reife des Entwicklungsprozesses
 - Komplexität der zu prüfenden Arbeitsergebnisse
 - rechtlichen oder regulatorischen Anforderungen
 - der Notwendigkeit für einen Prüfnachweis
- Reviewart ist abhängig von festgelegten Zielen des Reviews, z.B.
 - Finden von Fehlerzuständen
 - Verständnisgewinnung, Training von Testern oder neuen Mitgliedern
 - Diskussion und Entscheidung durch Konsens

Reviews: Arten (Gegenüberstellung)

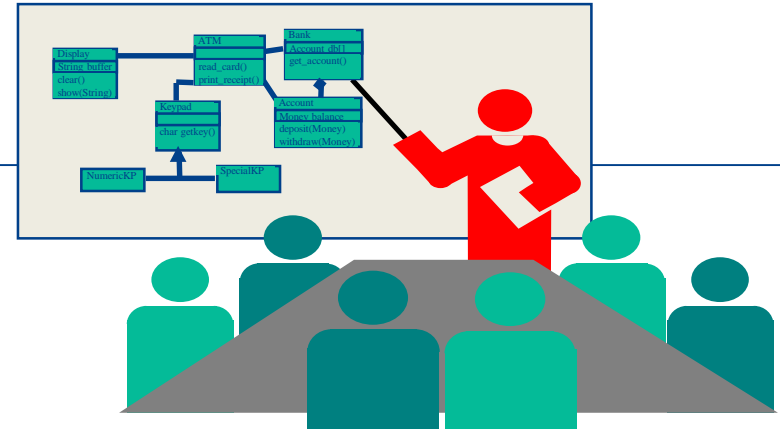
Reviewart	Definition nach ISTQB Glossar 3.2
Informelles Review	Eine Art von Reviews, die keinem formalen (dokumentierten) Ablauf folgt. [ISO 20246]
Walkthrough	Eine Reviewart, bei der ein Autor die Reviewteilnehmer durch ein Arbeitsergebnis leitet und die Teilnehmer Fragen stellen und potentielle Befunde kommentieren. [Nach ISO 20246]
Technisches Review	Eine formale Reviewart, bei der ein Team von technisch qualifizierten Personen die Eignung eines Arbeitsergebnisses für seinebeabsichtigte Verwendung prüft und Abweichungen von Spezifikationen oder Standards identifiziert. [Gilb and Graham, IEEE 1028]
Inspektion	Eine formale Reviewart deren Ziel die Identifizierung von Befunden in einem Arbeitsprodukt ist, und welche Messungen zur Verbesserung des Reviewprozesses und des Softwareentwicklungsprozesses liefert. [Nach ISO 20246]

Informelles Review

- Hauptzwecke:
 - Erkennen von potenziellen Fehlerzuständen
- Mögliche zusätzliche Zwecke:
 - Generieren von neuen Ideen oder Lösungen,
 - schnelle Lösung kleinerer Probleme
- Kostengünstiger Ansatz bei hohem Nutzen
- Basiert nicht auf einem formalen (dokumentierten) Prozess
- Kann von Kollegen des Autors durchgeführt werden (Buddy-Check)
- Reviewsitzung ist optional
- Dokumentation der Ergebnisse ist optional
- Nutzung von Checklisten ist optional
- Sehr verbreitet in der agilen Entwicklung
- Beim Pair Programming integriert in die Programmierung
- Der Nutzen ist abhängig von den Reviewern

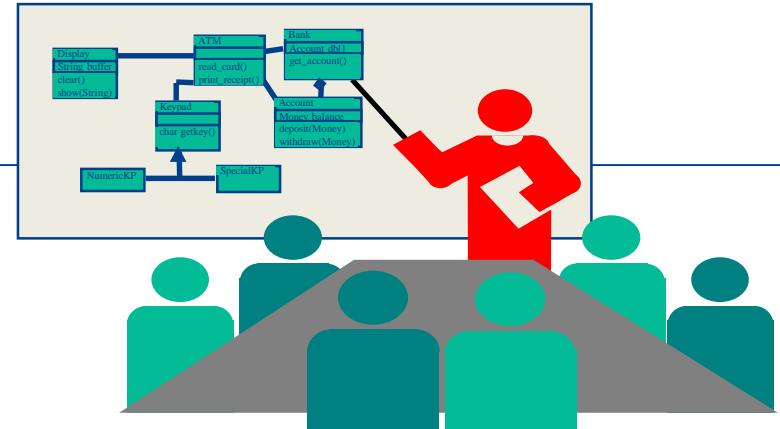


Walkthrough (1 von 3)



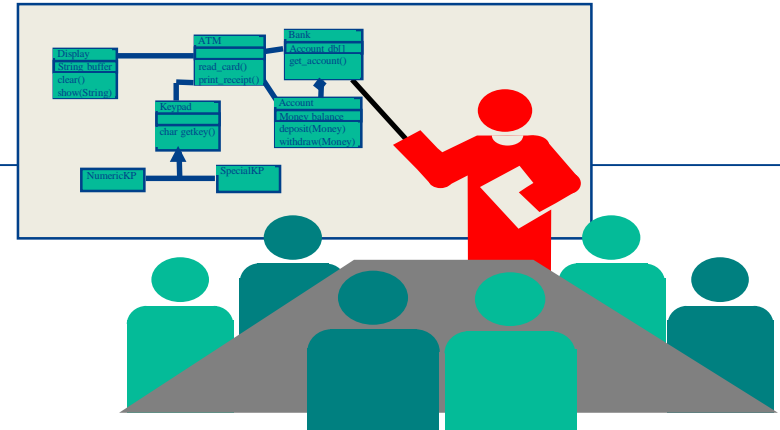
- Hauptzwecke:
 - Fehlerzustände finden
 - Softwareprodukt verbessern
 - Alternative Umsetzungen erwägen
 - Konformität mit Standards und Spezifikationen bewerten
- Mögliche zusätzliche Zwecke:
 - Ideenaustausch über Verfahren oder Stilvariationen
 - Ausbildung der Teilnehmer
 - Konsenserzielung
- Individuelle Vorbereitung vor der Reviewsitzung ist optional
- Protokollant ist obligatorisch
- Die Nutzung von Checklisten ist optional

Walkthrough (2 von 3)



- Schwerpunkt bildet die Sitzung (oft ohne Zeitbeschränkung)
- Vorbereitung hat im Vergleich zu den anderen Review-Arten den geringsten Umfang, teilweise kann sogar auf sie verzichtet werden
- Kann in Form von Szenarios, Dry Run (Probelauf) oder Simulationen durchgeführt werden
- Protokolle potenzieller Fehler und Reviewberichte können erstellt werden
- Kann in der Praxis von recht informell bis hin zu sehr formal variieren
- Geeignet für kleine Entwicklungsteams und für Prüfungen unkritischer Dokumente
- Wenig Aufwand, da Vor- und Nachbereitungen von geringem Umfang bzw. nicht zwingend erforderlich

Walkthrough (3 von 3)



- Autor ist oft Moderator und stellt das Prüfobjekt vor
 - Autor kann Verlauf des Walkthrough stark beeinflussen
 - Evtl. nachteilig für Ergebnis, wenn kritische Diskussion nicht intensiv genug
- Oft typische Anwendungsfälle ablaufforientiert durchgespielt
 - auch einzelne Testfälle möglich
 - Reviewer fragen meist spontan
- Nacharbeit: Autor verantwortlich, keine Kontrolle

Technisches Review (1 von 3)

- Hauptzwecke
 - Gewinnen von Konsens
 - Finden von potenziellen Fehlerzuständen
- Mögliche weitere Zwecke
 - Qualität bewerten und Vertrauen in das Prüfobjekt schaffen
 - Neue Ideen generieren
 - Autor motivieren und befähigen, zukünftige Arbeitsergebnisse zu verbessern
 - Alternative Umsetzungen bedenken
- Reviewer sollten fachliche Kollegen des Autors und fachliche Experten in der gleichen oder in anderen Disziplinen sein (Peers)

Technisches Review (2 von 3)

- Individuelle Vorbereitung ist erforderlich
 - Fachexperten als Reviewer prüfen das Prüfobjekt anhand der festgelegten Prüfkriterien
 - Reviewer halten Anmerkungen zu den einzelnen Prüfkriterien typischerweise schriftlich fest und übergeben sie vorab dem Moderator
 - Moderator priorisiert die Anmerkungen nach vermuteter Wichtigkeit
- Die Nutzung von Checklisten ist optional

Technisches Review (3 von 3)

- Reviewsitzung (ist sinnvoll, aber optional)
 - Idealerweise durch einen geschulten Moderator geleitet (nicht durch den Autor)
 - Zuerst die wesentlichen Anmerkungen diskutieren, wenn dann noch Zeit ist, dann die übrigen
 - Protokollführung ist obligatorisch: Protokollant (idealerweise nicht der Autor) dokumentiert alle Aussagen in der Reviewsitzung und erstellt Reviewbericht
- Auch ohne Reviewsitzung gibt es normalerweise einen Reviewbericht

Inspektion (1 von 3)



- Hauptzwecke:
 - Erkennen potenzieller Fehlerzustände
 - Bewerten der Qualität und Vertrauen in das Arbeitsergebnis schaffen
 - Durch das Lernen des Autors und Grundursachenanalyse zukünftige ähnliche Fehlerzustände verhindern
- Mögliche weitere Zwecke:
 - Autor motivieren und befähigen, zukünftige Arbeitsergebnisse und den Softwareentwicklungsprozess zu verbessern
 - Erzielen von Konsens
- Folgt einem definierten Prozess mit formalen dokumentierten Ergebnissen, basierend auf Regeln und Checklisten
- Nutzt klar definierte Rollen, die verpflichtend sind
- Individuelle Vorbereitung vor der Reviewsitzung ist erforderlich

Inspektion (2 von 3)



- Reviewer sind entweder gleichrangig mit dem Autor oder Experten in anderen Fachrichtungen, die für das Arbeitsergebnis relevant sind
- Protokollant ist obligatorisch
- Die Reviewsitzung wird von einem geschulten Moderator geleitet
- Der Autor kann nicht als Reviewleiter, Moderator oder Protokollant agieren
- Festgelegte Eingangs- und Endekriterien werden genutzt
- Protokolle potenzieller Fehler und Reviewberichte werden erstellt
- Metriken werden gesammelt und genutzt, um den gesamten Softwareentwicklungsprozess zu verbessern, einschließlich des Inspektionsprozesses

Inspektion (3 von 3)



- Reviewsitzung
 - kurze Einführung, z.B. Moderator trägt in straffer und logischer Weise die Inhalte des Prüfobjektes vor; evtl. auch Passagen vorlesen
 - Reviewer hinterfragen und diskutieren ausgewählte Aspekte intensiv
 - Autor beantwortet Fragen, bleibt sonst passiv
 - Moderator verhindert das Abschweifen der Diskussion
 - Protokollant erfasst die festgestellten Unstimmigkeiten, Fehlerzustände
 - abschließend Protokoll vorstellen, von Allen auf Vollständigkeit prüfen
- Inspektionsergebnisse
 - Qualitätsbeurteilung des untersuchten Dokuments
 - Bewertungen des Entwicklungs- und Inspektionsprozesses

Reviewarten: Auswahlkriterien (1 von 2)

- **Erlaubt der Formalisierungsgrad des Prüfobjekts werkzeuggestützte Analysen?**
 - Ja ⇒ Zunächst Werkzeuge einsetzen
 - Nein ⇒ Eine der Review-Arten einsetzen
- **Wie ist der geforderte Formalisierungsgrad des Prüfungsergebnisses?**
 - Ausführliche formalisierte Dokumentation
 ⇒ Inspektion oder technisches Review
 - Undokumentierte Umsetzung der Prüfergebnisse
 ⇒ Walkthrough oder Informelles Review
- **Ist Fachwissen aus mehreren Gebieten erforderlich?**
 - Ja ⇒ Technisches Review
 - Nein ⇒ Walkthrough oder Inspektion
- **Wie viel zusätzliches Fachwissen über das Prüfobjekt benötigen die Reviewer?**
 - Viel ⇒ Zunächst Walkthrough durchführen
 - Wenig ⇒ Direkt mit Inspektion oder technischem Review beginnen

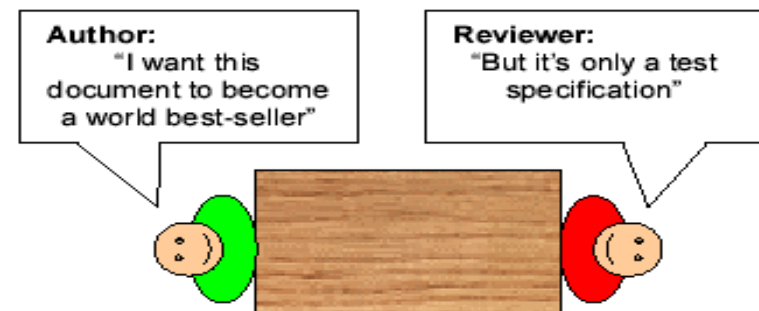
Reviewarten: Auswahlkriterien (2 von 2)

- **Terminkoordination einfach oder schwierig?**
(Fünf bis sieben Fachkräfte für einen oder mehrere Termine zu verpflichten kann sehr aufwändig sein)
 - Einfach ⇒ Inspektion oder Technisches Review
 - Schwierig ⇒ Walkthrough oder Informelles Review
- **Steht der Vorbereitungsaufwand in einem angemessenen Verhältnis zum erwarteten Ergebnis?**
 - Ja ⇒ Inspektion oder Technisches Review
 - Nein ⇒ Walkthrough oder Informelles Review
- **Wie hoch ist das Engagement der vorgesehenen Reviewer?**
 - Hoch ⇒ Inspektion oder Technisches Review
 - Niedrig ⇒ Walkthrough
(oder: gar kein Review durchführen, weil bei wenig Engagement meist auch wenig herauskommt)

Reviews: Organisatorische Erfolgsfaktoren (1 von 2)

- Jedes Review mit klaren Zielen
 - während Reviewplanung definiert, als Endekriterien genutzt
- Adequate Reviewarten genutzt
 - Passend für Ziele, geeignet sind für Art und Stufe des Softwarearbeitsergebnisses sowie der Teilnehmer
- Alle genutzten Reviewverfahren sind geeignet für die effektive Identifizierung von Fehlerzuständen im Arbeitsergebnis
- Alle Checklisten gehen auf wichtige Risiken ein und sind aktuell

Have a reasonable objective for the review



Reviews: Organisatorische Erfolgsfaktoren (2 von 2)

- Große Dokumente
 - in kleinen Teilen geschrieben
 - in Reviews geprüft
 - Qualitätslenkung: Autoren erhalten frühe, häufige Rückmeldungen
- Teilnehmer haben ausreichend Zeit für die Vorbereitung
- Reviews werden mit angemessener Vorankündigung geplant
- Das Management unterstützt den Reviewprozess (z.B. durch Bereitstellen angemessener Zeiträume für Reviewaktivitäten innerhalb der Projektpläne)

Reviews: Personenbezogene Erfolgsfaktoren (1 von 2)

- richtige Personen involvieren, um Reviewziele zu erreichen
 - Personen mit unterschiedlichen Fähigkeitsprofilen oder Sichtweisen, die das Dokument als Basis für ihre Arbeit nutzen könnten
- Tester als Reviewer wertgeschätzt
 - tragen zum Review bei
 - lernen das Arbeitsergebnis kennen, Vorbereitung für effektivere Tests
- Teilnehmer widmen den Details angemessene Zeit und Aufmerksamkeit
- Reviews in kleinen Schritten durchführen, sodass die Reviewer konzentriert bleiben



Reviews: Personenbezogene Erfolgsfaktoren (2 von 2)

- Gefundene Fehlerzustände werden objektiv anerkannt, bewertet und behandelt
- Die Reviewsitzung ist gut geleitet, so dass die Teilnehmer sie als wertvolle Nutzung ihrer Zeit ansehen
- Das Review wird in einer vertrauensvollen Atmosphäre abgehalten; das Ergebnis wird nicht für die Bewertung der Teilnehmer herangezogen
- Die Teilnehmer vermeiden Körpersprache und Verhaltensweisen, die Langeweile, Frust oder Feindseligkeit gegenüber anderen Teilnehmern ausdrücken
- Angemessene Schulungen, insbesondere für formale Reviewarten wie Inspektionen, werden bereitgestellt
- Eine Kultur des Lernens und der Prozessverbesserungen wird gefördert

Reviews: Vorteile (1 von 2)

- Fehlerbeseitigung kostet Ressourcen
 - frühes Erkennen führt zu Produktivitätssteigerung in Entwicklung
- Resultate:
 - oft kürzere Entwicklungszeiten
 - weniger Fehler im dynamischen Test, weniger Aufwand
 - Reduzierte Fehlerhäufigkeit im Einsatz des Systems
- Kostenreduzierung während der Produkt-Lebenszeit zu erwarten
 - Kundenwünsche in den Anforderungen werden oft durch Reviews geprüft
 - Anzahl der Änderungswünsche nach Inbetriebnahme verringern

Reviews: Vorteile (2 von 2)

- Überprüfungen im Team führen zu Wissensaustausch
 - Arbeitsmethoden, Wissen der einzelnen Personen verbessert
 - Qualität der Prozesse und der nachfolgenden Produkte dieser Personen verbessert
- Beteiligung mehrerer Personen erfordert verständliche Darstellung
- Allein die klare Darlegung bringt den Autor zu neuen Einsichten
- Verantwortung für Qualität des Prüfobjekts liegt beim Team

Reviews kosten und sparen Geld!

- Kosten für Reviews: ca. 10%-15% des Entwicklungsbudgets
- Einsparungen zwischen 14% und 25% möglich (nach Abzug der Mehrkosten für die Reviews)
- Bei effektiver Nutzung: bis zu 70% der Fehler in einem Dokument finden
- Reduzierung der Fehlerkosten bis zu 75%

Zahlenwerte aus

Gilb, T.; Graham, D.: Software Inspections. Addison-Wesley, 1996
und

Bush, M.: Software Quality: The use of formal inspections at the Jet
Propulsion Laboratory. In: Proc. 12th ICSE, IEEE 1990, 196-199
und

Frühauf, K.; Ludewig, J.; Sandmayr, H.: Software-Prüfung: eine Fibel.
vdf, Verlag der Fachvereine, Zürich, 4. Aufl. 2000

Reviews: Fallstricke (1 von 2)

- Benötigtes Personal ist nicht in ausreichendem Maße vorhanden oder verfügt nicht über die erforderliche Ausbildung
 - Schulungsmaßnahmen durchführen
 - entsprechendes Personal von Beratungsunternehmen ausleihen (Fähigkeiten des Moderators besonders wichtig)
- Fehlende Zeit (Fehleinschätzung bei der Ressourcen-Planung)
 - Evtl. kann eine weniger aufwändige Reviewart Abhilfe schaffen
- Fehlende Vorbereitung
 - Andere Reviewer auswählen
 - Bei fehlender Einsicht der Reviewer in die Bedeutung der Reviews und ihren hohen Wirkungsgrad zur Qualitätssteigerung der untersuchten Dokumente ist dies mit entsprechendem Zahlenmaterial zu belegen

Reviews: Fallstricke (2 von 2)

- Fehlende oder unklare Ziele
 - Fehler(zustände) finden? Prozess verbessern? Menschen „anschwärzen“?
 - Ziele für Review während der Reviewplanung festlegen (vor der Reviewdurchführung!)
- Fehlende oder unzureichende Dokumentation
 - Vorab prüfen, ob alle benötigten Dokumente in ausreichender Beschreibungstiefe vorhanden sind. Nur dann Review durchführen!
- Fehlende Unterstützung durch Management (in der Praxis leider oft)
 - Reviewprozess nicht erfolgreich, wenn die benötigten Ressourcen nicht bereitgestellt oder die erzielten Ergebnisse nicht zur Prozessverbesserung genutzt werden

Reviews: Tipps (1 von 2)

- Dokumente mit formaler Struktur (z.B. Code) vorab mit einem geeigneten Werkzeug analysieren
 - viele Aspekte automatisch erkennbar, Review dafür nicht notwendig
- gute Vorbereitung aller Personen ist Voraussetzung für erfolgreiche Reviewsitzung
- Reviewprozess verbessern durch Auswertung der Reviewsitzungen
 - Richtlinien und Checklisten aktualisieren
- Gefundene Fehler(zustände) positiv aufnehmen, objektiv ansprechen

Reviews: Tipps (2 von 2)

- Mit den Fragen der Beteiligten konstruktiv umgehen, psychologische Aspekte beachten
 - z.B. das Review für den Autor zu einer positiven Erfahrung werden lassen
- Wiederkehrende oder häufig vorkommende Fehlerarten weisen auf Defizite im Softwareentwicklungsprozess oder im Fachwissen der jeweiligen Personen hin
 - Notwendige Verbesserung des Entwicklungsprozesses planen und umsetzen
 - Mangel an Fachwissen durch Schulung ausgleichen

Kapitel 3

Statischer Test



Grundlagen

Reviews

Exkurs: Statische Analysen

Exkurs: Kontroll- und Datenflussanalyse

Exkurs: Metriken

Statische Analysen

- **Ziel: Aufdeckung vorhandener Fehlerzustände oder fehlerträchtiger Stellen in einem Dokument**
 - keine Ausführung der Prüfobjekte (eines Programms)
- Beispiele
 - Rechtschreibprüfprogramme, die (Rechtschreib- und teilweise Grammatik-) Fehler in Texten finden
 - Compiler führen eine statische Analyse des Programmtextes durch, indem sie die Einhaltung der Syntax der Programmiersprache prüfen
 - Spezielle Werkzeuge (eine Art erweiterter Compiler) prüfen die Einhaltung von Programmierkonventionen
- Weiteres Ziel: Ermittlung von Metriken, um eine Qualität zu messen

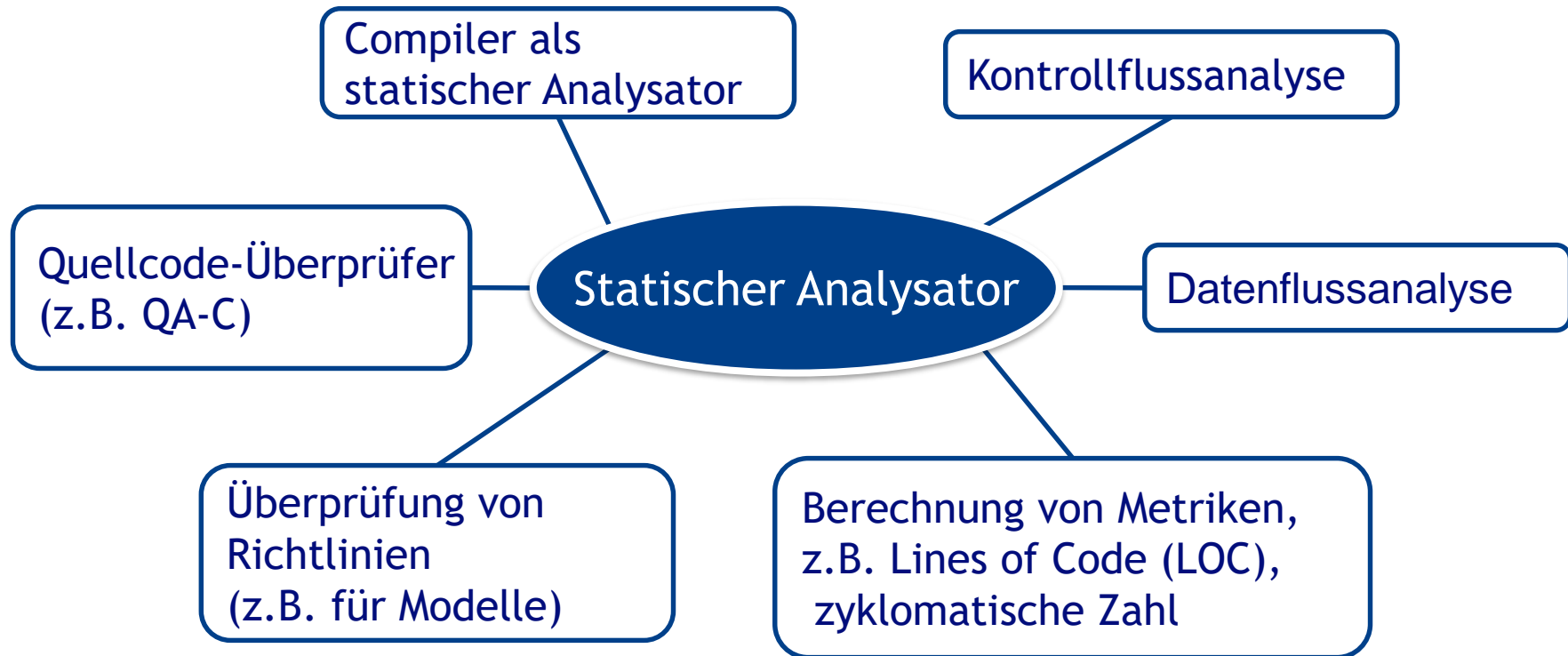
Produkt-Qualitätsmerkmale nach ISO 25010



Statische Analyse und Werkzeugunterstützung

- Statische Analyse ist nur mit Werkzeugunterstützung sinnvoll!
- zu analysierendes Dokument muss formale Struktur haben
 - nur so durch Werkzeug überprüfbar
- Beispiele für Dokumente, die einem Formalismus unterliegen können
 - Technische Anforderungen
 - Softwarearchitektur
 - Softwareentwurf
- Informeller Text kann unterhalb der Oberflächenstruktur (Rechtschreibung und elementare Grammatik) nur mit Methoden der KI (künstliche Intelligenz) analysiert werden
 - Linguistische semantische Analyse ist aktueller Forschungsgegenstand
 - Aber: Einführung einer Normsprache ermöglicht auch hier einfachere Analysen

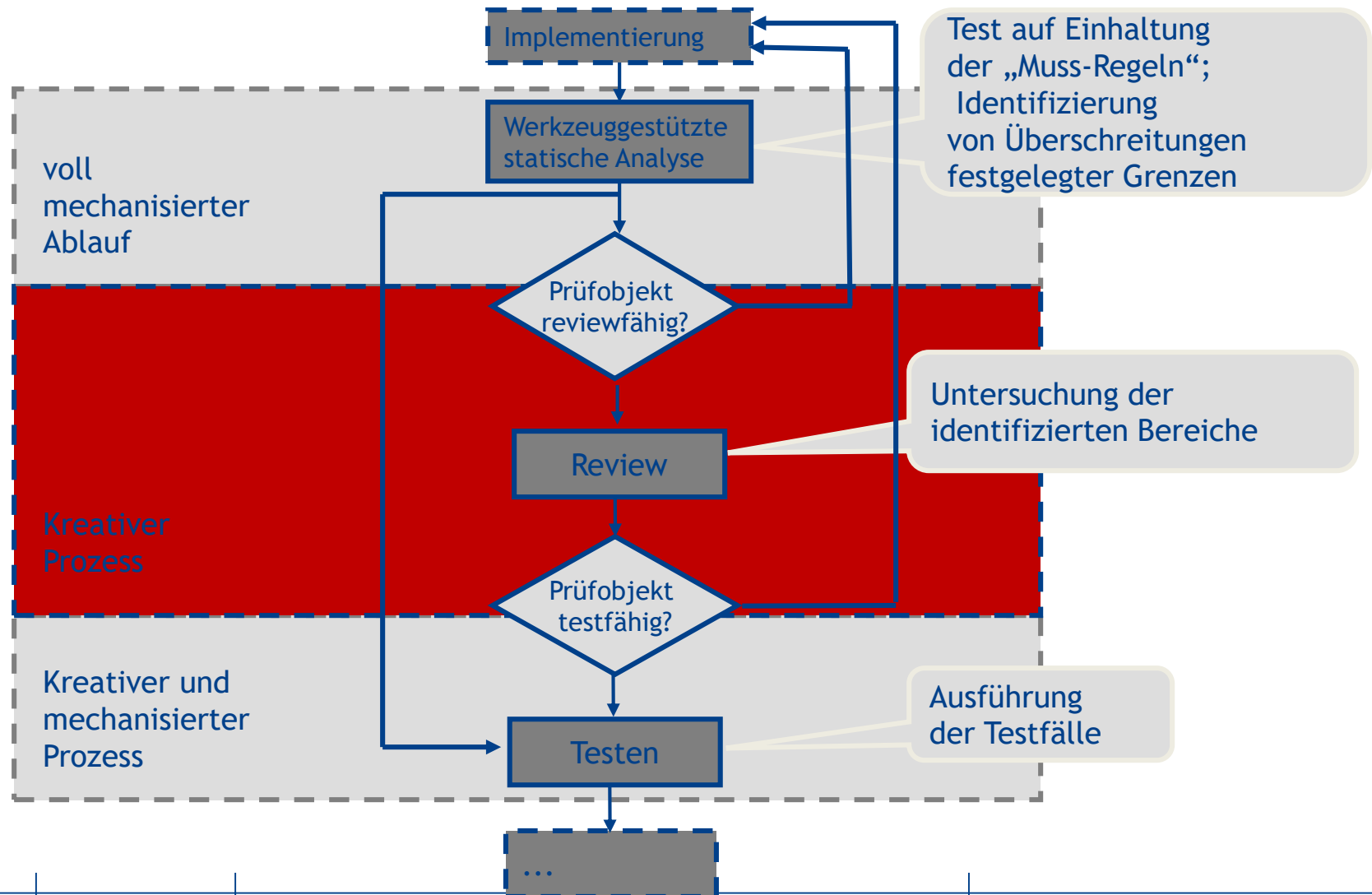
Elemente einer werkzeuggestützten statischen Analyse



Statische Analyse und Reviews

- Stehen in einem engen Zusammenhang
 - Erst statische Analyse, dann Review des formalen Dokumentes
 - die Menge der im Review zu betrachtenden Aspekte wird geringer
 - Da statische Analysen werkzeuggestützt durchgeführt werden, ist der Aufwand wesentlich niedriger als beim Review
- Vorgehen:
 - erst Statische Analyse durchführen und gefundene Fehlerzustände korrigieren (zumindest markieren)
 - danach Review durchführen

Statischer Test



Statischer Test

Welche Anomalien, Abweichungen und potentiellen Fehlerzustände können durch welche statischen Analyseansätze aufgezeigt werden?	Statische Analyseansätze									
	Compiler	Prüfung der Programmierkonventionen	Metriken	Datenflussanalyse	Kontrollflussanalyse	Schnittstellenanalyse	Impact Analyse		Review (z.B. Codereview)	Testen
unvollständige Variablendeklaration	X									
fehlende Klammern	X								X	
Verletzung von Programmierkonventionen (z.B. Namensgebungen, Methodenaufbau,...)		X								
Korrektes Eingrenzen (Einrücken) bestimmter Blöcke ("Pretty Printing")		X							X	
Verschachtelungstiefe			X							
Lines of Code			X							
Berechnung der zyklomatischen Zahl			X							
dd - Anomalie				X						
ur - Anomalie				X						
du- Anomalien				X						
unerreichbarer Code					X		X			
Falsche Anzahl übergebener Parameter						X				
Falsches Format übergebener Parameter						X				
Aufruf nicht existierender Programme							X			
Falsch eingesetzte Vergleichsoperatoren								X		X
unverständliche Kommentare								X		
nichtssagende Variablennamen								X		
Falsche Berechnungsreihenfolge (durch z.B. falsche Kammersetzung)										X

Statische Analyse von Programm Quellcode

- Compiler führen statische Analyse des Programmtextes durch
 - prüfen Einhaltung der Syntax der jeweiligen Programmiersprache
- Viele Compiler bieten noch zusätzliche Informationen
- Neben den Compilern gibt es Werkzeuge (Analysatoren) für spezielle Analysen
- Mögliche Fehler(zustände) bzw. fehlerträchtige Situationen, die mit der statischen Analyse von Programmcode nachgewiesen werden können
 - Verletzung der Syntax
 - Abweichungen von Konventionen und Standards
 - Sicherheitslücken
 - Kontrollflussanomalien
 - Datenflussanomalien

Einhaltung von Konventionen und Standards

- Einhaltung der Programmier- oder Modellierkonventionen durch entsprechende Analysatoren nachweisbar
 - Überprüfung benötigt wenig Zeit und kaum Personalressourcen
 - weiterer Vorteil: Ist den Programmierern bzw. Software-Designern bewusst, dass der Programmcode bzw. die Modelle von einem Werkzeug auf Einhaltung der Richtlinien überprüft wird, ist deren Bereitschaft zu ihrer Umsetzung bedeutend höher als ohne eine automatische Prüfung
- Tipp: Nur solche Richtlinien in einem Projekt zulassen, zu denen Prüfwerkzeuge vorhanden sind (andere Vorschriften erweisen sich in der Praxis ohnehin schnell als bürokratischer Ballast)
- Aber: Werkzeuge für statische Analyse können große Mengen von Warnungen und Hinweisen erzeugen, die gut verwaltet werden müssen, um eine effektive Nutzung des Werkzeugs zu erlauben

Aufdecken von Sicherheitslücken (Security)

- Verwendung fehleranfälliger Programmkonstrukte und fehlende Überprüfung resultiert in Sicherheitsproblemen
- Beispiele:
 - Fehlendes Abfangen von Speicherüberläufen
 - Keine Überprüfung des Einhaltens von Datenbeschränkungen bei der Eingabe
- Analysewerkzeuge können diese Mängel aufdecken
 - Mängel unterliegen meist einem „Muster“
 - kann von Werkzeugen gesucht und analysiert werden

Kapitel 3

Statischer Test



Grundlagen

Reviews

Exkurs: Statische Analysen

Exkurs: Kontroll- und Datenflussanalyse

Exkurs: Metriken

Kontroll- und Datenflussanalyse

- Suche nach Anomalien im Programmtext
- Anomalie: Unstimmigkeit, die zu einer Fehlerwirkung führen kann, aber nicht zwingend dazu führt
 - anomal: unregelmäßig, regelwidrig (laut Lexikon)
 - Kann ein Fehlerzustand sein, muss es aber nicht
- Statische Analyse kann nicht alle Fehlerzustände und Unstimmigkeiten nachweisen: manche treten erst bei Ausführung (also zur Laufzeit) auf
 - Wird z. B. bei einer Division der Wert des Divisors in einer Variablen gehalten, so kann diese Variable zur Laufzeit den Wert Null annehmen, was zu einer Fehlerwirkung führt
 - Statisch ist dieser Fehlerzustand meist nicht erkennbar

Kontrollfluss

- Kontrollfluss: Abstrakte Repräsentation von allen möglichen Reihenfolgen von Ereignissen während einer Programmausführung
 - Abfolge der während eines Programmlaufs ausgeführten Anweisungen
 - Muss nicht die Reihenfolge der Anweisungen im Programmtext sein
- Wird durch steuernde Anweisungen determiniert
 - Unbedingte Sprünge
 - Bedingte Verzweigungen
 - Schleifen
- Der Wahrheitswert von Bedingungen, z.B. in IF-Anweisungen im Programmtext, steuert den Kontrollfluss
 - Ist der ermittelte Wahrheitswert der Bedingung wahr, dann wird die Ausführung des Programms mit dem THEN-Teil der IF-Anweisung fortgeführt, im anderen Fall (falsch) wird der ELSE-Teil durchlaufen
- Schleifen führen zu vorherigen Anweisungen zurück, wodurch deren mehrfache Ausführung bewirkt wird, oder der Schleifenkörper wird umgangen

Kontrollflussgraph

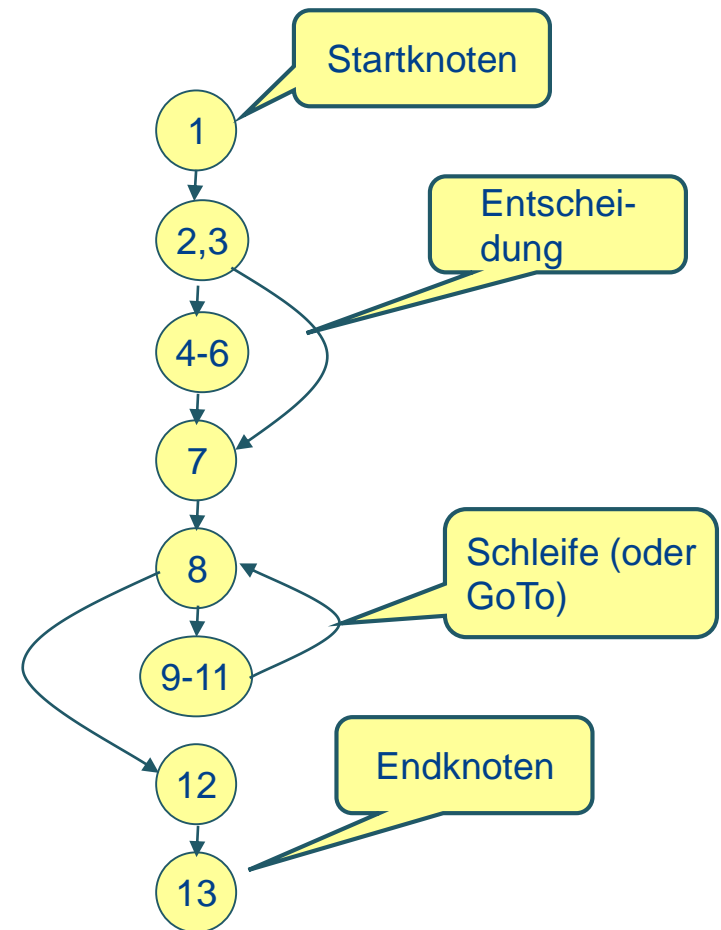
- Gerichteter Graph
- Knoten: Anweisungen des Programms
 - Sequenzen von Anweisungen können genau dann als ein einziger Knoten dargestellt werden, wenn aus der Ausführung der ersten Anweisung der Sequenz die Ausführung aller weiteren Anweisungen der Sequenz folgt
- Kanten: Mögliche Ausführungsreihenfolgen der durch die Knoten repräsentierten Anweisungen
 - Knoten mit mehreren ausgehenden Kanten repräsentieren den Kontrollfluss steuernde Anweisungen
 - IF-Anweisung und Schleifensteuerung: Zwei abgehende Kanten
 - CASE-Anweisung: mehrere abgehende Kanten
- Genau ein Startknoten (Programmanfang, Kopf der Methode)
- Genau ein Endknoten (Programmende, Ende der Methode)



Beispiel eines Kontrollflussgraphen

Bestimmung des größten gemeinsamen Teilers (ggT)
zweier ganzer Zahlen m und n:

```
1. public int ggt(int m, int n) {  
2.     int r;  
3.     if (n > m) {  
4.         r = m;  
5.         m = n;  
6.         n = r;  
7.     }  
8.     r = m % n;  
9.     while (r != 0) {  
10.        m = n;  
11.        n = r;  
12.        r = m % n;  
13.    }  
14.    return n;  
15. }
```



Kontrollflussanalyse

- Anschaulichkeit des Kontrollflussgraphen macht Abläufe durch ein Programmstück leicht erfassbar
 - Unübersichtliche (Teile des) Graphen mit unklaren Zusammenhängen und Abläufen erfordern eine Überarbeitung des Programmtextes, da Komplexität oft fehleranfällig
- Kontrollflussanomalie: Statisch feststellbare Unstimmigkeit beim Ablauf des Testobjekts (z.B. nicht erreichbare Anweisungen)
 - Sprünge aus Schleifen heraus
 - Sprünge in Schleifen hinein
 - Programmstücke mit mehreren Ausgängen
- Diese Unstimmigkeiten müssen keine Fehlerzustände sein, widersprechen aber den Grundsätzen der strukturierten Programmierung
- Voraussetzung: Graph nicht manuell, sondern von einem Werkzeug erstellt, das eine eins-zu-eins–Abbildung zwischen Programmtext und Graph gewährleistet

Kontrollflussanalyse: Vorgänger-Nachfolger-Tabellen

- Neben den Graphen können auch Vorgänger-Nachfolger-Tabellen erstellt werden, aus denen hervorgeht, ob und wie eine Anweisung mit anderen Anweisungen in Beziehung steht
- Gibt es zu einer Anweisung keinen Vorgänger, so ist diese Anweisung nicht erreichbar (sog. toter Code)
 - Ein Fehlerzustand oder zumindest eine Unstimmigkeit ist erkannt
 - Nur für die erste und letzte Anweisung eines Programm(teil)s gelten die Ausnahmen, dass es keine Vorgänger- bzw. Nachfolge-Anweisung geben darf
 - Für Programm(teil)e mit mehreren Eintritts- bzw. Austrittspunkten gilt entsprechendes



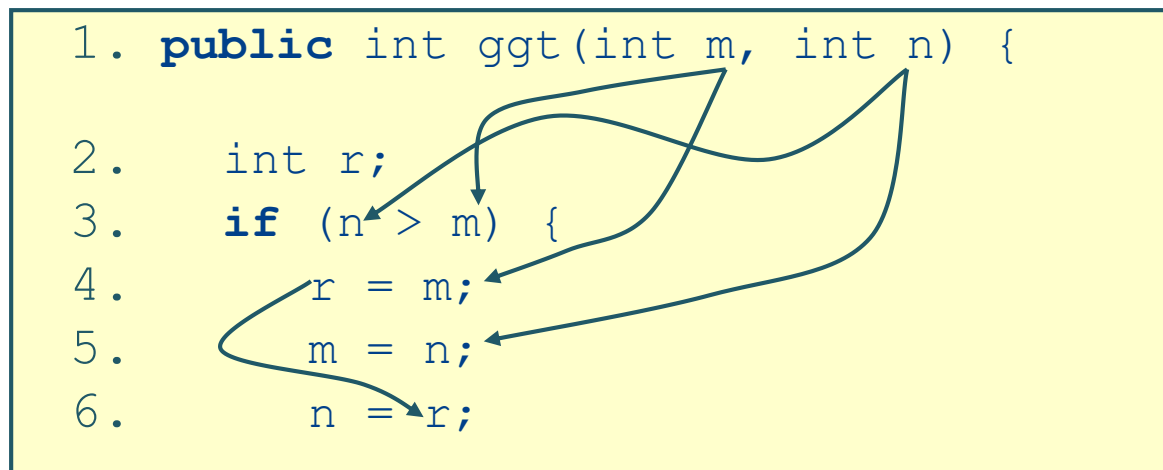
Beispiel: Vorgänger-Nachfolger-Tabelle

```
1. public int ggt(int m, int n) {  
2.     int r;  
3.     if (n > m) {  
4.         r = m;  
5.         m = n;  
6.         n = r;  
7.     }  
8.     r = m % n;  
9.     while (r != 0) {  
10.        m = n;  
11.        n = r;  
12.        r = m % n;  
13.    }  
14. }
```

Anwei- sung	Nach- folger	Vor- gänger
1	2	-
2	3	1
3	4, 7	2
4	5	3
5	6	4
6	7	5
7	8	3, 6
8	9, 12	7, 11
9	10	8
10	11	9
11	8	10
12	13	8
13	-	12

Datenflussanalyse

- Verwendung von Daten auf Pfaden durch den Programmcode



- Nicht immer können Fehlerzustände nachgewiesen werden, oft wird in diesem Zusammenhang dann ebenfalls von Anomalien oder Datenflussanomalien gesprochen,
 - Lesen einer Variablen ohne vorherige Initialisierung oder
 - Nicht-Verwendung eines zugewiesenen Wertes einer Variablen



Beispiel: Datenflussanomalien

```
1. void Tausch (int min, int max) { // d(min, max)
2.     int hilf;                      // u(hilf) (in Java hilf=0!!)
3.     if (min > max) {                // r(min, max)
4.         max = hilf;                // d(max), r(hilf)
5.         max = min;                 // d(max), r(min)
6.         hilf = min;                // d(hilf), r(min)
7.     }                              // u(hilf)
8. }
```

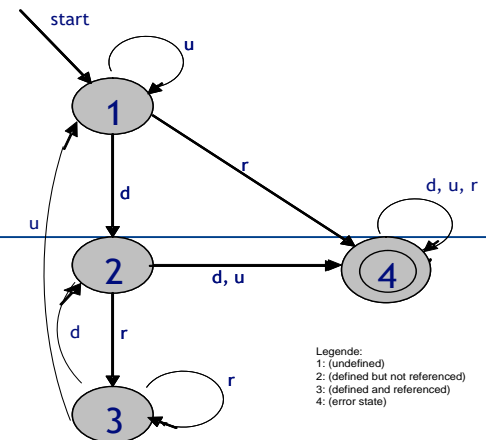
ur (hilf)

du (hilf)

dd (max)

Datenflussanomalien

- Analysiert wird die Verwendung jeder einzelnen Variablen
- Drei Verwendungen oder Zustände der Variablen
 - Undefined (**u**): die Variable hat keinen definierten Wert (z.B. am Programm-beginn oder wenn sie freigegeben oder ihr Gültigkeitsbereich verlassen wird)
 - Definiert (**d**): der Variablen wird ein Wert zugewiesen
 - Referenziert (**r**): der Wert der Variablen wird gelesen bzw. verwendet
- Drei Arten von Datenflussanomalien
 - **ur**-Anomalie: Ein undefinierter Wert (**u**) einer Variablen wird auf einem Programmpfad gelesen (**r**)
 - **du**-Anomalie: Die Variable erhält einen Wert (**d**) der allerdings ungültig (**u**) wird, ohne dass er zwischenzeitlich verwendet wurde
 - **dd**-Anomalie: Die Variable erhält auf einem Programmpfad ein zweites Mal einen Wert (**d**), ohne dass der erste Wert (**d**) verwendet wurde
- Zusammengefasst: Eine Datenfluss-Anomalie ist die
 - referenzierende Verwendung einer Variablen ohne vorherige Initialisierung
 - oder die Nicht-Verwendung eines Wertes einer Variablen



Bewertung der Datenflussanalyse

- Im Beispiel sind die Anomalien sehr offensichtlich
- Aber: Zwischen den betroffenen Anweisungen, die zu den Anomalien führen, können viele andere Anweisungen mit anderen Variablen stehen
 - Anomalien dann nicht mehr so offensichtlich
 - bei manueller Prüfung leicht zu übersehen
 - Werkzeug zur Datenflussanalyse deckt die Anomalien auf
- Nicht jede Anomalie führt direkt zu fehlerhaftem Verhalten
 - Beispielsweise hat eine du-Anomalie nicht immer direkte Auswirkungen, das Programm kann korrekt laufen
 - Aber: warum ist die Zuweisung an dieser Stelle des Programms vor dem Ende des Gültigkeitsbereichs der Variablen vorhanden?
 - genauere Untersuchung der anomalen Programmstellen ratsam, um weitere Unstimmigkeiten zu finden

Kapitel 3

Statischer Test



Grundlagen

Reviews

Exkurs: Statische Analysen

Exkurs: Kontroll- und Datenflussanalyse

Exkurs: Metriken

Maße und Messen in Softwareprojekten

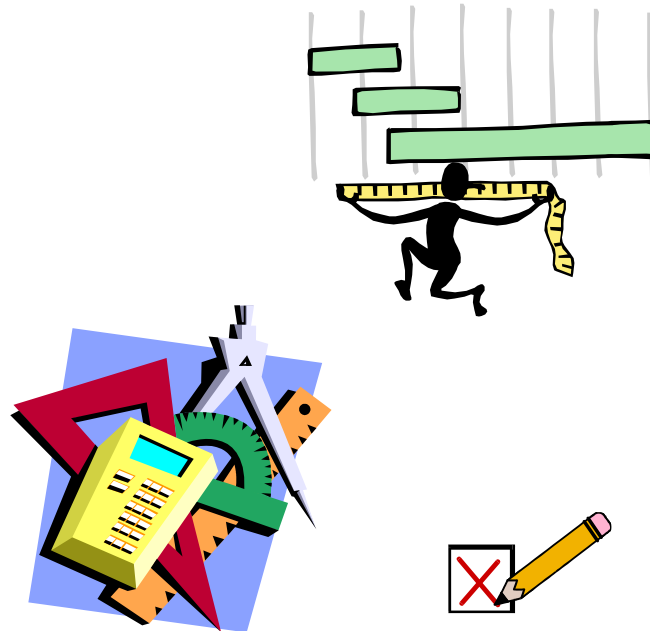
- Software-Metrik (Software-Messung) ist ein Zweig der Informatik, in dem Software-Prozess- und Produkteigenschaften sowie ihre Beziehungen zueinander quantifiziert und gemessen werden
- Grundlegende Fragen sind
 - Wie werden messbare Software-Merkmale ausgewählt?
 - Wie werden Software-Merkmale konsistent und objektiv gemessen?
 - Wie werden die Beziehungen zwischen verschiedenen Maßen hergestellt?
 - Wie werden Software-Maße zur Produktivitäts- und Qualitätsverbesserung genutzt?

Warum Messen in Softwareprojekten?

- Messungen helfen bei drei aufeinander aufbauenden Aufgaben
- **Verstehen**
 - Messung hilft zu verstehen, was während der Entwicklung eines Produkts passiert, indem sie Merkmale explizit herausstellt und quantifiziert.
- **Steuern**
 - Durch Messungen lassen sich Projekte steuern
 - Erfahrungsschatz aufbauen basierend auf früheren Projekten
 - Vorgehensweise (Methoden, Techniken, ...) mit Ergebnissen in Beziehung setzen
 - Wissen nutzen, um zukünftigen Projekte anzupassen, um Ziele zu erreichen
- **Verbessern**
 - Zusammenhänge zwischen Vorgehensweisen und Zielen verfestigen
 - Daraus allgemein akzeptierte Prinzipien entwickeln
 - Anwendung dieser Prinzipien bei zukünftigen Projekten von vorn herein planen

Was sind Software-Metriken?

- Software-Metriken sind Messungen bestimmter Merkmale von
 - Software-Produkten
 - Software-Projekten und
 - Software-Prozessen
- zu deren
 - Bewertung
 - Planung und
 - Überwachung



Maßtypen

- Produktmaße für Software-Systeme
 - Größe, Funktionalität, Komplexität, Bedienbarkeit, Zuverlässigkeit, Sicherheit, Wartbarkeit, Übertragbarkeit, Integrierbarkeit
- Projekt- bzw. Ressourcenmaße für Personen, Hardware, Software
 - Anzahl Entwickler, % Overhead, Preis, Leistungsrate, Speicherkapazität, Geschwindigkeit, Genauigkeit, Nutzen
- Prozessmaße für Software-Entwicklung und -Wartung
 - Dauer, Aufwände, Kosten, Fehlermeldungen, Änderungsanträge, Anforderungen, Anzahl Releases, Abstand zwischen Releases
- Statische Maße
messen eine Eigenschaft zu einem bestimmten Zeitpunkt
- Dynamische Maße
messen die Entwicklung der Eigenschaft im Laufe der Zeit

Metriken in der Qualitätssicherung

- Ziel: Quantitative Aussagen bezüglich des von Natur aus abstrakten Produktes Software
- Maßzahlen oder Metriken messen z.B. Qualitätsmerkmale
 - Gemessene Werte dahingehend beurteilen, ob sie den spezifizierten Anforderungen genügen
- Metriken liefern immer nur punktuelle Aussagen bezüglich des untersuchten Aspekts
 - Ermittelte Maßzahlen sind erst im Vergleich zu den Zahlen aus anderen untersuchten Programm(teil)en aussagekräftig
 - Interpretation der Maßzahlen nur in relativ stabilen, wiederholbaren Prozessen sinnvoll

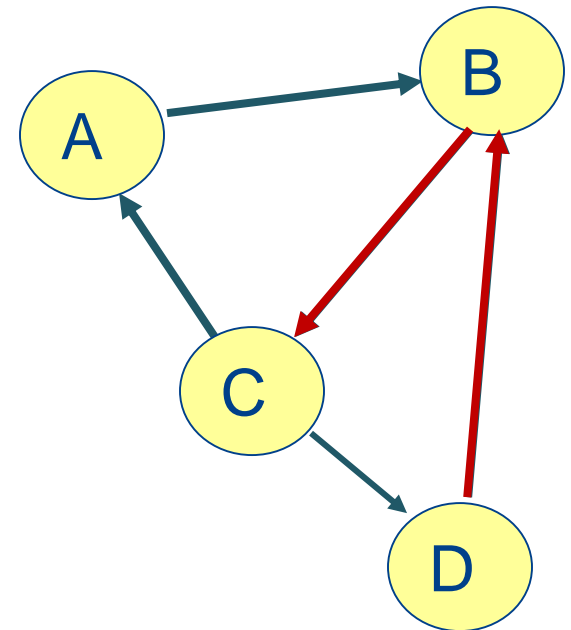


Graphentheorie: Die zyklomatische Zahl

- Frage: Anzahl zu entfernender Kanten eines stark zusammenhängenden Graphen, um einen Spannbaum zu erhalten
- $G = (V, E)$
 - $e = |E(G)|$, Anzahl der Kanten (edges)
 - $v = |V(G)|$, Anzahl der Knoten (vertices)
- Zyklomatische Zahl, *cyclomatic number* cn

$$cn(G) = e - v + 1$$

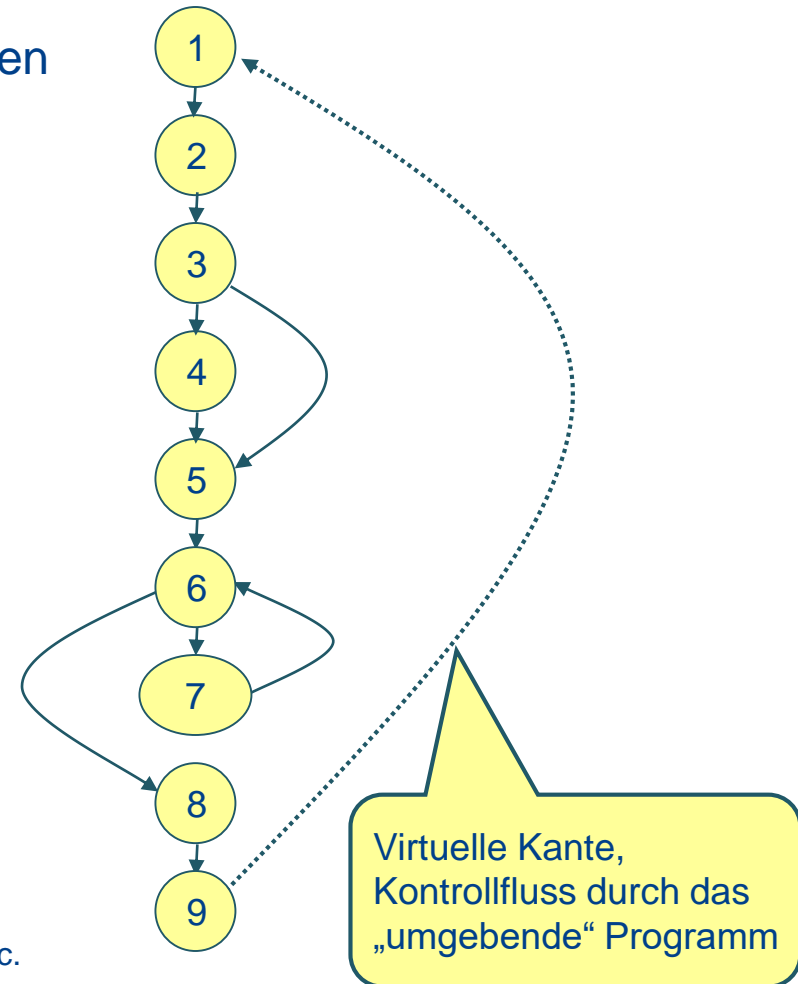
- Beispiel:
 $cn(G) = 5 - 4 + 1 = 2$
Entfernung der beiden Kanten B-C und D-B ergibt einen Spannbaum des Graphen!





Statische Analyse: Die zyklomatische Zahl

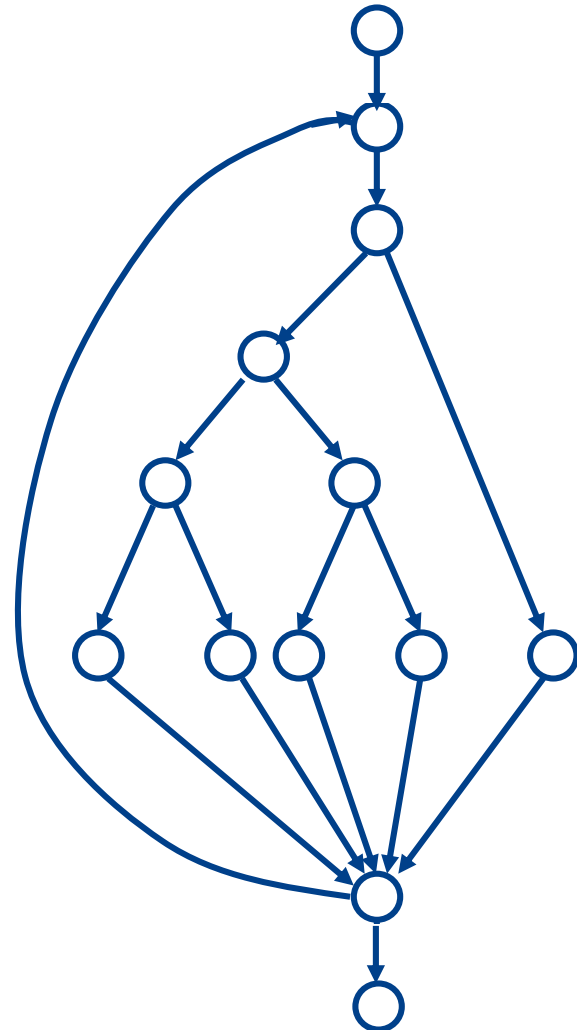
- Unabhängige Pfade: erzeugen in Linear-kombination alle Pfade durch den Graphen
- Frage: Anzahl der linear unabhängigen Pfade eines Kontrollflussgraphen
- $G = (V, E)$
 - $e = |E(G)|$, Anzahl der Kanten
 - $v = |V(G)|$, Anzahl der Knoten
- $v(G) = e - v + 2$ (wg. virtueller Kante)
- Beispiel:
 $v(G) = 10 - 9 + 2 = 3$
- Im Test-Jargon: „McCabe-Metrik“
Arthur H. Watson, Thomas J. McCabe: Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric.
NIST Special Publication No. 500-235, 1996





Beispiel: Zyklomatische Zahl

- Gesucht: $v(G) = e - v + 2$
mit
 - v = Anzahl der Knoten (vertices)
 - e = Anzahl der Kanten (edges)
- Lösung:
 $v = 13, e = 17$
 $v(G) = e - v + 2 = 17 - 13 + 2 = 6$
- $v(G) = e - v + 2$ ist auch
die Anzahl der Entscheidungen im
Kontrollflussgraph eines strukturierten
Programms +1,
hier: 5 (+ 1)





Bewertung der zyklomatischen Zahl

- Zyklomatische Zahl höher als 10 ist nach McCabe nicht tolerierbar
 - Überarbeitung des Programmteils!
 - Messwert 6 im Beispiel liegt im Bereich, den McCabe für akzeptabel hält
- Für die Wartbarkeit ist die Verständlichkeit eines Programmstücks von entscheidender Bedeutung
 - Je höher die ermittelte zyklomatische Zahl
 - desto schwieriger ist ein Nachvollziehen des Ablaufs des Programmstücks
 - desto schlechter ist damit die Verständlichkeit

Bewertung der zyklomatischen Zahl

- Zyklomatische Zahl gibt Auskunft über den Testaufwand
 - Zyklomatische Zahl = Anzahl der unabhängigen Pfade im Programmstück
 - Zyklomatische Zahl – 1 = Anzahl der Entscheidungen im Kontrollflussgraph eines strukturierten Programms
 - Oft 100%-ige Ausführung aller Anweisungen und Verzweigungsmöglichkeiten eines Programms verlangt
 - dafür könnten alle unabhängigen Pfade durch den Kontrollflussgraphen einmal durchlaufen werden
 - zyklomatische Zahl ist eine obere Grenze für Anzahl der benötigten Testfälle zur Erreichung dieses Kriteriums

Zusammenfassung Metriken

- **Verschiedene Arten** von Metriken wie
 - Produktmetriken, Projektmetriken, Prozessmetriken
 - Beispiele für Produktmetriken
 - Spezifikationsmetriken (bspw. Function Points)
 - Entwurfsmetriken (bspw. Modellbasierte Metriken)
 - Codemetriken (bspw. LoC)
 - Testmetriken (bspw. Überdeckungsgrade)
- Metriken wirken sich **nicht sofort** aus
- Metriken geben eine **Möglichkeit für Vergleiche**
- Metriken **liefern Abschätzungen** für Dauer, Kosten, etc.
- Metriken sind Instrumente die dem Projektmanagement **Entscheidungshilfen liefern**
- Typischerweise wird eine **Kombination von Metriken** passend auf den Entwicklungs- und Testprozess genutzt

Nutzen statischer Analysen

- Frühe Erkennung von Fehlerzuständen vor der Testdurchführung
- Frühe Warnung vor verdächtigen Aspekten in Code oder Design, durch Berechnung von Metriken wie z.B. hohes Komplexitätsmaß
- Identifizierung von Fehlerzuständen, die durch dynamischen Test nicht effektiv und effizient aufzudecken sind
- Aufdecken von Abhängigkeiten und Inkonsistenzen in Softwaremodellen, wie z.B. redundante oder die Architektur verletzende Links
- Verbesserte Lesbarkeit, Änderbarkeit und Wartbarkeit von Code und Design
- Vorbeugung von Fehlerzuständen, falls aus Erfahrung gelernt wird und sich dies in der Entwicklung niederschlägt

Mit statischer Analysen findbare Fehlerzustände

- Referenzierung einer Variablen mit nicht definiertem Wert
- Inkonsistente Schnittstellen zwischen Modulen und Komponenten
- Variablen, die nie verwendet werden
- Unerreichbarer (toter) Code (dead code)
- Verletzung von Programmierkonventionen
- Sicherheitsschwachstellen
- Syntax-Verletzungen von Code und Softwaremodellen



Zusammenfassung



- Statischer Test prüft ohne Ausführung des Testobjekts
 - kann bestimmte Fehlerzustände besser finden als dynamischer Test
 - wesentliche Arten: Statische Analyse und Review
- Grundprinzip Review: Mehrere Augenpaare sehen mehr als ein Augenpaar
- Der Reviewprozess umfasst Planung, Reviewbeginn, individuelles Review, Befundkommunikation und -analyse, Fehlerbehebung und Bericht
- Rollen im Review sind Management, Reviewleiter, Reviewmoderator, Autor, Reviewer und Protokollant
- Arten von Reviews sind informelles Review, Walkthrough, technisches Review und Inspektion
- Reviewverfahren sind Ad-hoc-Review, checklistenbasiertes Review, szenariobasiertes Review, perspektivisches Lesen, rollenbasiertes Review



Zusammenfassung Exkurs



- Werkzeuggestützte statische Analysen ohne Ausführung des Prüfobjektes sind für Dokumente mit gewissem Formalisierungsgrad möglich
- Compiler decken Fehlerzustände in der Syntax des Programmcodes auf, bieten aber meist noch weitere Möglichkeiten
- Analysewerkzeuge melden Verstöße gegen Standards und andere Konventionen
- Anomalien im Daten- und Kontrollfluss der Programmtexte weisen oft auf fehlerträchtige Stellen hin
- Metriken messen Eigenschaften von Produkten, Projekten und Prozessen. Um Aussagen über deren Qualität zu treffen, müssen sie aber interpretiert werden
- Die zyklomatische Zahl errechnet die Anzahl der linear unabhängigen Pfade im untersuchten Programmtext und gibt Hinweise zum Test- und Wartungsaufwand



Folgende Fragen sollten Sie jetzt beantworten können

- Welche typischen Arbeitsprodukte können einem Review unterzogen werden?
- Was sind die grundlegenden Schritte, die bei einem Review durchzuführen sind? Bitte beschreiben!
- Welche Reviewarten werden unterschieden?
- Welche Rollen wirken an einem technischen Review mit?
- Warum sind Reviews ein effizientes Mittel zur Qualitätssicherung?
- Was umfasst der Begriff „statische Analyse“? Bitte erklären!
- Wie stehen statische Analyse und Reviews in Zusammenhang?
- Warum kann die statische Analyse nicht alle in einem Programm enthaltenen Fehlerzustände aufdecken?
- Exkurs: Welche Arten von Datenflussanomalien werden unterschieden?
- Exkurs: Was sagt die zyklomatische Zahl über die Kontrollfluss-Komplexität eines Programms aus?
- Exkurs: Was sind typische Fehlerzustände im Quellcode und Entwurf, die durch eine werkzeuggestützte statische Analyse identifiziert werden können?



Muster-Prüfungsfragen

Testen Sie Ihr Wissen...



Frage 1

15. Welche Aktivitäten werden im Rahmen der Planung eines formalen Reviews durchgeführt? [K2]

a)	Sammeln von Metriken für die Bewertung der Effektivität des Reviews.	<input type="checkbox"/>
b)	Beantwortung von Fragen, die die Teilnehmer haben könnten.	<input type="checkbox"/>
c)	Definition und Prüfung von Eingangskriterien.	<input type="checkbox"/>
d)	Bewertung der Reviewbefunde gegenüber den Endekriterien.	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



Frage 1 – Lösung

15. Welche Aktivitäten werden im Rahmen der Planung eines formalen Reviews durchgeführt? [K2]

a)	Sammeln von Metriken für die Bewertung der Effektivität des Reviews.	<input type="checkbox"/>
b)	Beantwortung von Fragen, die die Teilnehmer haben könnten.	<input type="checkbox"/>
c)	Definition und Prüfung von Eingangskriterien.	<input checked="" type="checkbox"/>
d)	Bewertung der Reviewbefunde gegenüber den Endekriterien.	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



Frage 2

16. Welche der unten aufgeführten Reviewarten ist AM BESTEN geeignet, wenn das Review gemäß eines formalen Prozesses mit Regeln und unter Verwendung von Checklisten durchgeführt werden soll? [K2]

a)	Informelles Review	<input type="checkbox"/>
b)	Technisches Review	<input type="checkbox"/>
c)	Inspektion	<input type="checkbox"/>
d)	Walkthrough	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



Frage 2 – Lösung

16. Welche der unten aufgeführten Reviewarten ist AM BESTEN geeignet, wenn das Review gemäß eines formalen Prozesses mit Regeln und unter Verwendung von Checklisten durchgeführt werden soll? [K2]

a)	Informelles Review	<input type="checkbox"/>
b)	Technisches Review	<input type="checkbox"/>
c)	Inspektion	<input checked="" type="checkbox"/>
d)	Walkthrough	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



Frage 3

17. Welche ZWEI der folgenden Aussagen zu statischem Test sind am EHESTEN zutreffend? [K2]

a)	Statischer Test ist eine kostengünstige Möglichkeit, Fehlerzustände zu erkennen und zu beheben.	<input type="checkbox"/>
b)	Statischer Test macht den dynamischen Test weniger herausfordernd.	<input type="checkbox"/>
c)	Statischer Test erlaubt eine frühzeitige Validierung der Benutzeranforderungen.	<input type="checkbox"/>
d)	Statischer Test ermöglicht, Laufzeitprobleme frühzeitig im Lebenszyklus zu erkennen.	<input type="checkbox"/>
e)	Bei der Prüfung sicherheitskritischer Systeme hat der statische Test einen geringen Stellenwert, da der dynamische Test den Fehlerzustand besser findet.	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



Frage 3 – Lösung

17. Welche ZWEI der folgenden Aussagen zu statischem Test sind am EHESTEN zutreffend? [K2]

a)	Statischer Test ist eine kostengünstige Möglichkeit, Fehlerzustände zu erkennen und zu beheben.	<input checked="" type="checkbox"/>
b)	Statischer Test macht den dynamischen Test weniger herausfordernd.	<input type="checkbox"/>
c)	Statischer Test erlaubt eine frühzeitige Validierung der Benutzeranforderungen.	<input checked="" type="checkbox"/>
d)	Statischer Test ermöglicht, Laufzeitprobleme frühzeitig im Lebenszyklus zu erkennen.	<input type="checkbox"/>
e)	Bei der Prüfung sicherheitskritischer Systeme hat der statische Test einen geringen Stellenwert, da der dynamische Test den Fehlerzustand besser findet.	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



Frage 4

19. Was ist checklistenbasiertes Testen? [K1]

a)	Ein Testverfahren, bei dem Testfälle auf Basis des Wissens der Tester über frühere Fehler oder aus allgemeinem Wissen über Fehlerwirkungen abgeleitet werden.	<input type="checkbox"/>
b)	Ein Verfahren zur Herleitung und/oder Auswahl von Testfällen, das auf einer Analyse der funktionalen oder nicht-funktionalen Spezifikation einer Komponente oder eines Systems basiert, ohne Berücksichtigung ihrer internen Struktur.	<input type="checkbox"/>
c)	Ein erfahrungsbasiertes Testverfahren, bei dem der erfahrene Tester entweder eine Liste von Punkten nutzt, welche beachtet, überprüft oder in Erinnerung gerufen werden müssen, oder eine Menge von Regeln oder Kriterien nutzt, gegen welche ein Produkt verifiziert werden muss.	<input type="checkbox"/>
d)	Ein Testansatz, bei dem die Tester dynamisch Tests entwerfen und durchführen, basierend auf ihrem Wissen, der Erkundung des Testelements und dem Ergebnis früherer Tests.	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019



Frage 4 – Lösung

19. Was ist checklistenbasiertes Testen? [K1]

a)	Ein Testverfahren, bei dem Testfälle auf Basis des Wissens der Tester über frühere Fehler oder aus allgemeinem Wissen über Fehlerwirkungen abgeleitet werden.	<input type="checkbox"/>
b)	Ein Verfahren zur Herleitung und/oder Auswahl von Testfällen, das auf einer Analyse der funktionalen oder nicht-funktionalen Spezifikation einer Komponente oder eines Systems basiert, ohne Berücksichtigung ihrer internen Struktur.	<input type="checkbox"/>
c)	Ein erfahrungsbasiertes Testverfahren, bei dem der erfahrene Tester entweder eine Liste von Punkten nutzt, welche beachtet, überprüft oder in Erinnerung gerufen werden müssen, oder eine Menge von Regeln oder Kriterien nutzt, gegen welche ein Produkt verifiziert werden muss.	<input checked="" type="checkbox"/>
d)	Ein Testansatz, bei dem die Tester dynamisch Tests entwerfen und durchführen, basierend auf ihrem Wissen, der Erkundung des Testelements und dem Ergebnis früherer Tests.	<input type="checkbox"/>

Quelle: ISTQB Foundation Level Sample Paper; SET A; 2018; Deutschsprachige Fassung/ Lokalisierung; German Testing Board; 16.02.2019