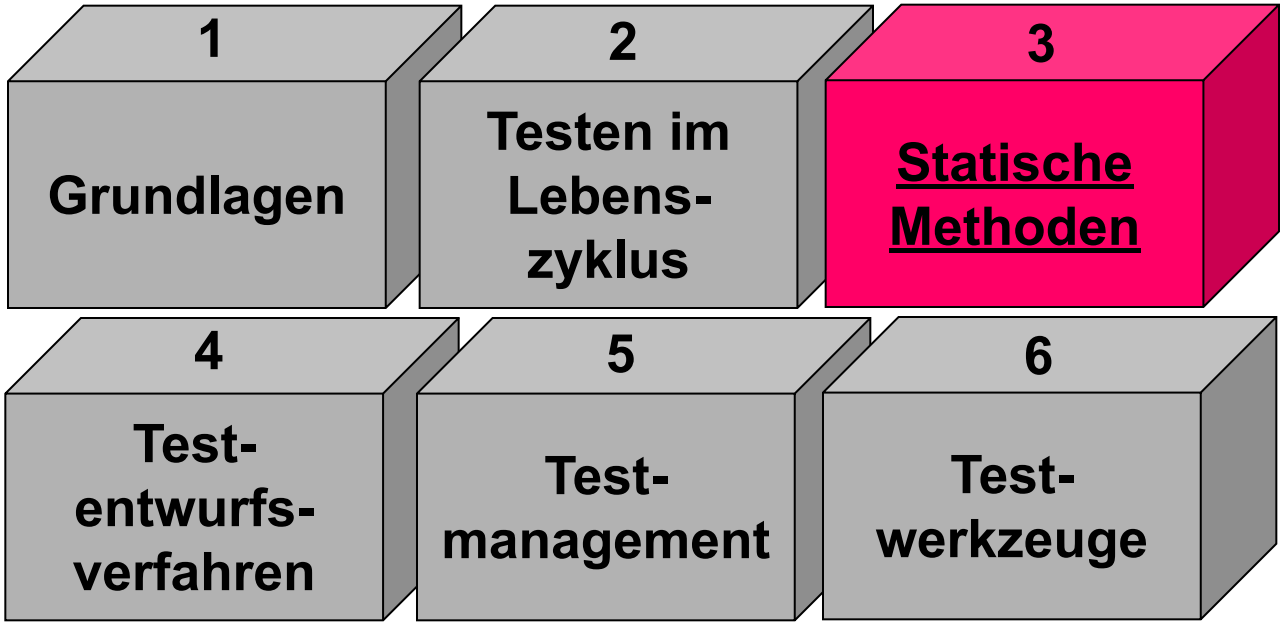
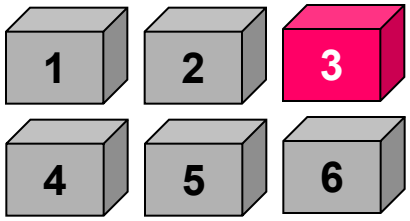




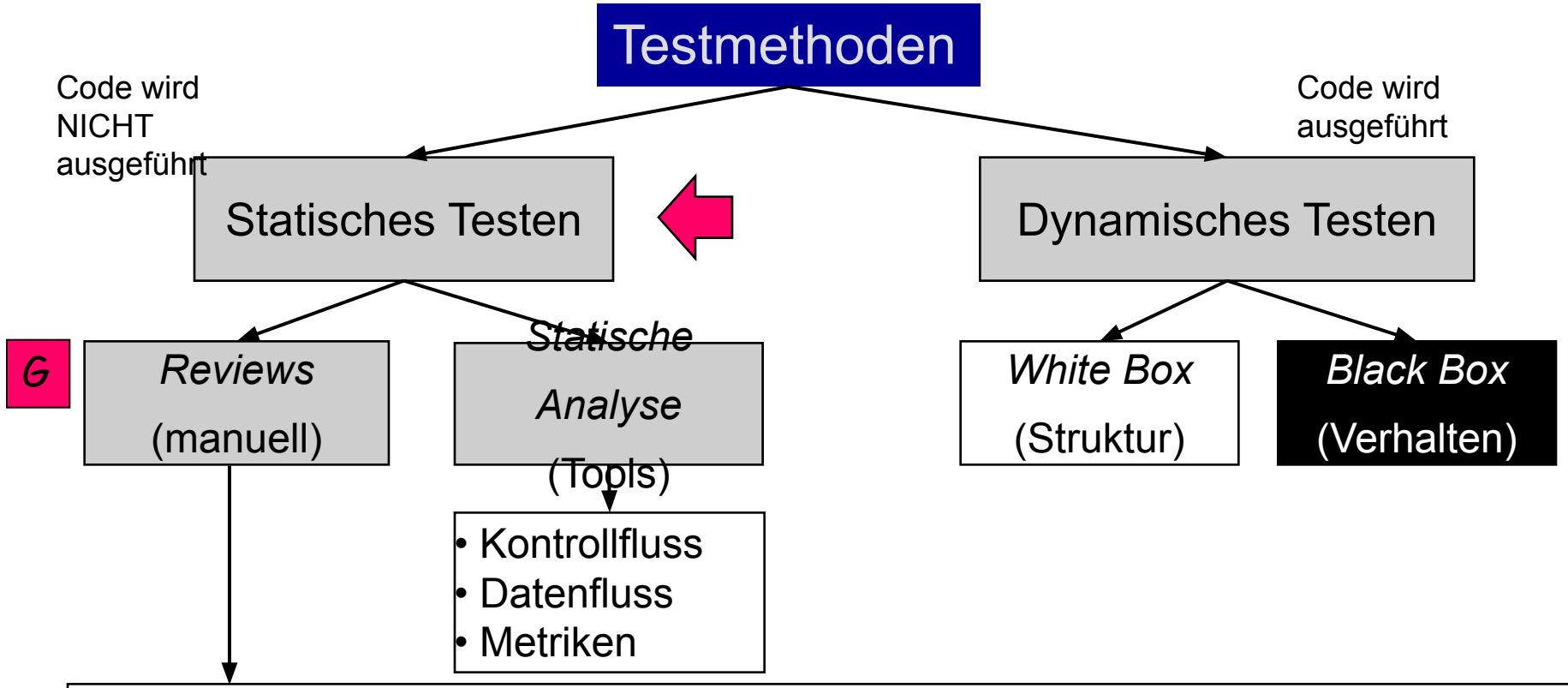
Statische Methoden

Lernzielstufe

- K2 — *Statische Prüftechniken* und der Testprozess
- K2 — Reviewprozess
- K2 — Werkzeuggestützte statische Analyse

Sehe Lehrplan, Kapitel 3 für die Lernzielbeschreibung dieses Moduls
Siehe Modul 1, Anhang A, Beschreibung der Lernzielstufen

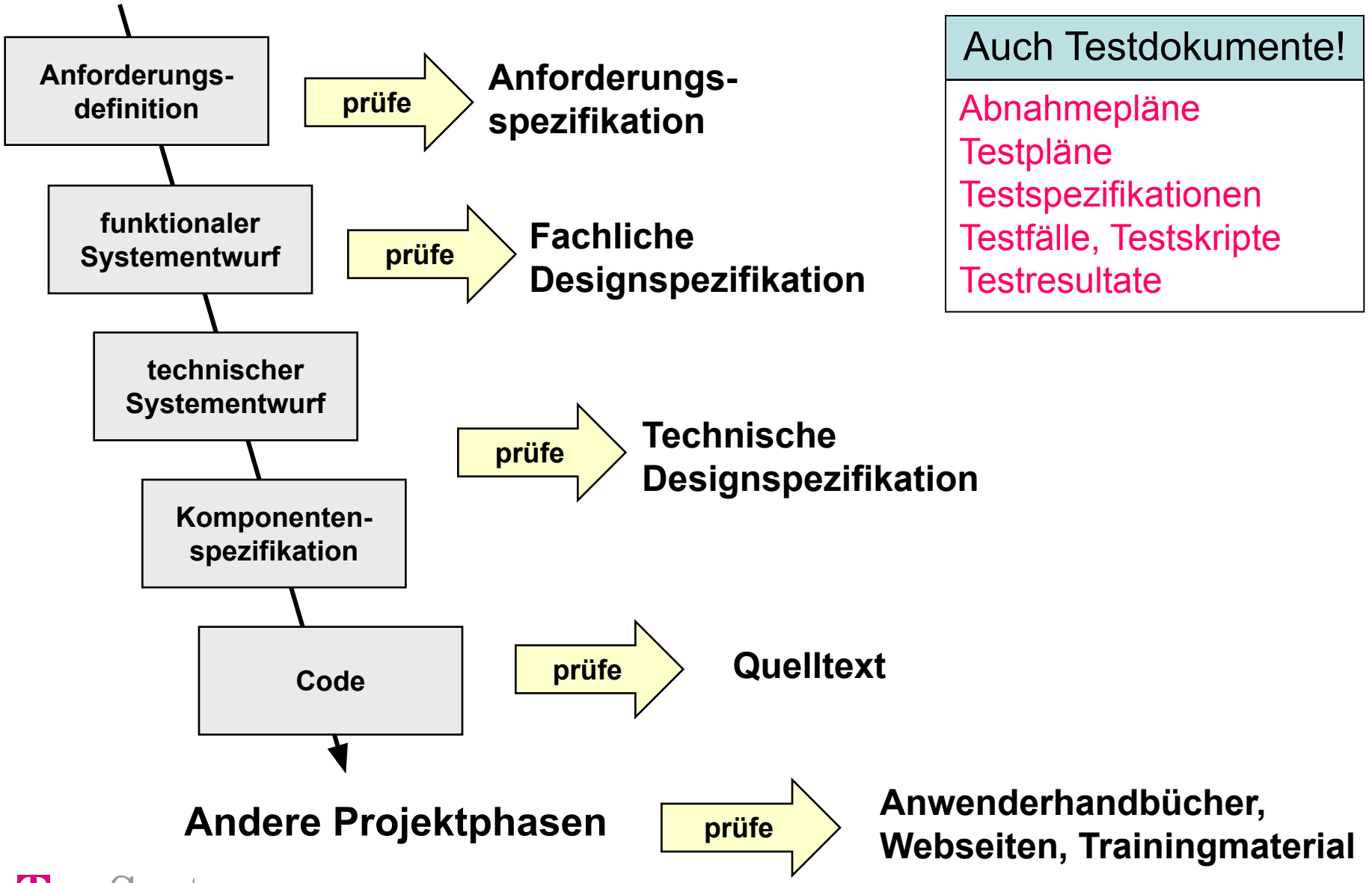
Statischer Test



- Durch Einzelne: Schreibtischtest
  Korrekturlesen
  Data Stepping
- Durch Gruppen: *Reviews*
  Walkthroughs
  Inspektionen

G Eine Bewertung eines Produkts oder eines Projektstatus. Sie dient dazu, Diskrepanzen zu den geplanten Ergeb-nissen aufzudecken und Verbes-serungspotenziale zu identifizieren.

Was wird statisch geprüft ?



Auch Testdokumente!

- Abnahmepläne
- Testpläne
- Testspezifikationen
- Testfälle, Testskripte
- Testresultate

Weitere Reviewkandidaten

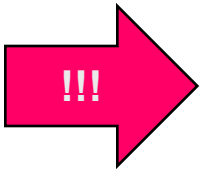
- Strategische Dokumente, z.B.
  - Businesspläne
  - Marketingmaterial
- Standards & Prozeduren, z.B. für
  - Qualitätsmanagement
  - Tests
  - Sicherheit
- Richtlinien & Styleguides, z.B.
  - zur Programmierung
  - zu Nutzeroberflächen

Welche Fehlerzustände finden Reviews?

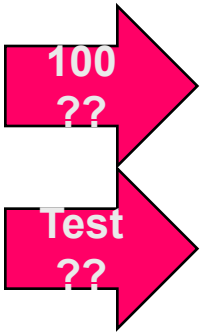
- Die spezifizierten Anforderungen sind
  - Unvollständig
  - Nicht testbar
  - Inkonsistent
- Designdokumente enthalten
  - Falsche Annahmen
  - Unvollständige Abdeckung der Anforderungen
  - Nicht realisierbares Design
  - Falsche Schnittstellenspezifikationen
- Code mit unzureichender Wartbarkeit
- Abweichungen von Standards und einzuhaltenden Richtlinien
- Prozessfehler, die weitere Fehlerzustände verursachen

Was finden Code Reviews?

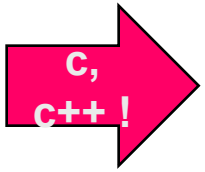
- Seltsame oder unlogische Funktionalität, die bei dynamischen Tests selten auffällt
- Wartungsprobleme, z.B.
  - Unerwünschter Testcode
  - „Magische“ Zahlen
  - Schlechte Kommentierung
- Logisch falsche Konstrukte




```
GetAccount (AccNumber)
If account in credit (AccNumber)
    add to account (AccNumber ,sum)
else
    issue letter (AccNumber)
    delete account (AccNumber+1)
endif
```



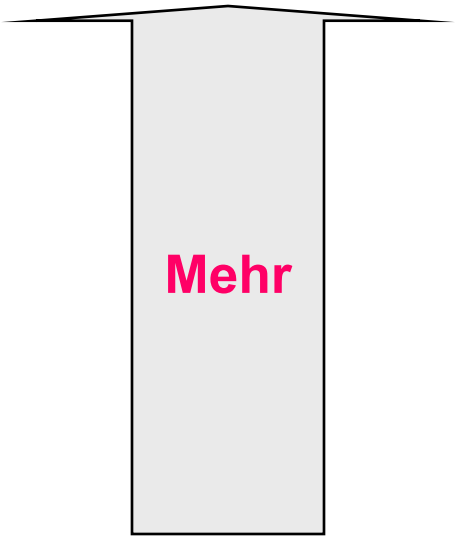

```
If current_account > 100
    add to account (AccNumber ,sum)
else
    issue letter (AccNumber)
    If Test then
       delete account (AccNumber+1)
endif
```




```
Sum = GetCurrentAccount (AccNumber)
if (Sum = 0)
    issue letter (AccNumber)

Print (Sum)
```

Vorteile statischer Methoden

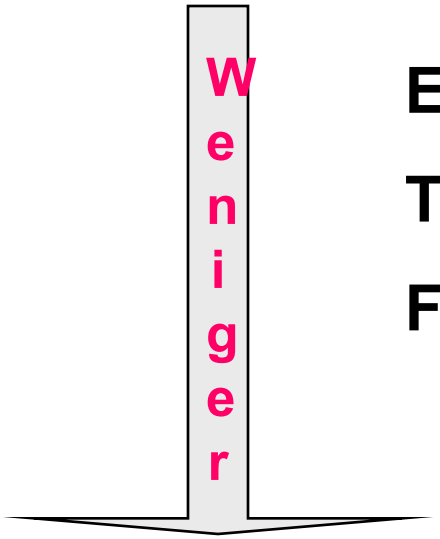


Zuverlässigkeit

Kundenvertrauen

Produktivität der Entwicklung

Vorbeugende Qualitätssicherung

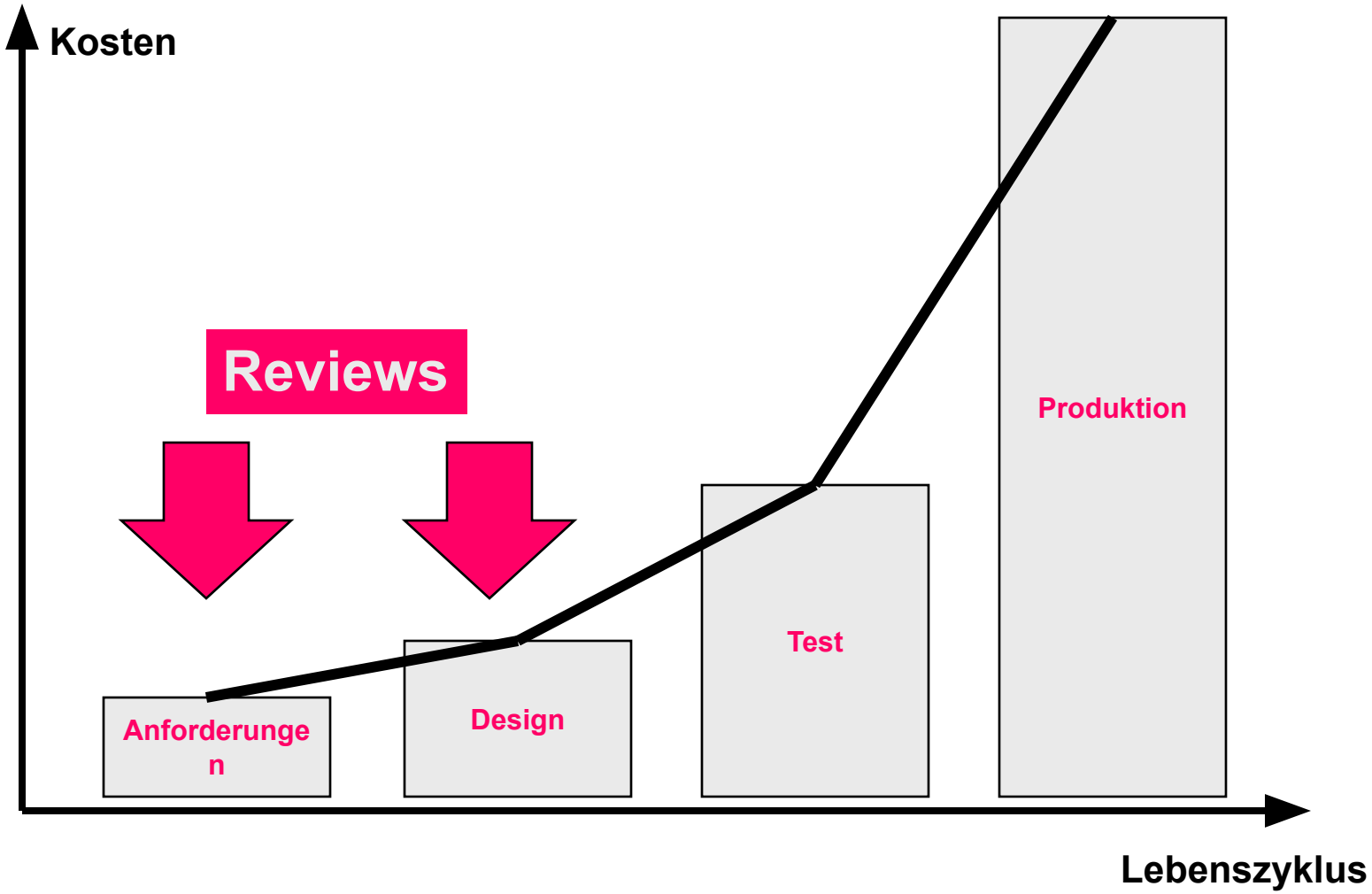


Entwicklungszeit

Testzeit und Gesamtkosten

Fehlerzustände in späteren Testphasen

Frühe Korrektur ist billiger



Kosteneffizienz – ein Beispiel

Projekt R&I bei Ericsson Telecom

Reviews & Inspections (R&I) für 4.000 Personen an 40 Standorten

Zentrales R&I Competence Team aus 7 Personen leitet Einführung

Lokale R&I „Champions“ an den einzelnen Standorten

„Buy-in“ durch Roadshows, Trainingspakete und lokale Steuerung der Prozesse.

Quelle:
Robert MacFarland, EuroSTAR 1998

Kosteneffizienz – ein Beispiel

Bilanz R&I bei Ericsson Telecom

Geschätzte Kosten: **450.000 Pfund**

Geschätzte Einsparungen: **1.800.000 Pfund**

Return on investment (ROI) **4:1 (Schwankung 3:1 < > 6:1)**

Geschätzte Zeitersparnis: 30.000 Personenstunden
10% der Arbeitsstunden

Ein 6-monatiges Projekt bräuchte ohne Inspektionen fast 3 Wochen länger !

Quelle:
Robert MacFarland, EuroSTAR 1998

Weitere Erfahrungswerte

Quelle : Software Inspections, Tom Gilb & Dorothy Graham

Kosteneffizienz

- Zeitrahmen um 25% reduziert
- Über 80% der Fehlerzustände in jedem Stadium beseitigt
- Wartungskosten um Faktor 28 reduziert

Kosten von Reviews

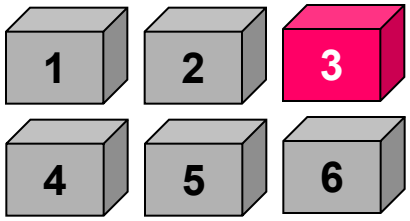
- Etwa 5 bis 15% der Entwicklungskosten abhängig von
  - Anzahl der Dokumente im Review
  - Tiefe und Art des Reviews
  - Erfahrung mit der Durchführung

Statischer vs. dynamischer Test

- Gemeinsames Ziel: Fehlerzustände finden
- Statische und dynamische Techniken sind komplementär, da sie auf verschiedene Fehlerarten abzielen
- Statische Techniken finden primär *Fehlerzustände*
- Dynamische Techniken finden primär *Fehlerwirkungen*
- Statische Techniken brauchen keinen lauffähigen Code
  - Früherkennung / Vermeidung von Fehlerzuständen

Definitionen:

- G *Dynamischer Test*: Prüfung des Testobjekts durch Ausführung auf einem Rechner.
- G *Statische Analyse:* Die Durchführung der Analyse eines Artefakts (z.B. Anforderung oder Quelltext) ohne Ausführung der Software.



Statische Methoden

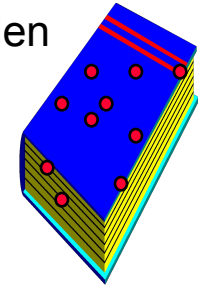
- Statische Prüftechniken und der Testprozess
- **Reviewprozess**
- Werkzeuggestützte statische Analyse

Reviews im Allgemeinen

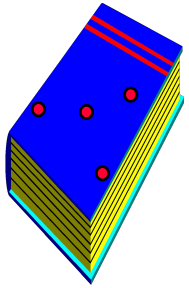
- G Review: Eine Reviewtechnik, welche durch ein dokumentiertes Vorgehen und Anforderungen charakterisiert ist
- G Reviewprozess: Planung, Vorbereitung, Durchführung und Nachbereitung eines Reviews [nach IEEE 610]
- G Peer Review: Ein Review eines Arbeitsergebnisses durch Kollegen des Erstellers mit dem Ziel, Fehlerzustände aufzudecken und Verbesserungsvorschläge zu identifizieren. Beispiele sind Inspektion, technisches Review und Walkthrough.
- g Reviews können sehr informell oder sehr formal (strukturiert und geregelt) sein. Der Grad des Formalismus ist abhängig von
  - g Reife des Entwicklungsprozesses
  - g Regulatorischen Anforderungen
  - g Bedarf an einem Prüfnachweis

Formale Reviews – der Prozess

Dokument mit Fehlerzustän den

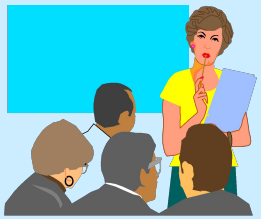


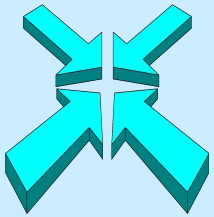
weniger Fehlerzustände



Planun g → **Kontrolle**

Kick-Off → **Individuelle Vorbereitung** → **Review-sit zung** → **Über-arbe itung** → **Nach-bere itung**





Beschreibung der Reviewphasen (K1)

- Planung & Kontrolle
  - Review- und Prüftkriterien festlegen
  - Benötigte Teilnehmer identifizieren und ihnen eine Reviewrolle zuordnen
    - Welche Spezialisten werden als Gutachter benötigt?
    - Wer soll die Rolle des Moderators übernehmen?
  - Eingangsdokument(e) festlegen (z.B. Anforderungsspezifikation)
  - Nach dem Kick-Off stellt der Moderator sicher, dass die vereinbarte Planung eingehalten wird und dass die Reviewsitzung stattfinden kann.
- Kick-Off
  - Planung und Reviewprozess abstimmen
  - „Commitment“ der Teilnehmer einholen (vor allem der Gutachter)
  - Eingangsdokument(e) verteilen
- Individuelle Vorbereitung
  - Überprüfen des Dokuments und Notieren von Fehlerzuständen

Bemerkung: Besonderheiten für Inspektionen (formale Reviewart) werden auf Seite 23 gelistet

Beschreibung der Reviewphasen (K1)

- Prüfen/Bewerten/Festhalten der Ergebnisse (Reviewsitzung)
  - Diskussion über individuelle Bemerkungen (vom Moderator geleitet)
  - Festhalner der Fehlerzustände
  - Empfehlungen und Entscheidungen treffen
  - Erstellen eines Protokolls (optional)
- Überarbeitung
  - Behebung der Fehlerzustände und Fragen beantworten (meist durch den Autor)
- Nachbereitung
  - Prüfung der Überarbeitung (meist durch den Moderator)
  - Freigabe des Dokuments
  - Sammeln von Metriken

Bemerkung: Besonderheiten für Inspektionen (formale Reviewart) werden auf Seite 23 gelistet

Reviews – die Rollen (K1)

Moderator (Leiter)
- Planung
- Coaching
- Nachbereitung
- Auswahl der Teilnehmer
- Leitung der Meetings
- Metriken

Autor
- Hauptverantwortlich für das zu prüfende Dokument
- Meist „passive“ Teilnahme an Meetings
- Notwendige Dokumentenanpassungen

Gutachter (Reviewer, Inspektor)
- Person mit notwendigem technischem oder fachlichem Hintergrund, um Befunde identifizieren zu können
- Einzelprüfung des Dokuments
- Vorbereitung der Meetings
- „Aktive“ Teilnahme an Meetings

Manager
- Entscheidung über Reviewgegenstand
- Ansetzen des Reviews
- Zeit im Projektplan zur Verfügung stellen
- Überprüfung der Zielerreichung

Protokollant (Protokollführer)
- Aufzeichnung der Meetingresultate
- Versorgt den Leiter mit Informationen zur effektiven Steuerung der Meetings

Aufwände für die Aktivitäten

Planung, Organisation	Moderator	5%
Indiv. Vorbereitung	Gutachter	65%
Reviewsitzung	Alle	10%
Korrektur, Anpassung	Autor, Moderator	10%
Metriken	Moderator	5%
Prozessverbesserung	Moderator	5%

Quelle : Software Inspections,
Tom Gilb & Dorothy Graham

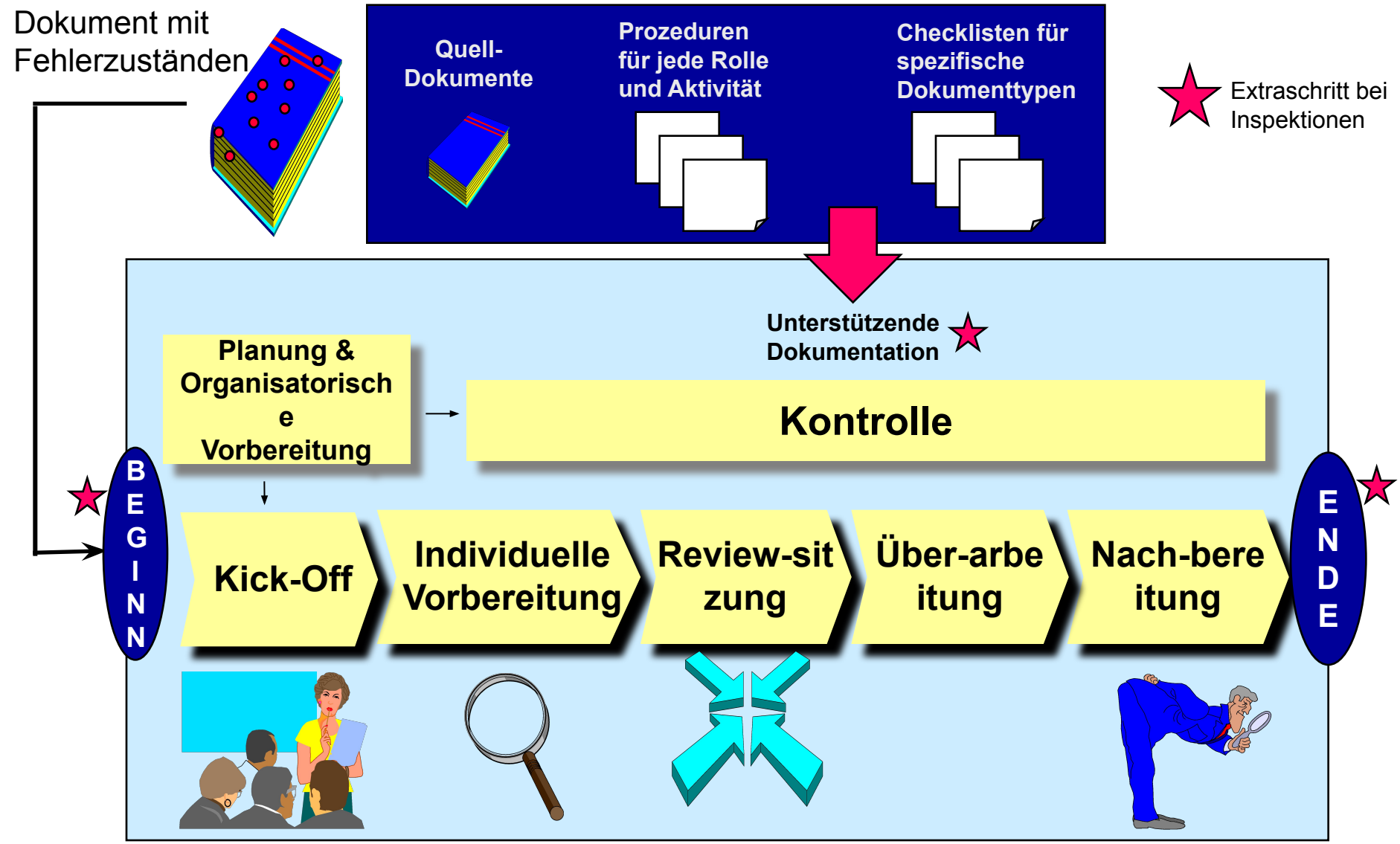
Reviewarten*

(*Siehe auch IEEE 1028
und die formalen Definitionen, Seite 44)

Alle Reviewarten können als *Peer Reviews* gestaltet werden

Reviewart	Beschreibung
Inspektion	Formale Prüfung gewöhnlich durch gleichgestellte Personen nach vorgegebenen Regeln und Checklisten. Definierte Eingangs- und Endebedingungen. Formalisierte Nachverfolgung. Leiter ist speziell geschult. Primärziel Fehlerzustände finden
Technisches Review	Fehlersuche durch technisch orientierte Personen. Häufig als Peer Review ohne Management-Beteiligung. Große Bandbreite bezüglich der Formalität. Ergebnisse werden dokumentiert. Primärziele Diskussion, Bewertung, Fehler finden, Problemlösung, Entscheidung
Informelles Review	Kein formaler Prozess. Häufig praktiziert als billige aber kaum definierte Reviewmethode. Dokumentation optional. Nutzen variiert abhängig vom Gutachter (z.B. technischer Leiter oder Teamkollege). Primärziel Kostengünstiges Fehler finden
Walkthrough	Der Autor verliest ein Dokument und erläutert den gewöhnlich gleichgestellten Teilnehmern spezielle Inhalte, Annahmen oder Entscheidungen. Durchspielen von Szenarien. Oft sehr langwieriger Prozess. Große Bandbreite bezüglich der Formalität. Primärziele Know-How-Transfer, Konsensbildung, Fehler finden

Inspektion – der Prozess

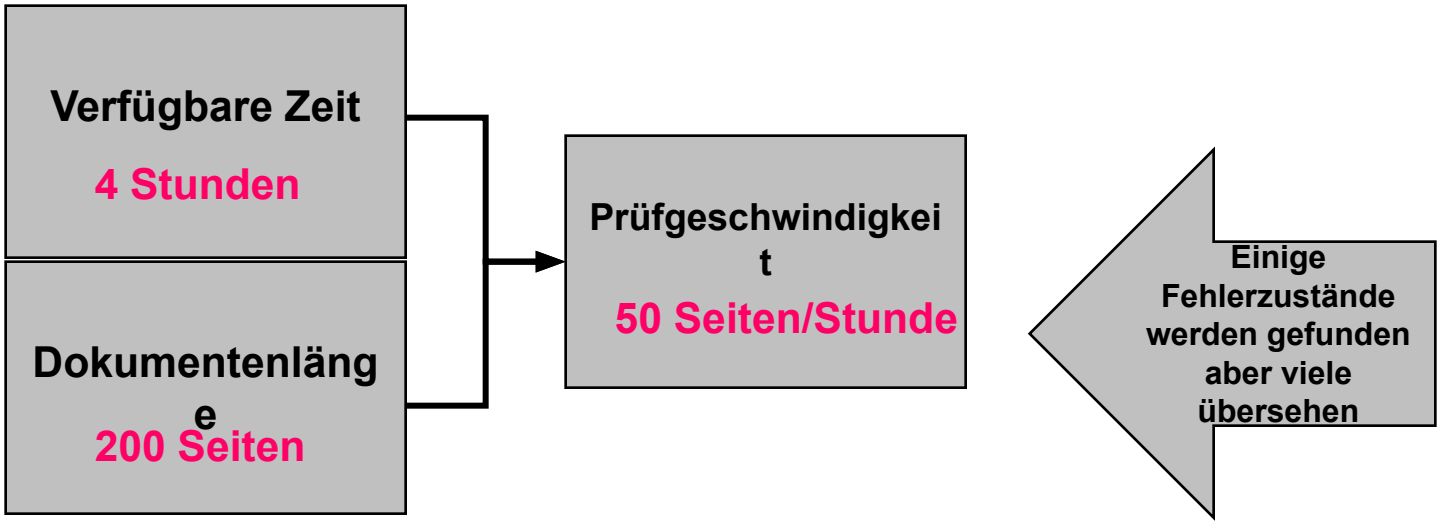


Besonderheiten der Inspektion

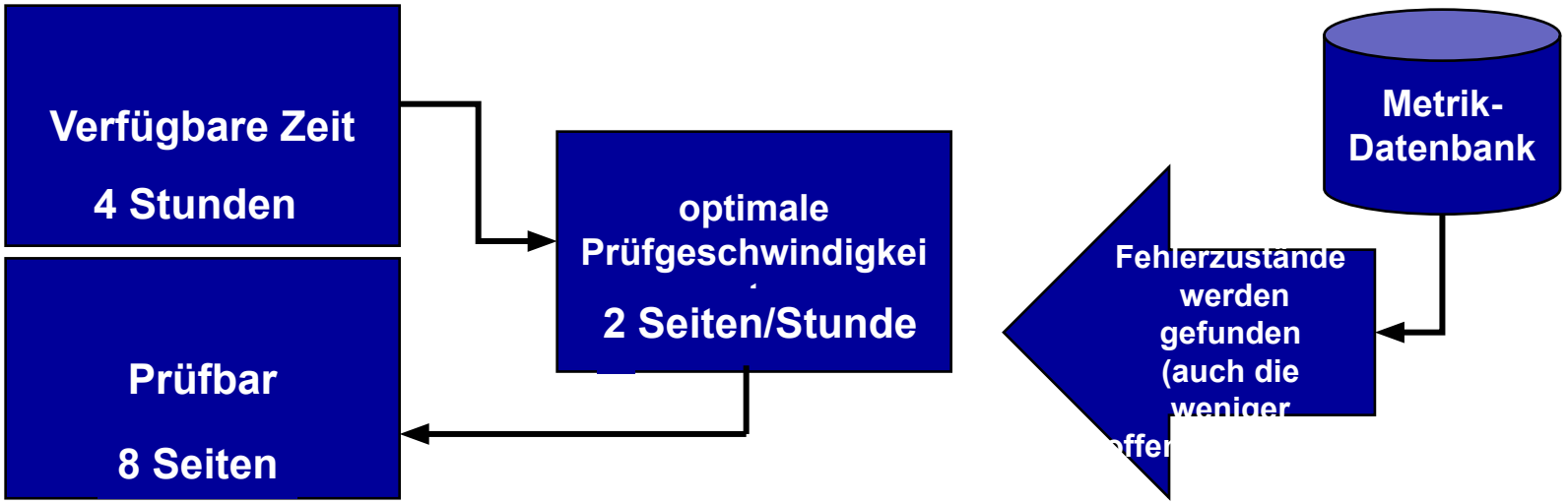
- Definierte *Eingangs*- und *Endbedingungen (*Prozess wird mit definierten Aufgaben und Kriterien begonnen bzw. beendet)
- Rollenspezifische Prozeduren unterstützen den Prozess
- Checklisten unterstützen die Prüfung und definieren schwere und geringfügige Fehler
- Hohe Effektivität durch eine relativ geringe, durch *Metriken* regulierte Prüfgeschwindigkeit
- Prozessverbesserung kann zu einem integralen Bestandteil des Inspektionsprozesses definiert werden
- Rollen und Verantwortlichkeiten erfordern spezielles Training
- Meetings laufen formaler ab mit weniger Diskussion und schnellerer Fehlererfassung. Autor ist nur „passiv“ dabei

Steuerung durch *Metriken*

Ansatz für *Technische Reviews:*



Ansatz für *Inspektionen*:



Inspektionen sind gründlicher

Aspekt	Technisches Review	Inspektion
Ansatz bei der Prüfung	„Bitte suchen Sie Probleme in diesem Dokument“	„Dies ist Deine Rolle, bitte prüfe das Dokument anhand dieser Prozedur nach dieser Checkliste“
Umfang der Prüfung	In der Regel das gesamte Dokument	Teile des Dokuments, die speziell für den Prüfer ausgewählt werden
Prüfgegenstand	Ein spezielles Dokument	Das Dokument selbst und alle bei der Erstellung verwendeten Dokumente
Umgang mit Fehlern	„Dies hier sieht irgendwie nicht richtig aus“	„Verstoß gegen Regel N aus Checkliste ABC“
Durchführung des Meetings	Wenig kontrolliert mit viel Diskussionen und häufig mit Zeitüberschreitung. Autor ist an Debatten über Lösungen und Rechtfertigungen beteiligt.	Strikte Moderation, auf maximal 2 Stunden begrenzt, kaum Diskussion, sehr fokussiert. Meeting findet nicht statt, bevor die Eingangskriterien erfüllt sind. Autor bleibt passiv.
Prüfmethode	Der Prüfer liest das Dokument relativ schnell durch und macht sich Notizen	Der Prüfer wertet das Dokument anhand von Richtlinien und Checklisten methodisch aus

Effektivität und Effizienz

	Techn. Review	Inspektion	
		Einführung	Ausgereift
Effektivität*	10 - 20%	30 - 40%	80 - 95%
Return on Investment	variabel	6 - 8	8 - 30

*** Effektivität :**
Aufdeckungsquote von Fehlerzuständen

Quelle : Software Inspections, Tom Gilb & Dorothy Graham

Reviewarten im Vergleich

Reviewtyp	Moderator	Team	Vorbereit.	Metrik	Resultat	Check-listen	Vorteile	Nachteile
Inspektion	Speziell geschult (nicht Autor)	3 - 6	Ja	Ja	Im Protokoll	Ja	Effektiv	Anfangs-investitionen
Technisches Review	Beliebig	3 - 6	Ja	Opt.	Im Protokoll	Opt.	Geringer Aufwand, findet Fehler	Subjektiv
Informelles Review	Beliebig	3 - 10	Ja	Opt.	Opt.	Nein	Geringer Aufwand	Uneffektiv, trügerische Sicherheit
Walkthrough	Autor	Viele Teil-nehmer	Opt.	Opt.	Opt.	Nein	Einführung für großen Teilnehmerkreis	Relativ aufwendig

Opt. = optional

Quelle : Software Inspections, Tom Gilb & Dorothy Graham

Generelle Zielsetzung von Reviews

- Bewertung und Prüfung von Artefakten (Dokumente, Spezifikationen, Code, usw.) anhand von Anforderungen und Spezifikationen
- Bewertung und Prüfung von Dokumenten anhand von Standards für alle Produkte des Unternehmens
- Früherkennung von Fehlerzuständen
- Aufbau von Vertrauen in das Projekt, noch bevor der Code entsteht
- Verbesserung des Prozesses, der zur Entstehung des geprüften Dokuments führte

Mit Ausnahme der Inspektionen...

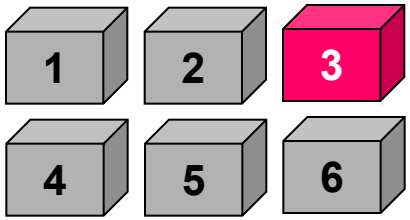
- Konsensbildung über ein breites Spektrum an Themen im Projekt

Kritische Erfolgsfaktoren 1/2

- Verständnis der Rollen und Verantwortlichkeiten
- Management-Unterstützung
- Angemessenes Training in den nötigen Methoden
- Einplanung benötigter Zeit und Ressourcen für Reviews
- Lernfähigkeit: Neben reiner Fehlerkorrektur werden auch die Fehlerursachen untersucht und bekämpft
- Identifizierte Prozessverbesserungen werden angegangen
- Überwindung von Widerständen gegen Formalien
- Der *Reviewprozess* wird durch Standards (z.B. Rollen und Checklisten) sinnvoll unterstützt
- Klar definierte Ziele

Kritische Erfolgsfaktoren 2/2

- Einbindung der dafür geeigneten Personen, z.B. auch Tester
  - Bringen spezifisches Know-How ein
  - Werden früh mit dem Produkt vertraut
- Einsatz von Techniken ist angepasst an Teilnehmer und Reviewgegenstand
- Offener und positiver Umgang mit
  - Gefundenen Fehlern
  - Verbesserungsvorschlägen
  - Fragen
- Berücksichtigung psychologischer Aspekte
  - Gegenseitige Wertschätzung
  - Vermeidung von Vorwürfen und Rechtfertigungen



- Statische Prüftechniken und der Testprozess
- Reviewprozess
- **Werkzeuggestützte statische Analyse**

Statische Analyse

- Prüfung, dass bestimmte Standards umgesetzt wurden:
  - Richtlinien zur Programmierung
  - Standards zur Dokumentation
  - Designgrundsätze
- Früherkennung realer oder potenzieller Fehlerzustände, die vom Compiler nicht und mit dynamischen Testmethoden nur unter hohem Aufwand gefunden werden.
- Aufschlüsse über Design und Code, die einen wertvollen Beitrag zur Risikobewertung liefern.
- Methoden sind z.B. *Datenfluss*- und *Kontrollflussanalyse*
- Toolunterstützung notwendig, bspw. auch als automatische Kontrolle beim Einchecken in einem Konfigurationsmgmt.werkzeug

Typische Datenfluss-Fehler

- Typenkonflikte
- Undeklarierte/uninitialisierte Variablen
- Verletzung von Feldgrenzen
- Schlechter Stil, der als Fehlerzustand gewertet werden kann, z.B. implizite Typumwandlung

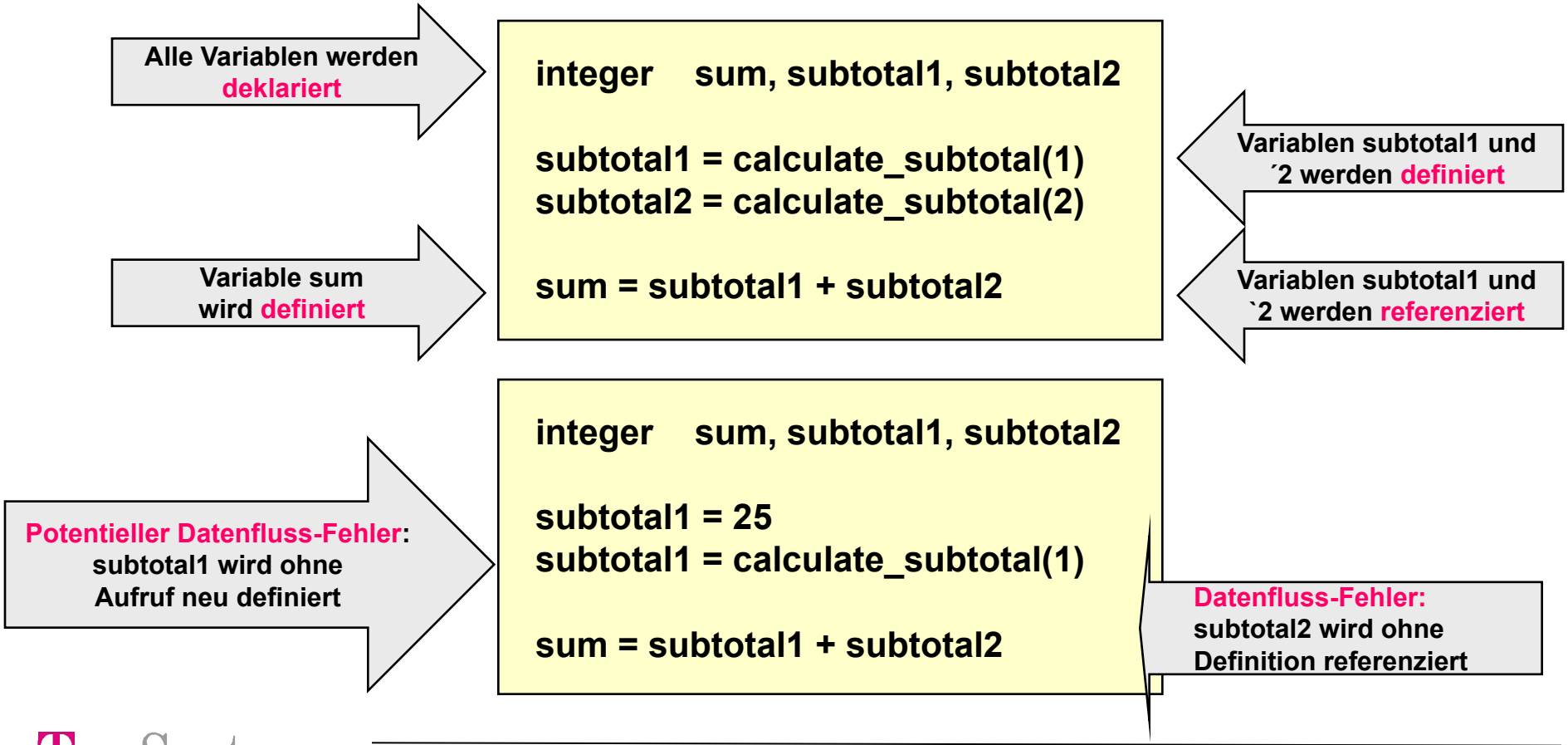
Solche Fehlerzustände werden häufig mit Tools ausgewertet

Zum Beispiel, Compiler können undeklarierte oder uninitialisierte Variablen entdecken

Datenflussanalyse

Deklaration, Definition und Referenzierung von Variablen

Eine Variable wird **deklariert,** wenn ihr ein Datentyp zugewiesen wird.
Eine Variable wird **definiert,** wenn ihr ein Wert zugewiesen wird.
Eine Variable wird **referenziert,** wenn dieser Wert verwendet wird.



Typische Kontrollfluss-Fehler

Fehlerzustände (oder Indikatoren für Fehler), die durch Kontrollflussanalyse aufgedeckt werden können:

- Nicht ausführbare Anweisungen („Toter Code“)
- Endlosschleifen
- Mehrere Eingänge oder Ausgänge für Schleifen
- Nicht definierte, nicht verwendete Sprungziele
- Schlechter Stil, z.B. komplexe Zeigerarithmetik
- Übertriebene *Komplexität* („Zyklomatische Komplexität“)
- Abweichungen von Styleguides zur Codestruktur

Kontrollflussanalyse

```
integer    sum, count, var1, subtotal1, subtotal2
integer    bonus = 10000

subtotal1 = abs(calculate_subtotal(1))
subtotal2 = abs(calculate_subtotal(2))

sum   = subtotal1 + subtotal2
count = int (subtotal1 / subtotal2)

if (sum < count) then
       count = 0
endif

do while count >= 0

       sum = sum + bonus
end do


count = count -1

write (sum)
```

Sonstiger Fehler: Nicht abgefangene Division durch Null

Kontrollflussfehler: Endlosschleife für count >= 0

Datenflussanomalie: count wird neu-definiert : Verwendung

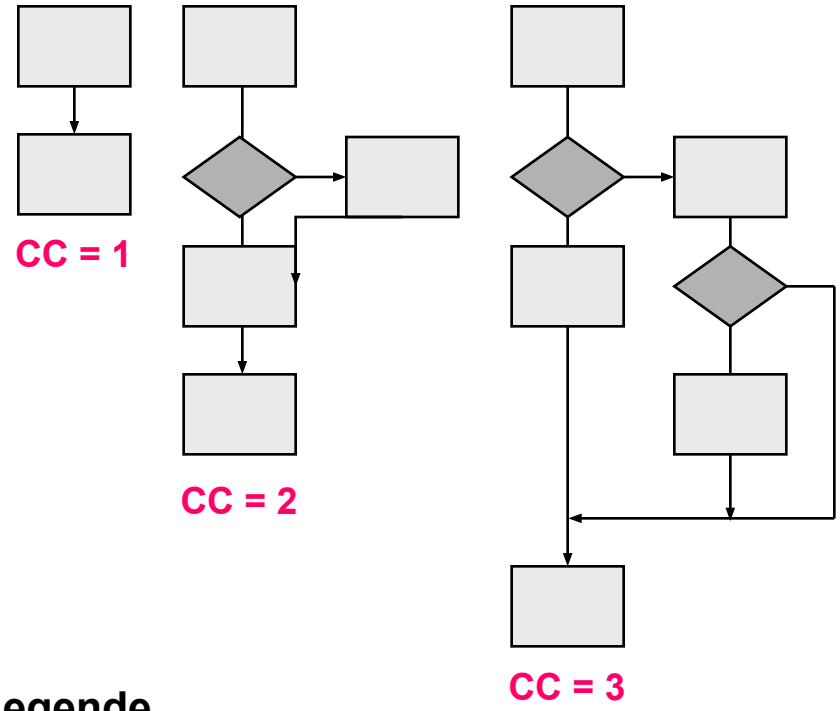
Kontrollflussfehler: Anweisung wird nie ausgeführt

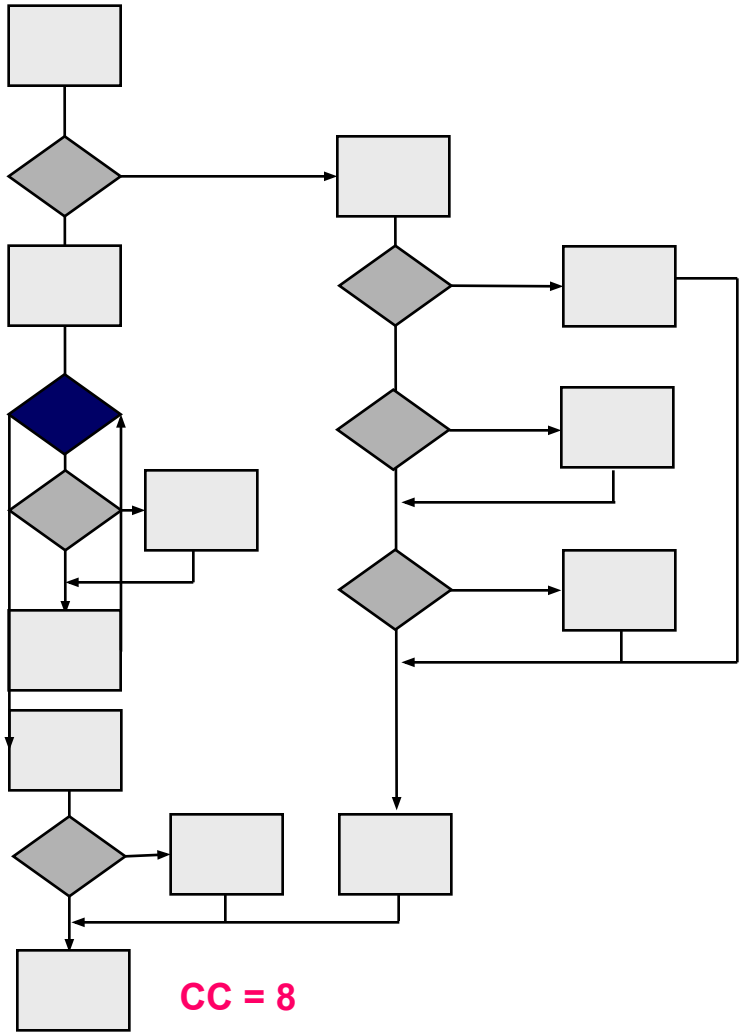
Sonstiger Fehler: Möglicher Integerüberlauf

Zyklomatische *Komplexität* (CC)

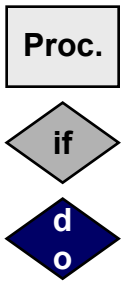
- Zyklomatische Komplexität (CC) misst die Komplexität des Kontrollflussgraphen
- Zyklomatische Zahl = Anzahl Verzweigungen + 1
- Anzahl unabhängiger Pfade
- Höhere CC bedeutet einen komplexeren Kontrollfluss
- Hohe *Komplexität* bedeutet häufig:
  - Code oder Design sind schwach strukturiert
  - Code oder Design sind fehlerträchtig
- CC kann verwendet werden als *Metrik* für die relative Wahrscheinlichkeit (d.h. das Risiko), dass ein vorliegendes Design oder Codestück Fehlerzustände enthält.

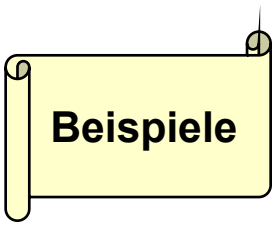
CC - Übung





Legende





Metriken der statischen Analyse

Design- oder Codequalität

- Kopplung
- Kohäsion

Datenflusskomplexität

- Operanden & Operatoren (Halstead)

Komplexität von OO-Systemen

- Tiefe des Vererbungsbaums
- Methoden in einer Klasse

Schnittstellenkomplexität

- Fan-in / Fan-out
- Funktionsaufrufe

Kontrollflusskomplexität

- Verschachtelungstiefe
- McCabe Metrik (CC)

Sonstige

- Lines of Code (LOC)

Viele Compiler und Entwicklungsumgebungen liefern solche Informationen

Die Rolle des Compilers

- Programmcode wird in Maschinencode „übersetzt“
- Analyse des Programmcodes („Syntaxprüfung“)
- Generierung von Informationen (nützlich in der Wartung):
  - Variablennutzung
  - Referenzen zwischen Variablen, deren Bezeichnern und Verwendung in Funktionen (Cross Reference Listen)
  - Datentypkonflikte
  - Speicherbelegung
- Generierung von *Metriken*

Eigenschaften statischer Analyse

Vorteile

- Findet Fehler, die kaum durch Inspektion und nur mit hohem Aufwand durch dynamische Tests aufgedeckt werden
- Wird durch Tools unterstützt
- Häufig früher durchführbar als dynamische Tests
- Auch auf Design anwendbar
- Objektive Aussagen über die Qualität von Design und Code

Nachteile

- Gefundene Fehler müssen auf Wichtigkeit interpretiert werden
- Output der Tools muss gefiltert werden, um die Informationen überschaubar zu halten
- Objektive Bewertung der Metriken ist entscheidend, um das „Na und?“ Problem zu vermeiden
- Fehler beim Betrieb des Systems (Timing, Speicherbelegung) werden nicht gefunden

Besondere Stärken

- Prüfung gegen Standards verbessert die Wartbarkeit von
  - Design
  - Code
- Früherkennung von Abhängigkeiten und Inkonsistenzen in
  - Softwaremodellen (z.B. Links)
  - Schnittstellen (von Modulen, Komponenten, Systemen)
- Fehlervermeidung (wenn neben individuellen Fehlern auch systematische Ursachen identifiziert und bekämpft werden)
- Aufdeckung von Sicherheitsschwächen

Zusammenfassung: Statische Methoden

- *Reviews* werden in frühen Projektphasen eingesetzt, um Fehlerzustände in der Dokumentation zu finden
- Es gibt verschiedene Reviewtypen: *Walkthrough, Technisches Review, Informelles Review, Inspektion* ...
- Ein Review kann als Prozess beschrieben werden
- Inspektionen sind formaler, aber auch effektiver bei der Fehlersuche
- Inspektionen verwenden Prozeduren und Checklisten
- Inspektionen sind kosteneffizient!
- *Statische Analysen* liefern Informationen über Qualität von Design bzw. Code und finden Fehlerzustände, ohne den Code auszuführen

Änderungsverzeichnis

Datum	Version	Autor	Bemerkung
22.02.2011	3.0	S.Massonet	Re-Akkreditierung zum LP 2010

Reviewarten definiert

Reviewart	Formale Definition nach G
Inspektion	Eine Reviewart, die Mängel durch die Sichtprüfung von Dokumenten finden soll. Solche Mängel können sein: Nichteinhaltung von Entwicklungsstandards, Nichtkonformität gegenüber zugrundeliegenden Dokumenten. Es ist die formalste Reviewtechnik und sie folgt deshalb einem dokumentierten Vorgehen [nach IEEE 610, IEEE 1028].
Technisches Review	Eine Diskussion in einer Gruppe gleichgestellter qualifizierter Mitarbeiter, die sich darauf konzentriert, eine Übereinstimmung über technische Vorgehensweisen zu erreichen [Gilb und Graham], [IEEE 1028].
Informelles Review	Review ohne festgelegten formalen (dokumentierten) Ablauf
Walkthrough	Eine schrittweise Präsentation eines Dokuments durch den Autor, um Informationen zu sammeln und ein gemeinsames Verständnis des Inhalts aufzubauen. [Freedman and Weinberg]