

Martin Glinz Harald Gall

Software Engineering

Kapitel 9

Reviews



Universität Zürich
Institut für Informatik

9.1 Grundlagen

9.2 Review-Formen

9.3 Durchführung eines Reviews

9.4 Review-Verfahren

9.5 Review-Aufwand

Terminologie

Review – Eine **formell organisierte** Zusammenkunft von **Personen** zur inhaltlichen oder formellen **Überprüfung** eines Produktteils (Dokument, Programmstück, etc.) nach vorgegebenen Prüfkriterien und -listen. Wird präziser auch als **technisches Review** bezeichnet.

Im Deutschen wird manchmal der Terminus **Durchsicht** gebraucht

- **Abgrenzung**: **keine Reviews** im hier betrachteten Sinn sind:
 - **informelle Prüfungen**, z.B. Durchlesen durch Kollegen
 - **Management-Reviews** zur Überprüfung von Kosten und Terminen
 - Sonderfall: **Selbst-Review**, d.h. eine Person überprüft ein eigenes Arbeitsergebnis mit den gleichen Verfahren, wie sie in formellen technischen Reviews zur Anwendung kommen
- Hier: nur **formelle technische Reviews (einschließlich Selbst-Review)**

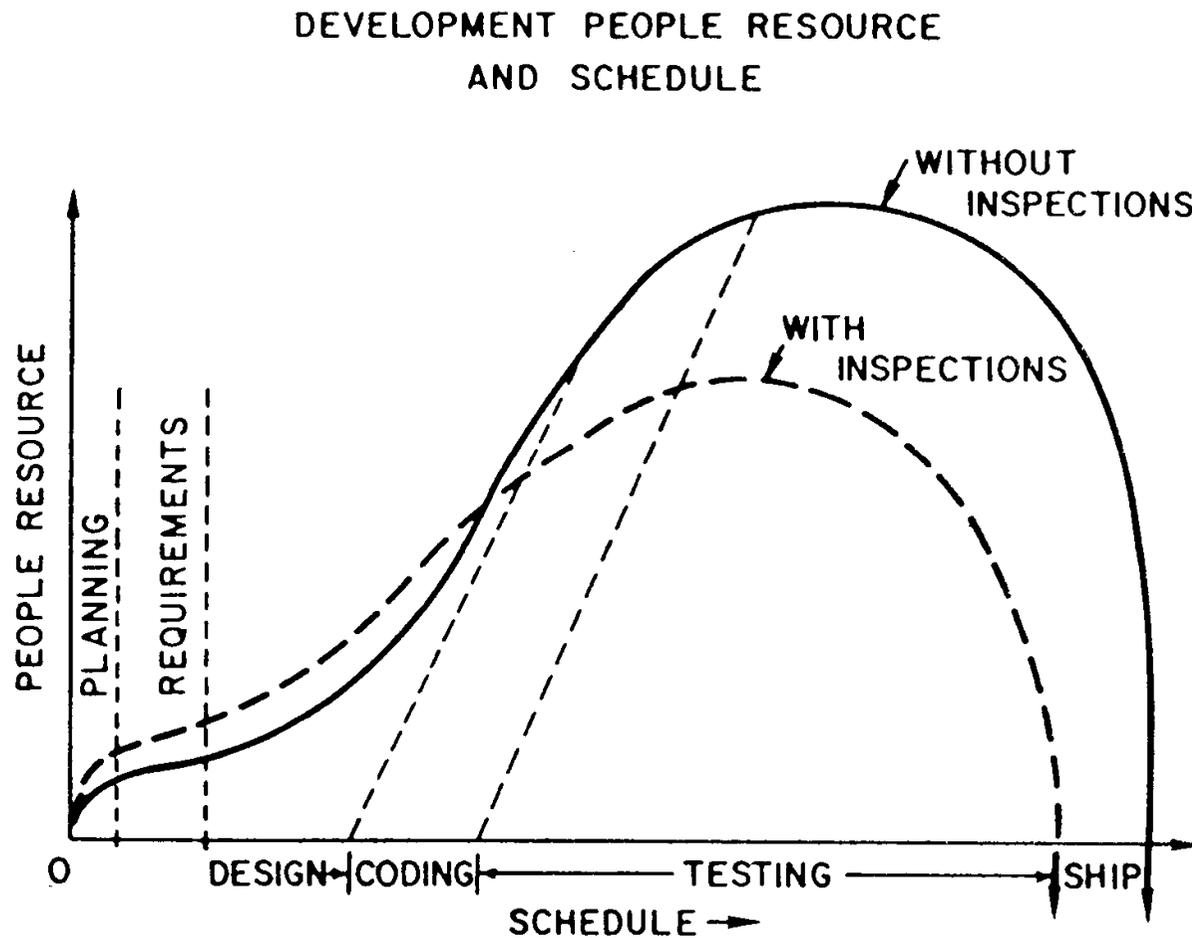
Einordnung

- Statisches Prüfverfahren
- Ziele
 - Aufzeigen der **Schwachstellen** und **Mängel** des Prüflings
 - Bestätigen der **Stärken** des Prüflings
 - Beurteilung der **Qualität**
- Ergebnis eines Reviews
 - Bericht mit einer Liste von Befunden

Warum Reviews?

- Fehler finden
 - Inspektionen finden die meisten Fehler
 - Reviewen ist billiger als testen:
 - weniger Vorbereitungs- und Durchführungsaufwand
 - Findet Fehlerursachen, nicht nur Symptome
 - Nicht jede Software-Einheit ist testbar, aber jede ist reviewbar
- Besser werden
 - Richtiges bestätigen / beibehalten / verstärken
 - Arbeitsweise vereinheitlichen
 - Ergebnisse auswerten ⇔ Prozesse / Qualität lenken
 - Aus Schwach- und Starkstellen lernen: Wissenstransfer von den guten zu den schlechten Software-Entwicklern
- Reviews sind wirtschaftlich

Wirtschaftlichkeit von Reviews



(Fagan, 1986)

9.1 Grundlagen

9.2 Review-Formen

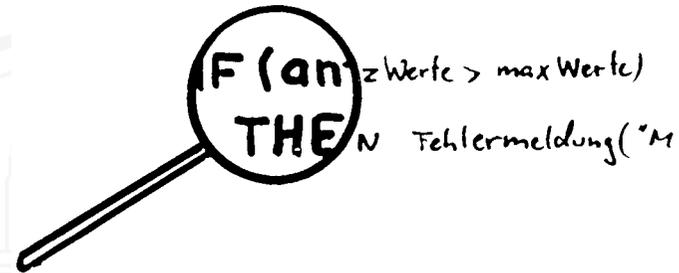
9.3 Durchführung eines Reviews

9.4 Review-Verfahren

9.5 Review-Aufwand

Zwei Grundformen

- Inspektion



- Walkthrough

WHILE Developing Software
DO Reviews - forever
ELSE You won't get
to programmer's heaven
END

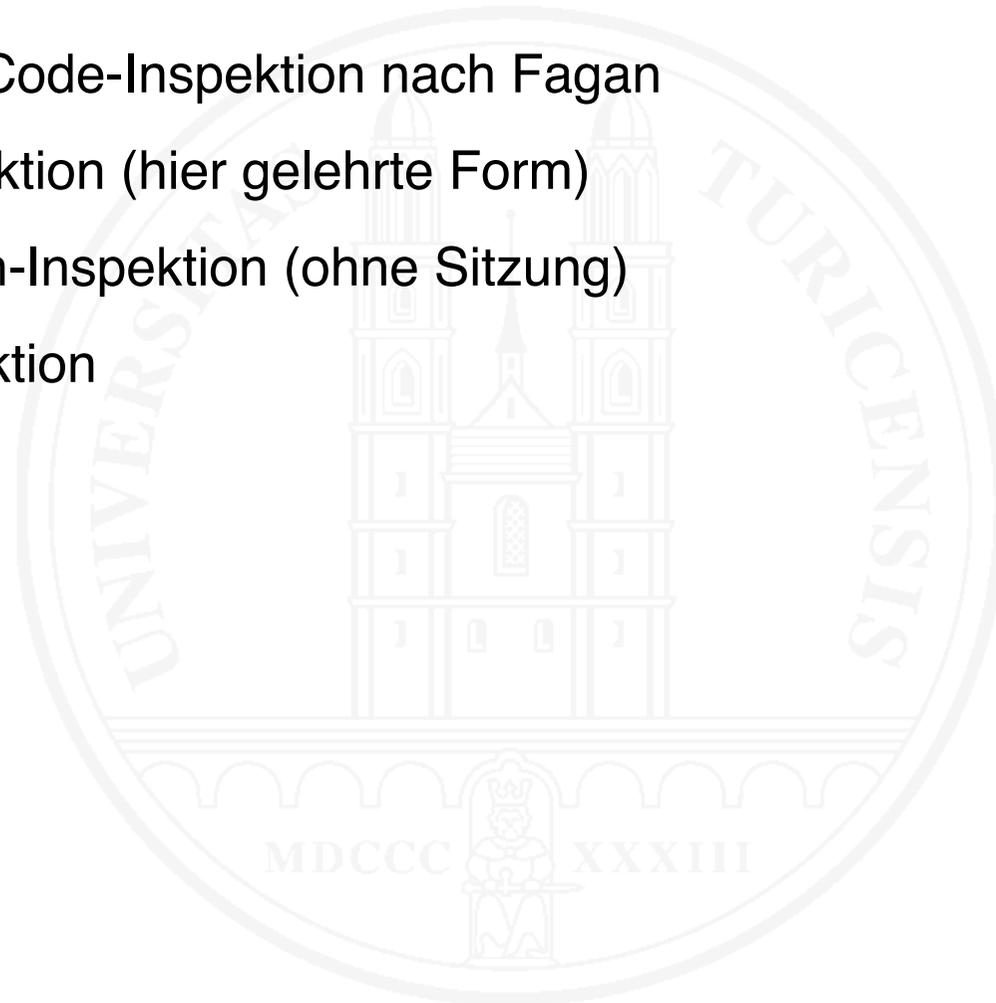
Inspektion

Prinzip: Der Prüfling wird von Gutachtern nach vorgegebenen Prüfkriterien Zeile für Zeile inspiziert

- Gutachter **bereiten sich individuell vor**
- Gutachter müssen mit den verwendeten Entwicklungsmethoden und -sprachen vertraut sein
- Prüfverfahren: siehe Kapitel 9.4
- Nur sinnvoll, wenn gegen klare Vorgabedokumente geprüft werden kann
- Von **Fagan** (1976) für Code- und Designüberprüfungen entwickelt
- **Wirkungsvollste Reviewtechnik**

Arten von Inspektionen

- Klassische Code-Inspektion nach Fagan
- Team-Inspektion (hier gelehrte Form)
- N-Individuen-Inspektion (ohne Sitzung)
- Selbstinspektion



Klassische Code-Inspektion nach Fagan

- Die **Urform** des technischen Reviews für Software (Fagan 1976)
- **Ablauf:**
 - **Startsitzung:** Vorstellen des Prüflings
 - Vorbereitung der Gutachter: Gutachter **lesen** den Code und müssen ihn **verstehen**; sie erstellen **keine individuellen Befundlisten**
 - Sitzung: Rollen wie in 9.3 beschrieben; ferner zusätzliche Rolle: **Vorleser**
- Der Vorleser **liest den Code** Zeile für Zeile **vor**
- Bei jeder Zeile können die Gutachter einhaken und **Befunde vorbringen**

Team-Inspektion

- Die hier gelehrt Inspektionsform
- Prüfung und Befunderhebung erfolgt individuell durch Gutachter in Vorbereitung, Gutachter erstellen individuelle Befundlisten bringen diese in die Sitzung mit
- Review-Sitzung dient nur zum Zusammentragen und Bewerten der Befunde
- Doubletten und falsche Befunde werden eliminiert
- Gemeinsame Befundliste und Empfehlung repräsentiert den Gruppenkonsens
- Nachteil: Aufwand für die Sitzung

N-Individuen-Inspektion (ohne Sitzung)

- Prüfung und Befunderhebung vollständig **individuell** durch Gutachter; **keine** gemeinsame **Sitzung**
- Review-Befund ist die **Summe der Gutachterbefunde**
- Wird aus Effizienzgründen teilweise propagiert
- **Spart** den **Aufwand** für die **Sitzung**
- Experimente zeigen, dass die Sitzung kaum neue Befunde erbringt (die kein Gutachter in der Vorbereitung erkannt hat)
- Kritische Durchsicht der Individualbefunde durch die Gruppe fehlt:
 - Befundliste enthält **Redundanzen**
 - **Befunde** sind **nicht bewertet**
 - Experimente zeigen, dass die Anzahl der **falschen Befunde ansteigt**

Selbstinspektion

- **Wie N-Individuen-Inspektion**, aber Autor ist sein eigener (und einziger Gutachter)
- **Weniger effektiv als** Inspektionen durch **Fremdgutachter**
- Jederzeit möglich (keine Probleme mit der Verfügbarkeit von Fremdgutachtern)
- **Effektiver** und **effizienter als** einfaches **Durchlesen** (da auf rigorosen Prüfverfahren basiert)
- Besser als Laufversuche und ad hoc Test

Walkthrough

Prinzip: Autor geht Prüfling mit Gutachtern durch, die Darstellung wird gemeinsam nachvollzogen

- Vorbereitung nur beschränkt möglich; **Gutachter prüfen in der Sitzung** durch kritisches Mitvollziehen der Ausführungen des Autors
- Gutachter müssen die verwendeten Entwicklungsmethoden und -sprachen nur soweit kennen, dass sie den Ausführungen des Autors folgen können
- Walkthroughs erfordern nicht zwingend klare **Vorgabedokumente**. Es kann auch **gegen Ideen und Vorstellungen** von (beim Walkthrough anwesenden) Anwendern, Fachgebietsexperten, o.ä. **geprüft** werden
- **Weniger wirksam** (geringere Fehlerentdeckungsrate) **als Inspektion**. Gefahr, dass Gutachter sich von geschickt und flüssig vortragendem Autor blenden lassen

9.1 Grundlagen

9.2 Review-Formen

9.3 Durchführung eines Reviews

9.4 Review-Verfahren

9.5 Review-Aufwand

Durchführung eines Reviews (Inspektion)

- **Planung**
Termine einplanen – Aufwand budgetieren – Teilnehmer einladen – Material verteilen
- **Vorbereitung**
Teilnehmer bereiten sich individuell vor – inspizieren den Prüfling – notieren Befunde
- **Sitzung**
Moderiertes Zusammentragen und Bewerten der Befunde – Erstellen des **Review-Berichts**
- **Überarbeitung und Nachkontrolle** (nach dem Review)
Entscheidung über Änderungen – Durchführen der Änderungen – Nachkontrolle durch Projektverantwortlichen oder neues Review

Review-Material

- Prüfunterlagen
 - Der zu prüfende Gegenstand in lesbarer Form
 - ⇒ Jedes von Menschen lesbare Artefakt ist reviewbar

- Referenzunterlagen
 - sachlich-inhaltliche Vorgaben
 - vorgeschriebene Standards
 - Prüflisten (Checklisten)

Rollen der Beteiligten (Inspektion)

- **Moderator**
organisiert und leitet
- **Gutachter**
prüft, nennt Befunde, bewertet, verhält sich positiv und kooperativ
- **Schreiber**
protokolliert für alle Beteiligten sichtbar
- **Autor**
hört zu, verhält sich passiv

Das sind
Rollen, keine
Geschlechter



Der Review-Bericht

- **Formblätter** verwenden
- **Liste der Befunde entsteht öffentlich** während der Sitzung
(Datenprojektion, Folien auf Hellraumprojektor, kopierbare Tafel)
- Bericht am Ende des Reviews **sofort fertigstellen**
- Alle **unterschreiben**

Mini-Übung 9.1

Warum ist es wesentlich, dass die Befundliste öffentlich während der Sitzung erstellt wird?

Review-Bericht: Deckblatt – 1

Review-Bericht	Review-Nr.:	Seite _____ von _____
	Datum:	Zeit von: _____ bis: _____
Zu prüfendes Produkt		
Nummer / Version	Titel	
Referenzunterlagen		
Nummer / Version	Titel	
Bewertung	<input type="checkbox"/> akzeptiert (kein neues Review erforderlich) <input type="checkbox"/> nicht akzeptiert (neues Review erforderlich) <input type="checkbox"/> Review nicht beendet	<input type="checkbox"/> wie es ist <input type="checkbox"/> kleine Änderungen <input type="checkbox"/> große Änderungen <input type="checkbox"/> Überarbeitung
Review-Team		

Review-Bericht: Deckblatt – 2

<input type="checkbox"/> Review errorieren)				<input type="checkbox"/> Überarbeitung	
<input type="checkbox"/> Review nicht beendet					
Review-Team					
Name		Funktion	Vorbereitungszeit	Unterschrift	
Freigabe (nach Nachkontrolle)					
Name		Datum		Unterschrift	

Review-Regeln – 1

- Material **rechtzeitig** vor Sitzung **verteilt**
- Alle Teilnehmer **rechtzeitig eingeladen**
- Alle Gutachter kommen **vorbereitet** (unverzichtbar bei Inspektionen!)
- **3** bis **7** Beteiligte
- Sitzungsdauer **maximal 2 h**
- Nicht mehr Material verteilen, als in Sitzung zu bewältigen ist
- Probleme nur **nennen**, nicht lösen
- **«Dritte Stunde»** nach Review-Sitzung zur Diskussion von Problemlösungen

Review-Regeln – 2

- Positives und Negatives nennen
- Keine Stilfragen diskutieren
- Produkt bewerten, nicht Produzenten
- Review-Berichte niemals zur schematischen Bewertung von Mitarbeitern verwenden
- Anhand von Standards und Prüflisten bewerten
- Fehler in Referenzunterlagen ebenfalls erheben und in separater Befundliste notieren

Mini-Übung 9.2 (Aufgabe 9.2 im Skript) – 1

Sie sind Projektleiter in einem Software-Entwicklungsprojekt. Soeben haben Sie den Review-Bericht über das Lösungskonzept der zu entwickelnden Software erhalten. Der Bericht weist neben einer Reihe leichter Fehler drei schwere Fehler und eine als kritisch eingestufte Auslassung aus. Sie sind gegenüber dem Zeitplan drei Wochen in Verzug und möchten daher so rasch wie möglich mit dem Detailentwurf und der Codierung beginnen lassen.

Wie verhalten Sie sich?

Was halten Sie von folgenden Maßnahmen?

Mini-Übung 9.2 (Aufgabe 9.2 im Skript) – 2

- a) Sie lassen alle festgestellten Mängel beheben und beginnen nicht mit dem Detailentwurf, bis das Dokument das Nachreview bestanden hat und freigegeben ist.
- b) Sie lassen sofort mit Detailentwurf und Codierung beginnen und beschließen, die Fehler und Lücken erst zu beseitigen, wenn man beim Entwerfen bzw. Codieren an diese Stellen kommt.
- c) Sie gehen den Review-Bericht durch, markieren die Punkte, die Ihnen schwerwiegend erscheinen und lassen nur diese beheben.
- d) Sie veranlassen die Behebung der festgestellten Fehler, lassen aber gleichzeitig mit dem Detailentwurf eines Teilsystems beginnen, in dessen Konzept nur zwei leichte Mängel gefunden wurden.

9.1 Grundlagen

9.2 Review-Formen

9.3 Durchführung eines Reviews

9.4 Review-Verfahren

9.5 Review-Aufwand

Mögliche Verfahren

Verfahren	Inspektion	Walkthrough	Selbstinsp.
Paraphrasieren	+	0	+
Checklisten durchgehen	+	-	0
Prüfprozeduren durcharbeiten	+	-	0
Szenarien durchspielen	+	+	+
Manuell ausführen	+	-	+
Verschiedene Perspektiven einnehmen	+	0	+
Vorlesen	0	0	-
Pfade verfolgen	0	+	0
Rollenspiele	-	+	-

Erläuterungen

- Paraphrasieren:
Das Gelesene mit **eigenen Worten wiedergeben** und **erklären**
- Verschiedene Perspektiven einnehmen:
Sich in eine **andere Rolle** versetzen und den Prüfling aus dieser Perspektive untersuchen, insbesondere die Rollen des
 - **Benutzers**
 - **Testers**
 - **Pflegeprogrammierers**
- Manuell ausführen:
Abläufe mit ausgewählten Daten von Hand durchspielen („**Schreibtischtest**“)
- Pfade verfolgen:
beispielsweise **Ablaufpfade** von Prozessen oder **Datenflüsse**

Checklisten

- Auflistung von **Prüfpunkten**
- Müssen **auf die Art des Prüflings abgestimmt** sein (Anforderungen, Entwürfe, Code, ...)
- Gegebenenfalls **abzustimmen auf die Methodik**, mit welcher der Prüfling erstellt wurde

Beispiel: Checkliste für Anforderungsspezifikationen (Auszug)

- Allgemeine Fragen
 - Sind alle Anforderungen adäquat, d.h. drückt jede Anforderung ein Bedürfnis der Benutzer aus?
 - Sind alle Anforderungen klar und eindeutig formuliert?
 - ...
- Methodengebundene Fragen
 - Hat jede Klasse einen eindeutigen, treffenden Namen?
 - ...
- Fragen zur Adäquatheit und Vollständigkeit
 - Sind alle genannten Funktionen notwendig zur Erreichung der Systemziele?
 - Ist jedes Systemziel durch Funktionen abgedeckt?
 - ...

Prüfprozeduren

- **Handlungsanweisungen** zur Prüfung
- So aufgebaut, dass die **Prüfpunkte einer Checkliste abgedeckt** sind
- Werden in der Literatur (z. B. Porter, Votta und Basili 1995) als Szenarien bezeichnet (was eigentlich ein Missbrauch des Szenarienbegriffs ist)
- Beispiel:
Bestimme alle Datenobjekte, welche das System mit seiner Umgebung austauscht. Für jedes Datenobjekt
 - Ist der Name adäquat?
 - Ist der Datentyp spezifiziert und adäquat?
 - Wenn das Datenobjekt ein Ausgabeobjekt ist, gibt es mindestens eine Funktion, welche dieses Objekt erzeugt?
 - ...

Szenarien

- **Durchspielen** problembezogener **Benutzungsszenarien**
- Beispiele:
 - Prüfung der Anforderungen an ein Bibliothekssystem
 - Ausleihen eines Buchs
 - Zurückgeben eines vorgemerkten Buchs
 - ...
 - Inspektion des Codes für eine verkettete Liste
 - Einfügen in die leere Liste
 - Einfügen in der Mitte einer Liste
 - Suchen des letzten Elements der Liste
 - ...

9.1 Grundlagen

9.2 Review-Formen

9.3 Durchführung eines Reviews

9.4 Review-Verfahren

9.5 Review-Aufwand

Aufwand-Anteile

Im Wesentlichen nur **Personalaufwand**:

Gesamtaufwand =

Zeitbedarf für **Organisation** des Reviews

+ Summe der **Vorbereitungszeiten** aller Gutachter

+ **Dauer** der Sitzung x **Anzahl der Teilnehmer**

+ Aufwand für **Nachreviews** bei zu vielen Befunden

Aufwandzahlen aus der Literatur

	1	2	3	4	5	6	7
Vorbereitung	1 p 60 l	3-5 p 150 l	k A 125 l	k A 150 l	10 p 150 l	25-50 p 150 l	k A <200 l
Sitzung	2 p 120 l	3-5 p 150 l	90- 125 l	k A 150 l	25 p 300 l	12-20 p 75 l	20 p k A

p Seiten eines Dokuments

l Codezeilen ohne Kommentar (NCSS)

k A keine Angabe

1 Gilb und Graham 1993, p. 376-377, 408

2 Strauss und Ebenau 1994, p. 89

3 Fagan 1986

4 Barnard und Price 1994

5 Frühauf, Ludewig und Sandmayr 1991, p. 113 (1 p in [5] entspricht 30 NCSS, pers. Mitteilung, 1994)

6 Jones 1991

7 Weller 1993

Literatur

Barnard, J., A. Price (1994). Managing Code Inspection Information. *IEEE Software* **11**, 2 (Mar 1994). 59-69.

Fagan, M.E. (1976). Design and Code Inspections to Reduce Errors in Program Development. *IBM Systems Journal* **15**, 3. 182-211.

Fagan, M.E. (1986). Advances in Software Inspections. *IEEE Transactions on Software Engineering*, **SE-12**, 7 (Jul 1986). 744-751.

Freedman D.P., G.M. Weinberg (1982). *Handbook of Walkthroughs, Inspections and Technical Reviews*. 3rd edition. Boston, Toronto: Little, Brown and Co.

Frühaufl, K., J. Ludewig, H. Sandmayr (1991). *Software-Prüfung. Eine Fibel*. Zürich: vdf und Stuttgart:Teubner.

Gilb, T., S. Graham (1993). *Software Inspection*. Wokingham, etc.: Addison-Wesley.

Jones, C. (1991). *Applied Software Measurement*. New York: McGraw-Hill.

Porter, A.A., L.G. Votta and V.R. Basili (1995). Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment. *IEEE Transactions on Software Engineering* **21**, 6 (Jun 1995). 563-575.

Porter, A. and L. Votta (1997). What Makes Inspections Work? *IEEE Software* **14**, 6 (Nov-Dec 1997). 99-102.

Russell, G.W. (1991). Experience with Inspection in Ultralarge-Scale Developments. *IEEE Software* **8**, 1 (Jan 1991). 25-31.

Literatur – 2

Schnurer, K. (1988). Programminspektionen. Erfahrungen und Probleme. *Informatik-Spektrum* **11**, 6 (Dez 1988). 312-322.

Strauss, S.H., R.G. Ebenau (1994). *Software Inspection Process*. New York, etc.: McGraw-Hill.

Weller, E.F. (1993). Lessons from Three Years of Inspection Data. *IEEE Software* **10**, 5 (Sept 1993). 38-45.

Siehe auch Literaturverweise im Kapitel 9 des Skripts.

Im Skript [M. Glinz (2005). *Software Engineering*. Vorlesungsskript, Universität Zürich] lesen Sie bitte Kapitel 9.4.

Im Begleittext zur Vorlesung [S.L. Pfleeger, J. Atlee (2006). *Software Engineering: Theory and Practice*, 3rd edition. Upper Saddle River, N.J.: Pearson Education International] lesen Sie bitte Kapitel 8.3 sowie Seite 389.