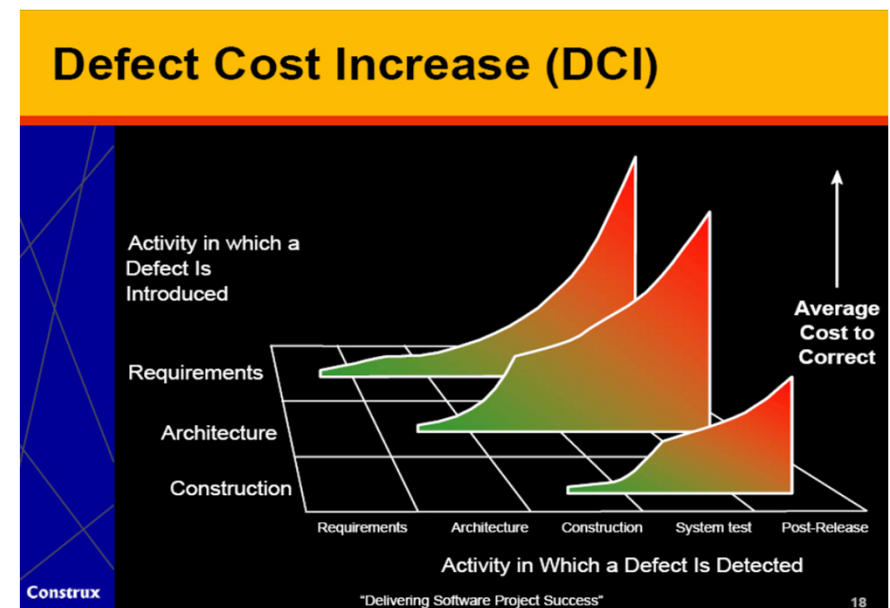


- 01 Einführung
- 02 Prozessmodelle
- 03 Konfigurationsmanagement
- 04 Requirements Engineering
- 05 Modellierung
- 06 Qualitätsmanagement
 - 06.1 Qualität und Qualitätssicherung
 - 06.2 Manuelle Prüfmethoden

Wozu manuelle Prüfung?

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethoden

- Prüfung, ob Inhalte/Code schlüssig, sinnvoll sind
 - Verständnis des Inhalts/Codes (Zählen genügt nicht)
- Dokumente sind nur manuell prüfbar
- Manuelle Prüfung auch Mittel, um
 - Informationen zu verbreiten
 - Inhalte abzustimmen
 - Mitarbeiter einzuarbeiten
- Manuelle Prüfungen **finden mehr Fehler pro Stunde als Tests**



Review – Was ist das?

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethode

- Bei einem Review werden **Dokumente** aller Art nach bestimmten Qualitätskriterien geprüft
- **Was** wird geprüft?
 - Studien
 - Spezifikation (Fachkonzepte), Entwürfe
 - Planungen, Projekthandbücher, Angebote, ...
 - Code
- **Welche Eigenschaften** werden untersucht?
 - Korrektheit
 - Angemessenheit
 - Lesbarkeit
 - Änderbarkeit
- Häufig ist ein Phasenübergang im Projekt an ein Review gebunden (**Gateway**)



Durchführung eines Reviews

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethoden

1. Planung

Reviews müssen in der Entwicklung eingeplant werden.

2. Vorbereitung

Die Teilnehmer an einem Review erhalten das zu prüfende Material sowie die Referenzunterlagen im Voraus und bereiten sich individuell auf die Sitzung vor.

3. Sitzung

Die Review-Sitzung wird von einem Moderator geleitet. Er sorgt für einen geordneten Sitzungsablauf und wacht über die Einhaltung der Review-Regeln.

4. Review-Protokoll und Review-Bericht

Während der Sitzung wird ein Review-Protokoll erstellt.

5. Überarbeitung und Nachkontrolle

Der Projektverantwortliche entscheidet aufgrund des Review-Protokolls über die durchzuführenden Änderungen.



Wie läuft ein Review ab? (1)




06 Qualitätsmanagement / 06.2 Manuelle Prüfmethoden

- Ein **Qualitätsbeauftragter** wird bestimmt, der das Review organisiert
- Der Qualitätsbeauftragte ...
 - ... legt das Review-Team und den **Termin** fest (Experten aus Fachbereich/Technik)
 - ... erhält die fürs Review vorgesehenen **Dokumente**
 - ... **lädt** spätestens eine Woche **vor** dem Termin zum Review ein
 - ... **verteilt** spätestens eine Woche **vor** dem Termin die Dokumente
 - ... sammelt vorab eingereichte Anmerkungen des Review-Teams
- Das Review-Team ...
 - ... enthält **keinen Autor** der Dokumente (nicht sich selbst reviewen)
 - ... liest die Dokumente **vor** dem Review-Termin und macht Anmerkungen



Wie läuft ein Review ab? (2)

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethode

- Der Review-Termin hat eine begrenzte, vorher bestimmte Dauer
 - er dauert aber maximal zwei Stunden
- Am Review nehmen das Review-Team und die Autoren teil
- Der Qualitätsbeauftragte
 - moderiert das Review
 - bestimmt Protokollanten
- Der Protokollant führt ein Ergebnisprotokoll (Review-Anmerkungen sowie Beschlüsse dazu)
- Am Ende des Reviews: „Qualitäts-Ampel“
 -  Schwere Fehler, deutliche Nachbesserungen, kein Gateway
 -  Nachbesserungen gemäß Anmerkungen, Gateway
 -  kaum Nachbesserungen, Gateway



Review Anmerkungen

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethode

- Globale Identifikation
- Identifikation des Reviewers
- Dokument
- Seite(n), Zeile(n)
- Anmerkung
- Art der Anmerkung, Priorität
- Stellungnahme zur Anmerkung
- Verantwortlicher und Erledigungsdatum

ID	Reviewer	Dokument	Seite(n)	Anmerkung	Art/ Prio	Antwort des Autors	Verant- wortung	Erledi- gungs- datum
1.	<Name>	<Name des Dokuments>	<Seite mit der Anmerkung>	<Beschreibung der Anmerkung, des Fehlers>		<Kommentar des Autors, z.B.: akzeptiert, Beschreibung der Änderung oder zurückgewiesen>	<Verantwortlicher für die Behebung des Fehlers / Durchführung der Änderung>	<Bis wann ist die Änderung durchgeführt?>
2.								
3.								
4.								
5.								


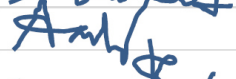

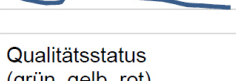
Art: A = Anmerkung/Kommentar; E = Ergänzung; K = Korrektur / Berichtigung; S = Streichung
Priorität: 1 = Sehr hoch / Kritisch; 2 = hoch; 3 = mittel; 4 = gering



Review Protokoll

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethode

- Ergebnisprotokoll!
- Zweck
 - Dokumentation der QS
 - Aufgabenliste (Aufgabenverteilung + Nachkontrolle)
- Inhalte
 - Teilnehmer (mit Unterschrift)
 - Verteiler
 - Review-Gegenstände
 - Beschlüsse, Aufgaben, Feststellungen
 - Review-Ergebnis (z.B. Qualitätsampel)

Review Protokoll				
Termin vom:	01.07.2020			
Ort:	TH Rosenheim			
Protokoll am:	01.07.2020			
Ersteller:	U.-N. Bekannt			
Teilnehmer:	Name	Kürzel	Unterschrift	
	U.N. Bekannt	UB		
	Anubis	AN		
	Isis	IS		
	Osiris	OS		
Review-gegenstände:				Qualitätsstatus (grün, gelb, rot)
1	Spezifikation xy			<input type="radio"/> <input checked="" type="radio"/> <input type="radio"/>
Nr.	A/B/F	Text	Wer?	Termin
1.	F	Datenmodell nicht Review-fähig, da es zu spät geliefert wurde.		
2.	A	Zeitraum optional einfügbar bei AF_Mietverhältnis. Default-Wert 3 Monate	Anubis	3.7.2020
3.	A	1.3 Dialog Vertrag eingeben Feld für Datenbankfelder Bemerkung und qmPreis einfügen	Osiris	3.7.2020
4.				
A = Aufgabe, B = Beschluss, F = Feststellung				



Review Regeln (1)

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethode

- Zu prüfendes Material wird **rechtzeitig** vor der Sitzung **verteilt**.
Ein „Review“ von Tischvorlagen ist Zeitvergeudung.
- Teilnehmer werden **rechtzeitig eingeladen**.
- Alle kommen **vorbereitet** zur Sitzung.
Der Moderator bricht das Review ab, wenn mehrere Teilnehmer nicht vorbereitet sind.
- An einem Review nehmen mindestens drei und höchstens sieben Personen teil.
- Sitzung dauert maximal 2 Stunden.
Daher Umfang des zu prüfenden Materials einschränken!



Review Regeln (2)

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethode

- Probleme werden in der Sitzung **nur genannt, nicht gelöst**.
- Jeder Gutachter nennt mindestens **einen positiven und einen negativen Punkt**.
- Stilfragen werden nicht diskutiert.
- Das **Produkt wird bewertet, nicht deren Produzenten**.
- Prüfung einfacher und effizienter, wenn Standards und Prüflisten vorhanden, nach denen bewertet werden kann.
- Häufig: Beim Review auch Fehler in den Referenzunterlagen gefunden. Diese werden auch notiert.



● Was tut folgender Quelltext?

```
public class X {  
    private static int M = 1;  
    public static Map hugo(Collection c) {  
        Iterator q = c.iterator(); Map a = new HashMap();  
        while (q.hasNext()) {  
            Object x = q.next();  
            Integer b = (Integer) a.get(x);  
            if (null == b) a.put(x, new Integer(M));  
            else a.put(x, new Integer(b.intValue() + M));  
        }  
        return a;  
    }  
}
```

● Besser?

```
public class Statistik
{
    public static Map berechneHistogramm(Collection listeMitDuplikaten)
    {
        Iterator iterator = listeMitDuplikaten.iterator();
        Map histogramm = new HashMap();
        while (iterator.hasNext())
        {
            Object zuZaehlendesObjekt = iterator.next();
            Integer anzahlObjekte = (Integer) histogramm.get(zuZaehlendesObjekt);
            if (null == anzahl)
                histogramm.put( zuZaehlendesObjekt, new Integer(1));
            else
                histogramm.put( zuZaehlendesObjekt,
                               new Integer(anzahlObjekte + 1));
        }
        return histogramm;
    }
}
```

Gütebegriff/Qualitätsmodell für Code

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethoden

- Code ist für Menschen!
 - Ca. 50% des Änderungsaufwands nur Lesen und Verstehen des Codes
 - Daher: Je verständlicher der Code desto weniger Wartungsaufwand
 - Aber: Gütebegriff ist individuell für die Organisation
- Wie misst man denn nun Code-Qualität?
 - Zusammenhänge kaum empirisch erforscht
 - Daher: Auf Erfahrung der Entwickler bauen und Festlegungen treffen
 - Verständlichkeit über gute Namen, Kommentare und Formatierung
→ siehe nachfolgende Folien
- Wozu dient der Gütebegriff / das Qualitätsmodell?
 - Sie können in Ihrem Projekt feststellen, wie gut Ihr Code ist
 - Festlegung von Optimierungszielen
 - Lieferantenkontrolle = Element der Abnahme, der permanenten Kontrolle



- Manuelle Prüfung des Quelltextes
 - Kostet Zeit, nur für kleine Code-Mengen möglich (weniger 100 Zeilen)
 - Liefert **qualitative** Aussagen
 - Kann zur Einarbeitung und zur Wissensverbreitung dienen
- Was sollte mit Reviews geprüft werden?
 - Verständlichkeit des Codes
 - Güte der Namen von Variablen, Klassen und Modulen
 - Güte der Kommentare
 - Wahl der korrekten Datenstrukturen und Algorithmen
- Wichtig: **Systematisch** den kompletten Quelltext reviewen
- Werkzeugunterstützung nutzen
 - Z.B. Eclipse Plug-Ins zur Unterstützung von Reviews
 - WWW Schlagworte → ... source code review plug-in ...

Was ist guter Code?

Einer muss sich plagen – der Schreiber oder der Leser [Wolf Schneider]

Guter Code – Passende Namen sind wichtig!

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethoden

- Passende Namen für **Klassen**, **Methoden**, **Attribute**

- Bedeutung der Namen im Quelltext

- Beispiel: Code Analyse Eclipse (F. Deißeböck, M.Pizka)
- Anteil der Namen (Bezeichner) im Eclipse Quelltext = 72% (!)
- Ca. 98.000 verschiedene Bezeichner-Namen (~Oxford Dictionary)
- Zum Vergleich: Grundwortschatz Englisch: ca. 5000 Begriffe

Type	#	%	chars	%
Keywords	967.665	11%	4.650.273	13%
Delimiters	4.096.112	47%	4.096.112	11%
Operators	531.444	6%	669.932	2 %
Identifiers	2.873.232	32%	25.646.263	72%
Literals	301.081	3%	708.308	2%
Total	8.769.534	100%	35.770.888	100%

Quelle: F. Deißeböck, M.Pizka: *Concise and Consistent Naming*, 13th IEEE Workshop on Program Comprehension, 2005

- Verwendete Namen = **Vokabular** Ihres Programms

- = Abstraktion oberhalb der Programmiersprache

- Ziel Namensgebung: **Menschlicher Leser** versteht, **was gemeint ist**

- Hat richtige Assoziation, korrektes mentales Modell

- Name = Vermutung über Inhalt der Variable / Klasse / Methode

- → Vermutung sollte richtig sein (Prinzip der kleinsten Überraschung)



- Beispiel: Java aus SUN Coding Conventions
 - Namen von Methoden, Variablen, Attributen und Packages starten immer mit Kleinbuchstaben
 - Namen von Klassen und Interfaces starten immer mit Großbuchstaben
 - Konstanten nur Großbuchstaben (z.B. **PI**, **JAVA_HOME**, ...)
 - Groß/Klein-Buchstaben-Mix, um Namen **lesbarer** zu machen
 - z.B. **isReadyForUse()** oder **getFirstElement()** für Methoden
 - oder **UniqueNameSignature** für eine Klasse
 - Ganze Namen verwenden, welche die Variable, Klasse oder Methode sinnvoll beschreiben
 - Z.B. **firstName** anstelle von **fName**
 - oder **xPosition** anstelle von **x1**

Prüfung mit **Checkstyle** oder einem anderen Style-Checker

Guter Code – Angemessene Kommentare sind wichtig (1)

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethode

- Kommentare helfen **menschlichem Leser zum Verständnis**
 - zum Ändern des Codes
 - zum Finden von Fehlern
 - zum Benutzen des Codes
- Kommentare erklären das **WARUM** (das „Wie“ steht ja im Code)
- Kommentare mildern Probleme der Programmiersprache ab
 - Design by Contract in Java nicht direkt umgesetzt (Vor- und Nachbedingungen als Kommentare + assert im Code)
 - z.B. @pre, @post, @invariant
- Kommentare helfen, den Kontext herzustellen
 - z.B. verweisen auf Fachkonzept, Literatur, ...
- Kommentare zeigen Verwaltungsinformationen
 - Autor, Release-Datum, ID der Anforderung, ...



Guter Code – Angemessene Kommentare sind wichtig (2)

- ... aber nicht so!
 - z.B. bei (vorgeschriebenen) Kommentaren für Methoden und Funktionen als JavaDoc

```
/** Das ist nicht besonders nützlich ...
 * @param title           The title of the CD
 * @param author          The author of the CD
 * @param tracks           The tracks of the CD
 * @param durationInMinutes The duration of the CD in Minutes
 * @return nothing
 */
public void addCD(String title, String author, int tracks,
                  int durationInMinutes)
{
    // ...
}
```

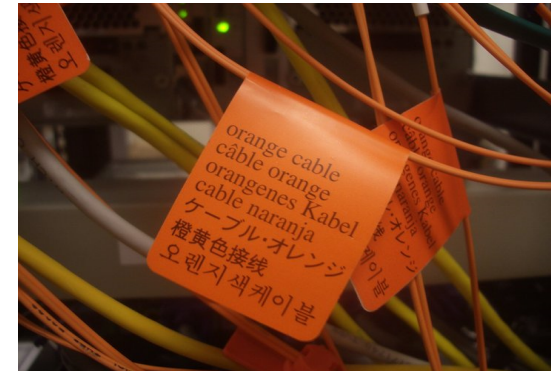
- Problem häufig: Werkzeug prüft Existenz von Kommentaren
 - Entwickler generiert Kommentar mit Eclipse oder kopiert alten Kommentar
 - Daher immer Kommentare manuell nachprüfen

Guter Code – Angemessene Kommentare sind wichtig (3)

06 Qualitätsmanagement / 06.2 Manuelle Prüfmethode

- Unsinnige Kommentare

```
// i wird bis 30 gezählt
for (int i=0; i <=30; i++)
...
// Diese Exception wird ignoriert
} catch (Exception achWas) {}
```



- Besser: Warum tut der Code etwas?

```
// Alle Tage im Monat durchlaufen
for (int i=0; i <=30; i++)
// Noch besser: Sprechender Name für i (z.B. tag) und 30
// durch Konstante ersetzen
...
// Beim Freigeben der Ressourcen keine
// sinnvolle Ausnahme Behandlung möglich
} catch (Exception ignore) {}
```

● Statische Analyse

- Sind alle Operationen / Methoden / usw. kommentiert?
- Voraussetzungen für Werkzeugunterstützung
 - Style-Checker kennt die Kommentarrichtlinien (aus Style Guide)

● Metriken

- Gibt es ein Code-zu-Kommentar Verhältnis von ca. 25%?

● Manuelles Review auf Verständlichkeit und Vollständigkeit

- Operationen / Methoden: Dokumentation für Parameter / Ergebnis vorhanden?
 - Vorbedingungen = Erlaubte Werte für Parameter
 - Nachbedingungen = Mögliche Ergebnisse + Ursachen
 - Invarianten = müssen immer erfüllt sein
- Erklären die Kommentare das „Warum“ und nicht das „Wie“?