

# Einführung in Software Engineering

**Barbara Paech, Marcus Seiler**

Institute of Computer Science  
Im Neuenheimer Feld 326  
69120 Heidelberg, Germany  
<http://se.ifi.uni-heidelberg.de>  
[paech@informatik.uni-heidelberg.de](mailto:paech@informatik.uni-heidelberg.de)



RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG

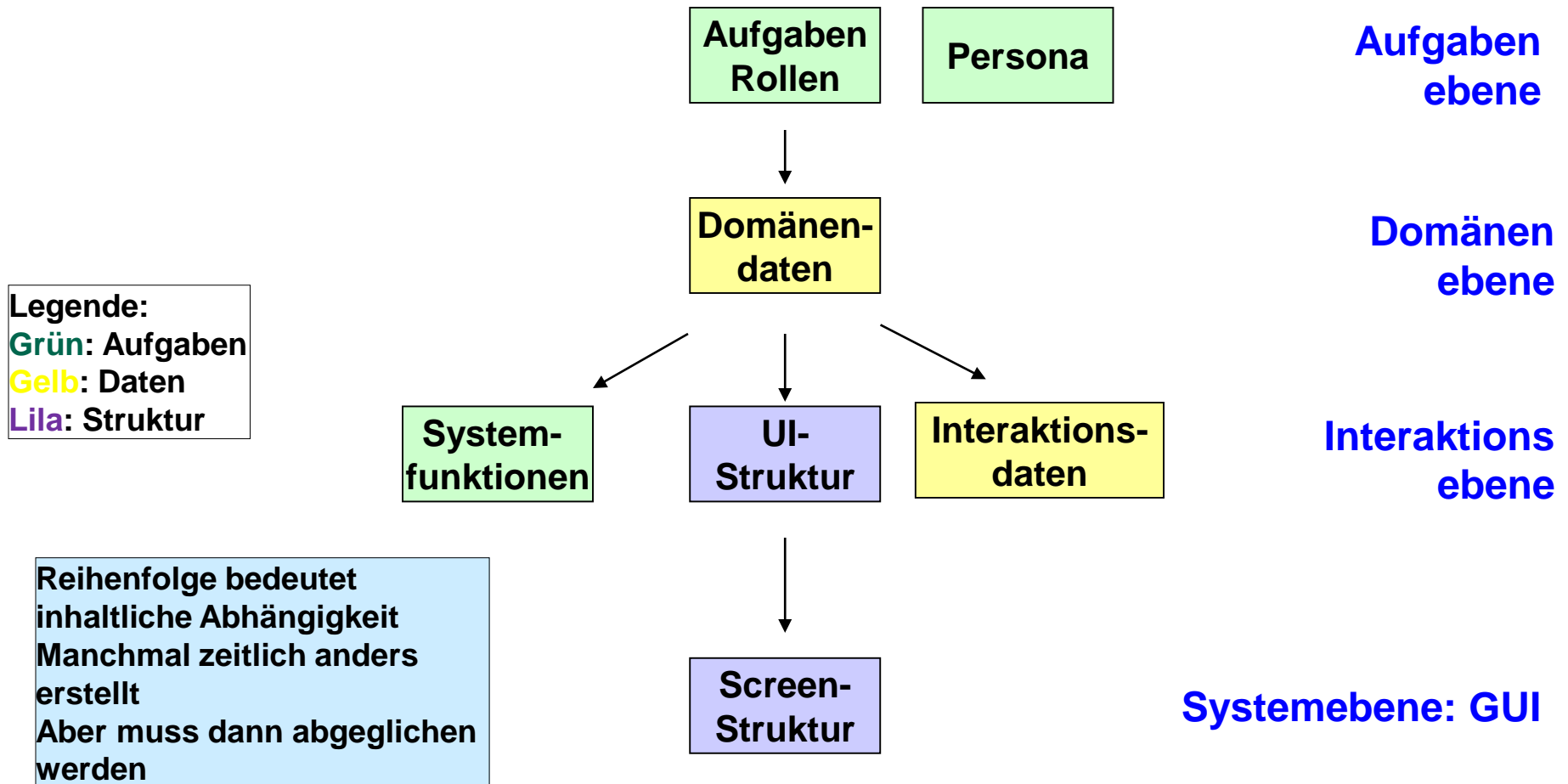


## 3. SWE im Großen (2. Teil)

---

- 3.1. Vorgehen
  - SCRUM Agiles Vorgehen im Großen
- 3.2. Einführung Kommunikation mit KundInnen
- 3.3. Nutzungsmodellierung
  - Einführung
  - Aufgaben, Rollen
  - Domänen- und Interaktionsdaten
  - Funktionen und UI-Struktur
  - Persona
  - GUI
- 3.4. Dokumentenqualität
- 3.5. Analytische QS für SWE im Großen

# Wdh. Beschreibungsebenen (vereinfacht)





- Für eine Software zur Filmverwaltung
- **Rollenbeschreibung FilmverwalterIn**
  - **Aufgaben:**
    - Filmverwaltung
  - **Erfolgskriterium:**
    - Vollständige Übersicht, schnelle Suche
  - **KommunikationspartnerInnen:**
    - FreundInnen
  - **Innovationsgrad:**
    - gering
- **Nutzerprofil**
  - **Vorwissen Filmverwaltung:**
    - Wahrscheinlich hoch
  - **Vorwissen Software:**
    - Hier kann es Neulinge und Erfahrene geben

**Sehr abstrakt**

- *Nicht alle NutzerInnen sind gleich.*
- **User-Centered Design (UCD)** stellt die NutzerInnen in den Mittelpunkt, um gebrauchstaugliche Produkte erstellen zu können
  - [https://en.wikipedia.org/wiki/User-centered\\_design](https://en.wikipedia.org/wiki/User-centered_design)
- Ein wichtiges Element von UCD sind **Personae**
  - Eine Persona ist eine möglichst realitätsnahe Beschreibung einer fiktiven Person, die stellvertretend für eine Gruppe von NutzerInnen steht
  - Leicht zu lesen, aber schwer zu erstellen
  - Sollte aus NutzerInnenbefragungen gewonnen werden

# Beispiel Persona

Beschreibung der  
aktuellen  
Lebensumstände und  
Persönlichkeit

Beschreibung der  
Einstellung zur  
Technik

Beschreibung  
besonderer  
Nutzungswünsche

**Confident learner**

**Samantha Bell**  
"I'd love to keep in contact with my friends"

Sam is about to go abroad for her gap year, so her parents decided to get her a new camera, to make sure she's able to record everything she gets up to.

She likes the camera as it looks so modern, and it's able to do so much more than a lot of her friends' cameras.

She loves being in contact with people all the time, and finds it's a great way to kill time like when waiting for the bus. She uses a lot of the more advanced features – panoramic shots, online upload and .

When she encounters a problem she ignores it most of the time - she's not sure if she even got a manual with the camera. When she has trouble she can't ignore she speaks to her friends, or goes into a camera store – she wants to be talked through the problem.

**First time user**  
Female, 27 year old, single Student  
Sam prefers to learn how to things by trying things out by herself. She isn't worried about 'breaking' anything. If she does need help she would prefer to not to refer to a manual but "do it herself".

**Needs**  
In order of preference:  
1. To share pictures with her parents  
2. To share her pictures with her friends  
3. To share her pictures with people she meets whilst travelling

**Ideal features**

- Ability to take pictures
- Ability to upload images to personal site using 3G/Wifi
- Allowing others to access her pictures remotely
- Long battery life
- Ability to name and add comments to uploaded images
- Ability to create several albums, and upload pictures to each

**Frustrations**

- Lack of wireless/3G access
- Slow uploads
- Low battery life
- Need to be plugged in to upload images
- Slow shutter speed
- Want to be able to name/add comments to uploaded images
- Getting online is confusing
- Creating new albums

**Key attributes**

Low High

Knowledge ———●————

Experience —●————

Help use —●————

Confidence —————●——

Picture credits – Nerdcoregirl, Flickr CC  
<http://www.flickr.com/photos/nerdcoregirl/>

Webcredible – user experience research & design March 2010

Im Gegensatz zu  
Rolle (Typ) hier  
konkrete Ausprägung  
(Person)  
(wie Klasse und Objekt)



Group: Name	
<b>Biographic Facts</b> (age, gender, etc.) Incl. task attitude	<ul style="list-style-type: none"> <li>soll die Wünsche/Einstellung bzgl. der Software verständlich machen</li> </ul>
<b>Knowledge and attitude with respect to technology</b>	<ul style="list-style-type: none"> <li>beschreibt Kenntnisse und Einstellung zu Technik und Software allgemein und insbesondere zu der zu nutzenden Software</li> </ul>
<b>Needs</b>	<ul style="list-style-type: none"> <li>beschreiben die Hauptnutzungsszenarien, in denen die Person die Software verwenden möchte</li> </ul>
<b>Frustrations</b>	<ul style="list-style-type: none"> <li>beschreibt Feature (Eigenschaften oder Funktionalität) der Software, die die Person ärgern würden</li> </ul>
<b>Ideal Features</b>	<ul style="list-style-type: none"> <li>beschreibt Feature, über die sich die Person besonders freuen würden, insbesondere unter Berücksichtigung von Knowledge und Needs</li> </ul>

- Erstellen Sie die Beschreibung einer Persona für die Nutzung eines Kühlschranks.

## Versierter Nutzer: Hans

### Biographic Facts (age, gender, etc.) Incl. task attitude

- Ist gerade mit seiner Freundin zusammengezogen

### Knowledge and attitude with respect to technology

- Arbeitet in den Semesterferien als Verkäufer im MediaMarkt
- 

### Needs

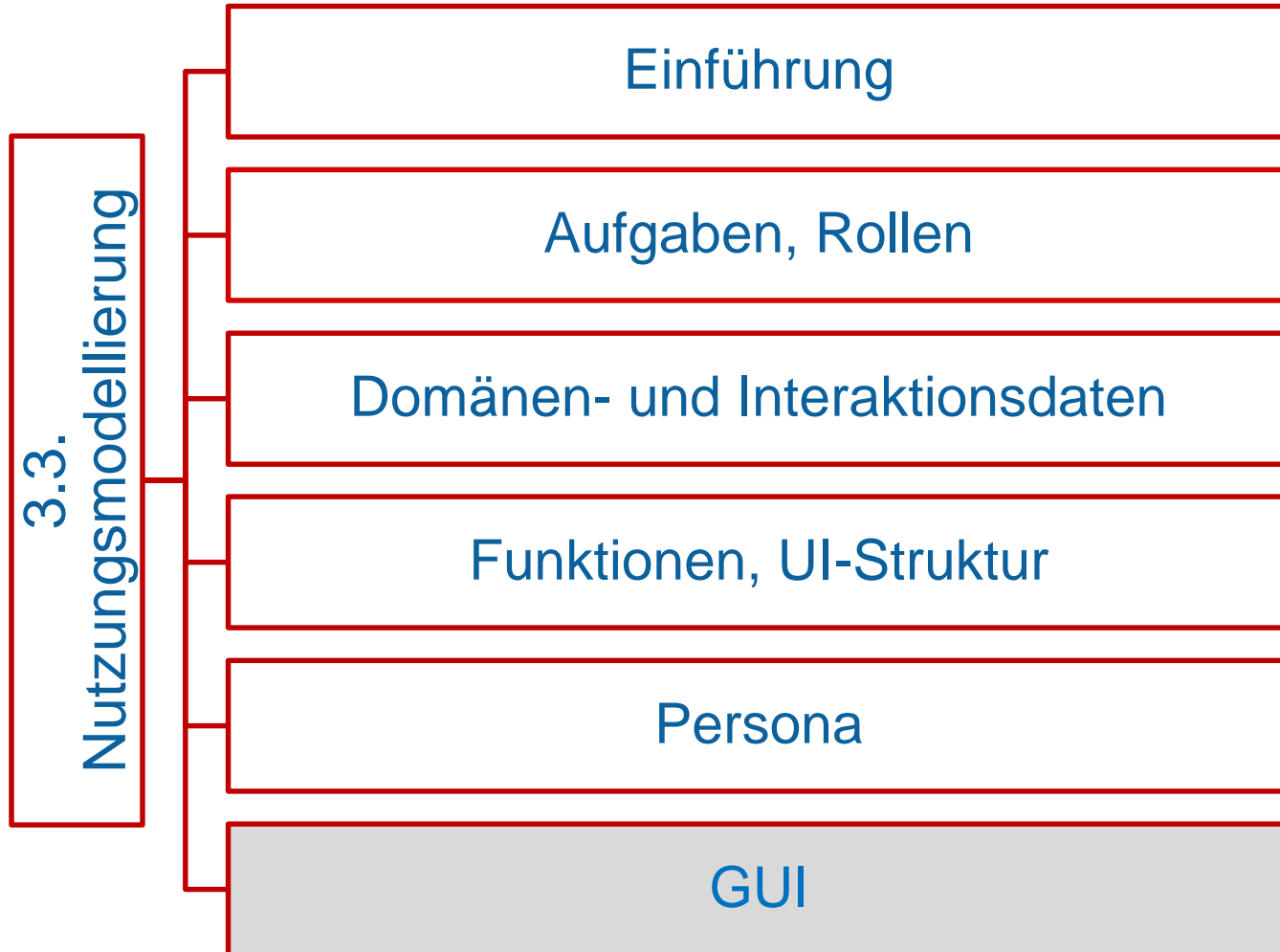
- Möchte im Sommer große Mengen Eistee kühlen
- 

### Frustrations

- Energieverschwendung
- 

### Ideal Features

- Zeitgesteuerte Kühlung



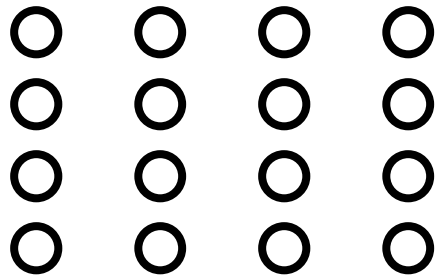
- GUI = Graphical User Interface = konkretes Layout der Benutzungsschnittstelle
  
- Wodurch wird ein GUI definiert?
  - Datendarstellung
  - Funktionsdarstellung
  - Bildschirmgestaltung
  - Dialogbeschreibung

- Wie werden Daten auf dem Bildschirm dargestellt?
  - UI-Daten
- Welche Hilfsfunktionen sind nötig, um die Berechnung von Funktionen sowie die Navigation zwischen Bildschirmen zu kontrollieren?
  - Navigations- und Hilfsfunktionen
- Wie werden Daten und Funktionen am Bildschirm angeboten?
  - Sichten
- Wie können NutzerInnen die Ausführung von Funktionen kontrollieren?
  - Dialoge

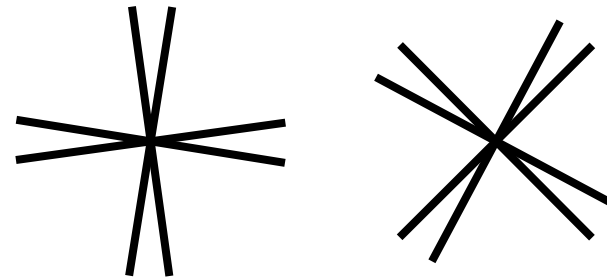
- Wie erkennen wir Zusammenhänge am Bildschirm?
  - Gestaltgesetze (z.B. Gesetz der Nähe, Abgeschlossenheit, Ähnlichkeit, parallele Bewegung)
- Wie lesen wir Text?
  - Textregeln (Zeit zum Lesen : Zeilenlänge und Abstand)
- Wie erwecken wir Aufmerksamkeit?
  - Kontraste (Form, Farbe, Bewegung)
- Welche Darstellungsmöglichkeiten gibt es?
  - Z.B. Listen, Grafiken, Text
- Hier muss die tatsächlich verfügbare Bildschirmgröße (z.B. PC oder Hand-Display) berücksichtigt werden

# Gestaltgesetze (1)

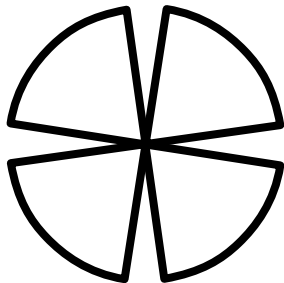
**Example A:**  
**Law of proximity**



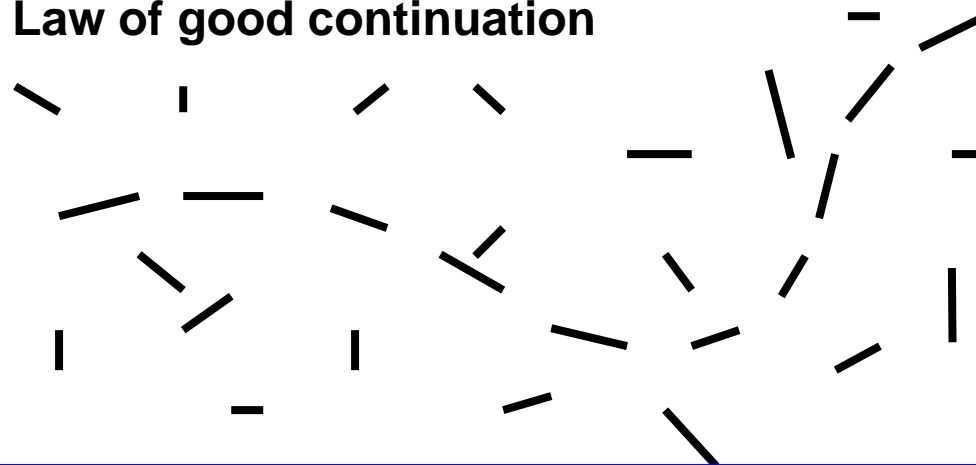
**Example B: Mill wheels**  
**Law of proximity**



**Example C:**  
**Law of closure**

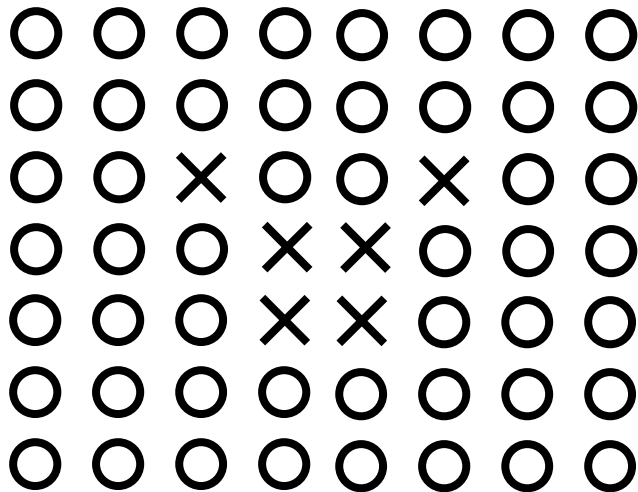


**Example D:**  
**Law of good continuation**

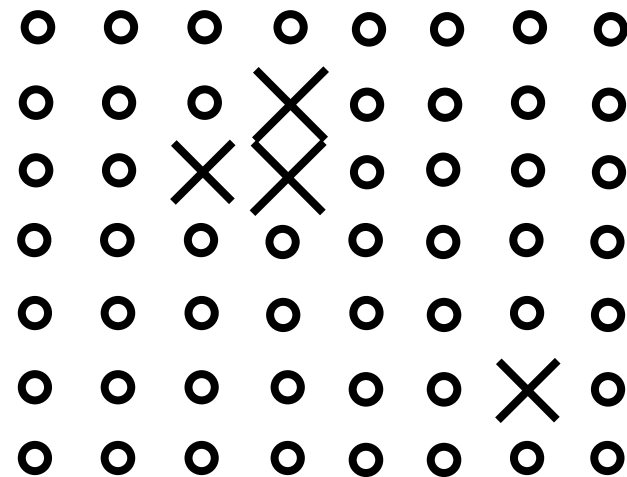


## Law of similarity

Example G



Example H

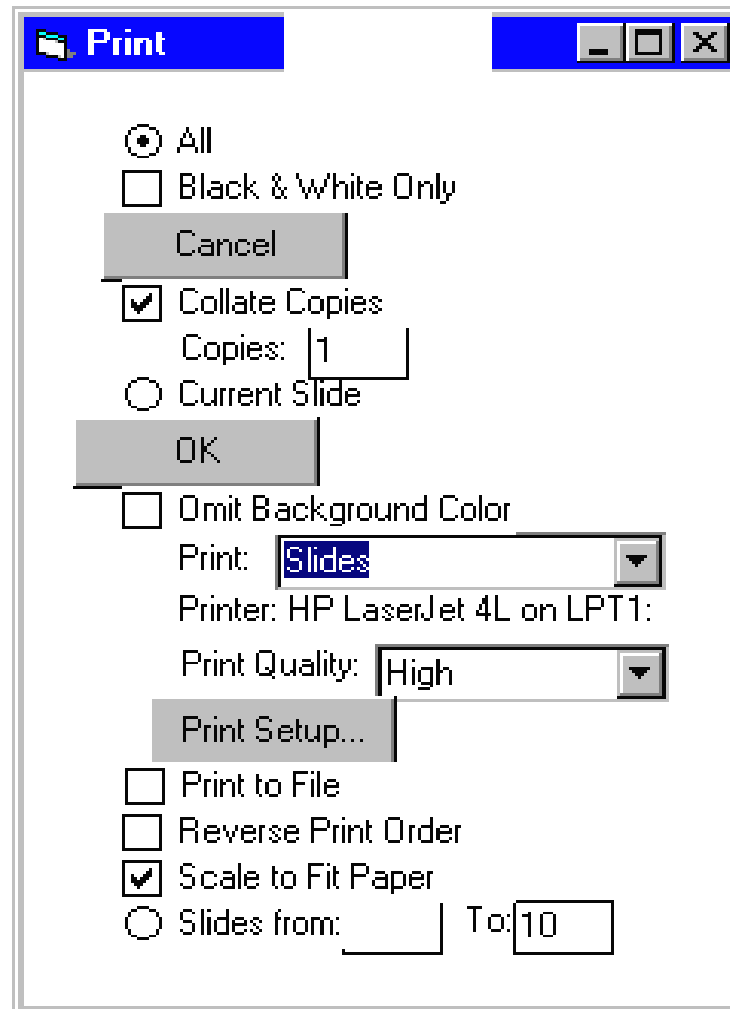


Analog für Bewegung: Law of parallel movement

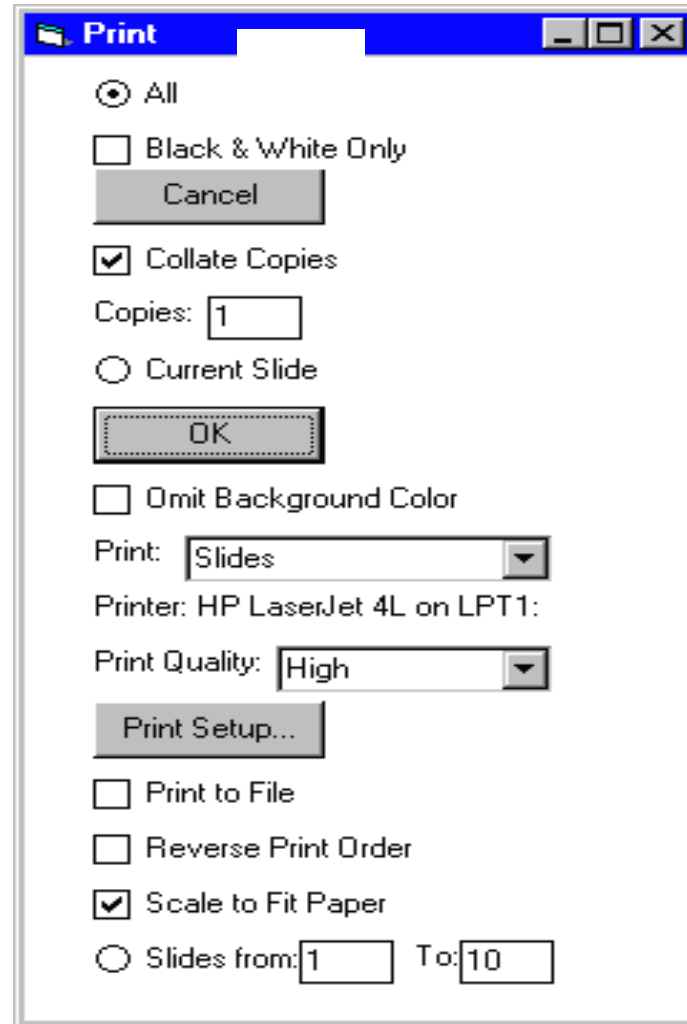


- Finden Sie für die folgenden Daten und Funktionen zum Drucken von Folien ein gutes Layout auf einem Bildschirm, so dass inhaltlich zusammengehörige Elemente leicht als zusammengehörig wahrgenommen werden können.
  - Druckername
  - Abbruch
  - Drucken in umgekehrter Reihenfolge
  - Druckqualitätseinstellung (Auswahl)
  - Druck auslösen
  - Skalierung auf Papiergröße
  - Auswahl der zu druckenden Folien
  - Weglassen der Hintergrundfarbe
  - Druckereinstellungen
  - Anzahl Kopien
  - Drucken in Datei
  - Nur Schwarz/Weißdruck

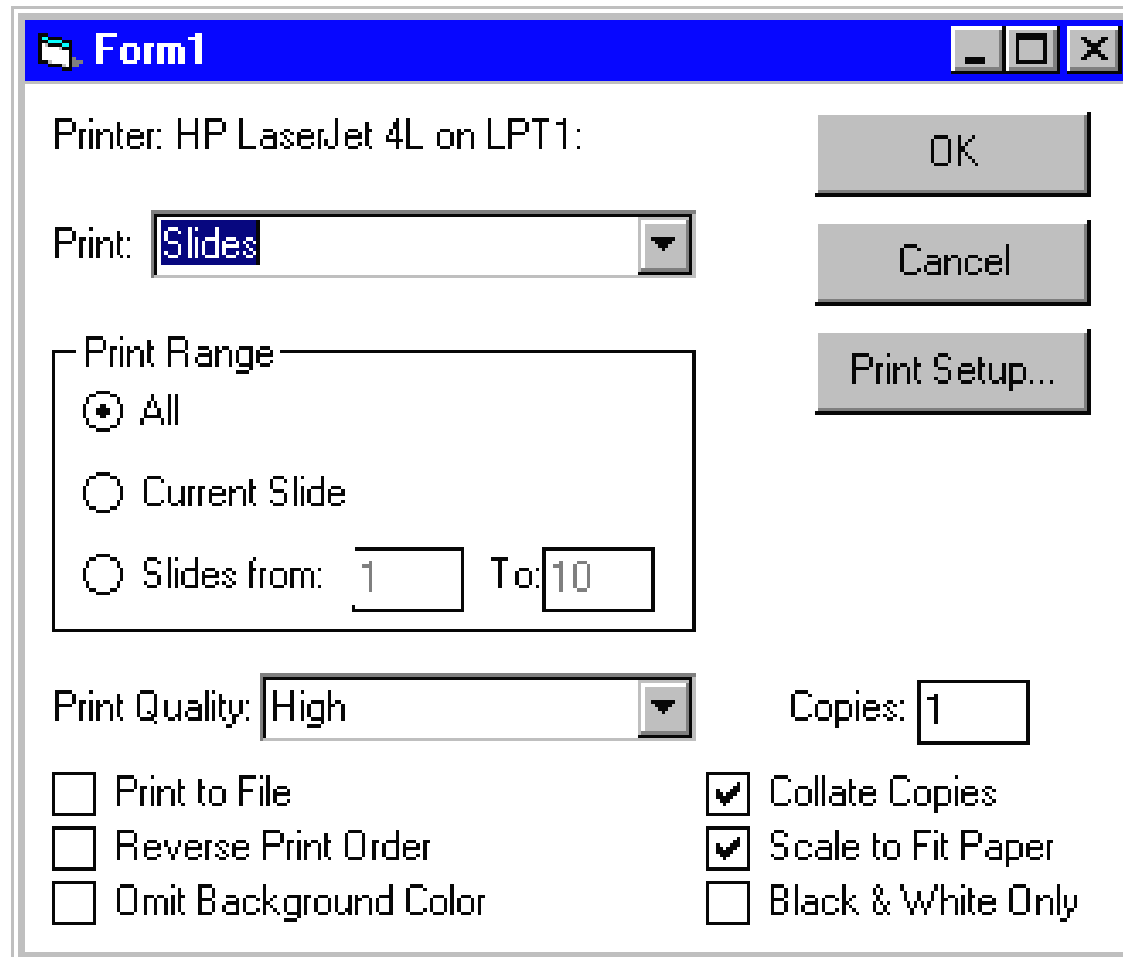
# Beispiel (1)



# Beispiel (2)



## Beispiel (3)



Form1

Printer: HP LaserJet 4L on LPT1:

Print: Slides

Print Range

☒ All

☐ Current Slide

☐ Slides from: 1 To: 10

Print Quality: High

Copies: 1

☐ Print to File

☐ Reverse Print Order

☐ Omit Background Color

☒ Collate Copies

☒ Scale to Fit Paper

☐ Black & White Only

OK

Cancel

Print Setup...



## ■ Funktionsarten

### • Semantische Funktionen

- Beschreiben die eigentlichen Datenänderungen im System (z.B. Berechnung, Abspeichern)
- Sind oft zusammengesetzt, z.B. „Erzeuge Dokument“ besteht aus „Neues Dokument anlegen“, „Eingabe“, Speichern“
- Oft **Hilfsfunktionen** nötig, die für NutzerInnen sichtbare Zwischenergebnisse bereitstellen

### • **Hilfsfunktionen:** Datenänderungen auf dem Bildschirm (z.B. Font, Texteingabe)

### • Suche

### • **Navigationsfunktionen** (z.B. Navigation zwischen Fenstern, Drag and Drop, Web Link)

# Darstellung von Funktionen

- Knöpfe, Checkbox,
- Menu-Auswahl, Shortcut, Icons
- Scroll-Bar
- Drag and Drop



[http://de.wikipedia.org/wiki/Kategorie:Grafische\\_Benutzeroberfl%C3%A4che](http://de.wikipedia.org/wiki/Kategorie:Grafische_Benutzeroberfl%C3%A4che)

- **Sichten konkretisieren Arbeitsbereiche** mit konkreten Angaben für
  - Datendarstellung
  - Funktionsauswahl und –darstellung
- Oft mehrere Sichten für einen Arbeitsbereich (abhängig von Bildschirmgröße)
- Sicht auch Bildschirm genannt

- Prototype, Folge von Mock-Ups



- Bei der GUI-Beschreibung sind viele Detailentscheidungen zu treffen
  
- Wichtig ist so viel festzulegen, dass die Sichten und Dialoge
  - alle Funktionen (insbes. Use Cases) unterstützen
  - **konsistent** sind (d.h. für gleiche Informationen gleiche Sichten)
  - **benutzungsfreundlich** sind (siehe später Gebrauchstauglichkeit)

- Lauesen S (2005) *User Interface Design - A software engineering perspective*, Addison-Wesley

- Die Entscheidungen auf der Interaktionsebene sind oft schwierig, da hier Mensch und Maschine „aufeinander treffen“.
- Hier wird sichergestellt, dass **die richtige Funktionalität benutzungsfreundlich** angeboten wird.
- UI-Struktur ermöglicht **frühe** Diskussion mit NutzerInnen (**ohne großen Aufwand!**)
- Bei Use Case Erstellung ohne UI-Struktur ist es schwierig, die geeignete Abstraktionsebene zu finden.

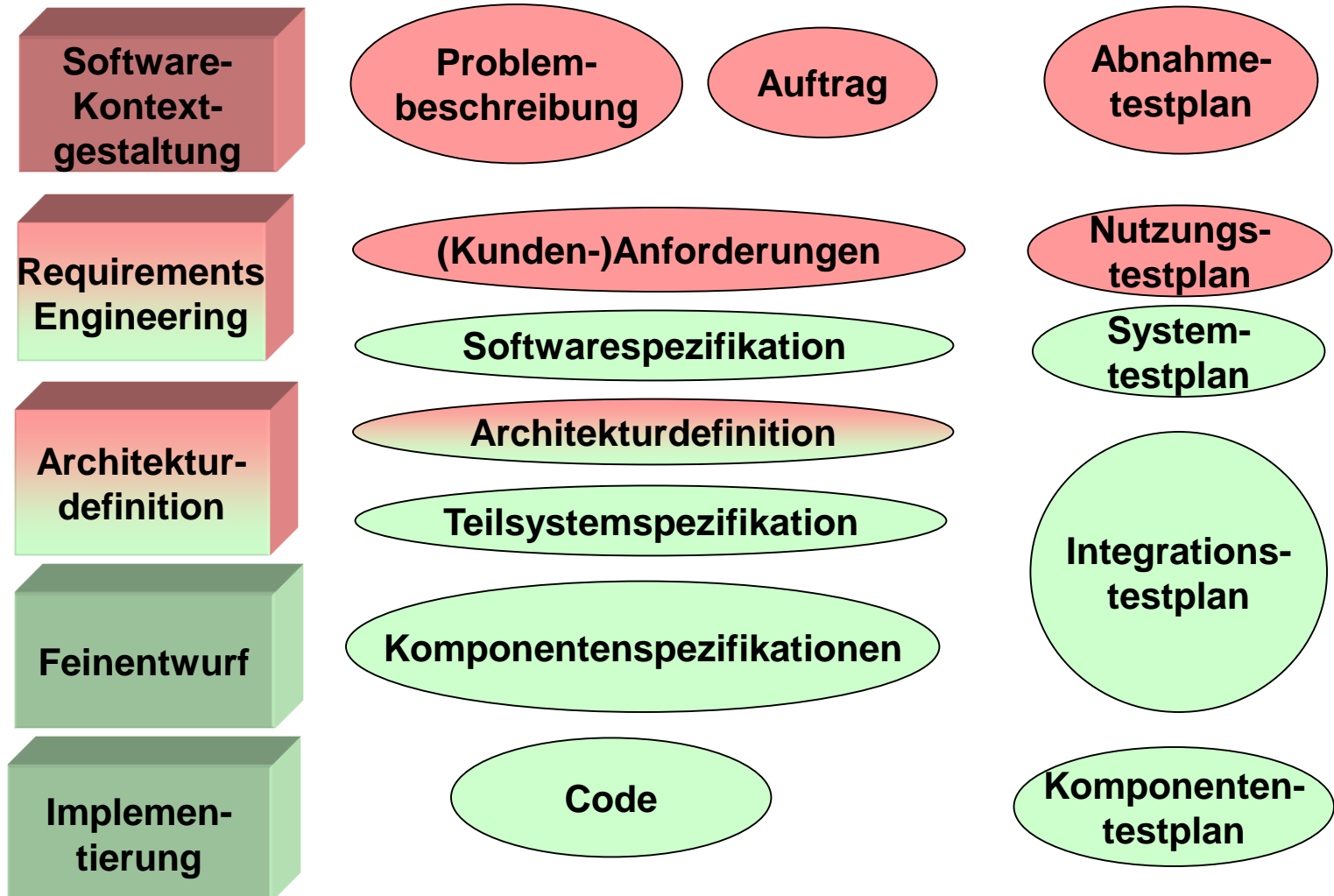
### 3.4. Dokumentenqualität

Einführung und Vorlage

Merkmale und Stilratgeber

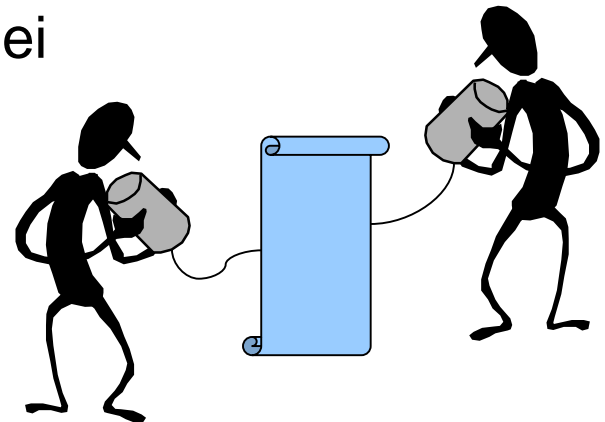
Perspektivenbasierte Inspektion

# Es gibt viele Dokumente für die Softwareentwicklung



# Kommunikation durch Dokumentation

- Die Beteiligten eines Softwareprojekts sind fast nie räumlich an einem Ort
- Direkte Kommunikation ist deshalb eher selten, insbesondere zwischen KundInnen (NutzerInnen) und EntwicklerInnen.
- Gemeinsames Verständnis muss durch **Dokumente** „gespeichert“ werden.
- Dokumente sind insbesondere wichtig bei
  - **Langlebigkeit** der Inhalte
  - **Rechtlicher Relevanz** der Inhalte
  - **Komplexität** der Inhalte
  - **Zugänglichkeit** für **viele** Beteiligte
- Dokumente beschreiben oft ein Modell der Software



## 1. Einleitung

### 1.1 Zweck

Wer hat das Dokument wie erstellt?

Wer soll das Dokument wozu lesen?

Für wen ist es wie verbindlich (Geltungsbereich)?

### 1.2 Zusammenfassung

Kerninhalte

### 1.3 Definitionen und Abkürzungen

Definition von Begriffen und Abkürzungen  
(inkl. Glossar)

## *1.4 Referenzen, Standards und Vorschriften*

Liste der referenzierten Dokumente,  
insbesondere Standards und Vorschriften

## *1.5 Überblick*

Inhalt und Aufbau des Dokuments

## **2. – X. Kerninhalte**

## **X+1. Zusammenfassung**

## **X+2. Anhang**



## 2. Beschreibung des Produktkontexts

### 2.1 *Zweck des Produktes*

zu erreichende Geschäftsziele

### 2.2 *Stakeholder*

Von dem Produkt Betroffene,  
daran Beteiligte (Rollen) und Interessierte

### 2.3 *Abläufe im Kontext*

Geschäftsprozesse, in denen das Produkt  
benötigt wird (inkl. Aufgaben der Rollen)

## 3. Systemanforderungen

### 3.1 Systemkurzbeschreibung

Systemverantwortlichkeiten

### 3.2 Architekturideen

### 3.3 Funktionale Anforderungen

Ablauf und Verhaltensbeschreibung (Use Cases und Systemfunktionen), Datenbeschreibungen, GUI

### 3.4 Nicht-funktionale Anforderungen

## 4. Projektanforderungen

### 3.1. Annahmen und Abhängigkeiten (inkl. Risiken)

### 3.2.2. Abnahme

- Standards geben im Wesentlichen eine **Kapitelgliederung** des Dokuments und eine **kurze Inhaltsbeschreibung** der einzelnen Kapitel vor
- Z.B. Anforderungen (<http://www.ieee.org>):
- ISO/IEC/IEEE 29148-2011 Systems and software engineering — Life cycle processes — Requirements engineering
- speziell: V-Modell (D), Dod-MIL-Std. 498 D Department of Defense (USA), European Space Agency (EU), PSS-05 Ministry of Defense (GB), INCOSE, VDI-VDE,...

Gut als Checkliste

.... sind abhängig von der Beantwortung der Frage: Wie hoch ist das **Risiko**, wenn die LeserInnen nicht die benötigten Information finden?

- Hoher **Aufwand zur Klärung von Fragen** für die LeserInnen
  - Z.B. wenn AutorInnen zeitlich, räumlich schwer zu erreichen
- Hohes **Risiko bei Auftreten von Fehlern** aufgrund von fehlender oder missverständlicher Information
  - hoher **Qualitätsanspruch** an die Software?
- Hohe **Änderungswahrscheinlichkeit**
  - Information wichtig, aber noch nicht fest

### 3.4. Dokumentenqualität

Einführung und Vorlage

Merkmale und Stilratgeber

Perspektivenbasierte Inspektion

- **Verständlich für alle!**
  - Eindeutig, vollständig, konsistent
  
- **Verständlich für die NutzerInnen des Dokuments**  
z.B. NutzerInnen der Anforderungen
  - Auftraggeber:
    - **korrekt**: Ist „Auftrag“ richtig wiedergegeben?
  - RE-IngenieurIn: interne Abstimmung bei Änderungen, Stand des RE-Prozesses
    - **gewichtet, nachvollziehbar, änderbar**
  - EntwicklerIn:
    - **umsetzbar**: Klare Vorgabe?
  - TesterIn:
    - **verifizierbar**: Testfälle ableitbar?

## ■ Nach ISO 29148

- **eindeutig** (d.h. nur eine mögliche Interpretation, klare Struktur, gute Überblickstabellen, keine Redundanz)
- **vollständig** (d.h. alle nicht-funktionalen Anforderungen, alle Systemreaktionen - auch auf ungültige Eingaben, explizite Kennzeichnungen von offenen Punkten)
- **konsistent** (keine Widersprüche, konsistente Terminologie)
- **notwendig** (d.h. nur "richtige" Anforderungen)
- **verifizierbar** (d.h. durch ein Verfahren kann Erfüllung geprüft werden)
- **gewichtet** bzgl. Wichtigkeit und Stabilität
- **umsetzbar**
- **nachvollziehbar** (d.h. Quelle der Anforderungen festhalten und eindeutige Identifikatoren)

# Beispiel: Unscharfe Formulierung

---

Sie öffnen also morgens das Schloss am Haupteingang?

Ja, habe ich Ihnen doch gesagt.

Jeden Morgen?

Natürlich.

Auch am Wochenende?

Nein, am Wochenende bleibt der Eingang zu.

Und während der Betriebsferien?

Da bleibt er natürlich auch zu.

Und wenn Sie krank sind oder Urlaub haben?

Dann macht das Herr X.

Und wenn auch Herr X ausfällt?

Dann klopft irgendwann ein Kunde ans Fenster, weil er nicht reinkommt.

Was bedeutet „morgens“? ...



# Beispiele für Anforderungen (1)

---

## Negativ

Das System ermöglicht die Eingabe von Film- und Schauspielerbewertungen. Es werden dann die *Gesamtbewertungen* ermittelt.

Problem: **Passivformulierung**. Wer ermittelt? Das System oder der Akteur?

## Positiv

Das System ermöglicht die Eingabe von Film- und Schauspielerbewertungen und ermittelt dann die *Gesamtbewertung* des Filmes.

# Beispiele für Anforderungen (2)

---

## Negativ

Die letzten 10 Filme und die *Gesamt-*  
*bewertungen* sollen angezeigt werden.

Problem: **Referenzen** „letzte“ (zeitliche oder in  
Reihenfolge), „und Gesamtbewertungen“  
(alle oder der letzten 10?),  
**Passivformulierung** (wer soll anzeigen?)

## Positiv

Das System soll bei Wiederaufruf der  
Funktion *Bewerten* die 10 zuletzt  
bearbeiteten Filme und deren  
*Gesamtbewertungen* anzeigen.

# Beispiele für Anforderungen (3)

---

## Negativ

Das System soll eine Warnung anzeigen, wenn ein Film 1 Jahr nicht ausgeliehen und mehrmals geändert wurde oder Daten fehlen.

Problem: „und/oder“-Schachtelung, „und“-Bedingungen gleichzeitig? Was bedeutet „soll“ (ist das optional)?

## Positiv

Das System zeigt eine Relevanzwarnung an, wenn ein Film 1 Jahr nicht ausgeliehen wurde. Es zeigt eine Überarbeitungswarnung an, wenn ein Film mehrmals geändert wurde oder Daten bei ihm fehlen.

# Beispiele für Anforderungen (4)

---

## Negativ

Das System soll mit der Esc-Taste die aktuelle Bearbeitung abbrechen.

Problem: „die aktuelle Bearbeitung“ ist **zu generell**.  
Gilt das wirklich für jede aktuelle Bearbeitung?  
Sollen dabei Zwischenstände gespeichert werden?

## Positiv

Wird während der Dateneingabe die Esc-Taste gedrückt,... Wird während einer Systemberechnung ...

In jedem Fall holt das System vor einem Abbruch des aktuellen Vorgangs eine Bestätigung von der/dem BenutzerIn ein.

- Ziel: Dokumente sind **leichter zu lesen** und somit leichter zu verstehen
- Unser Stilratgeber behandelt die **häufigsten Probleme**, projektspezifisch sind Ergänzungen sinnvoll
- Regeln sollten zu einem firmenspezifischen Stilratgeber zusammengefasst werden

- **Kurze Sätze** und **kurze Absätze** (max. ca. 7 Sätze), da das menschliche Kurzzeitgedächtnis begrenzt ist
- **Nur eine Aussage pro Satz** formulieren: 'und' vermeiden, Verschachtelte logische Zusammenhänge vermeiden
  - Pseudocode oder Entscheidungstabellen verwenden
- **Konsistente Terminologie:** Jargon vermeiden, Abkürzungen sparsam verwenden, Wortwiederholungen sind erwünscht!

- Offene Punkte kennzeichnen: „TBD“, „ToDo“
- Generalität vermeiden: klare Referenzen, Wörter wie „alle“ vermeiden
- Verbindlichkeit klar formulieren: „Muss, kann, soll“ etc. mit Bedacht verwenden
- Aktiv formulieren: „Das System ....“

# Beispiel: Stilratgeber für User Task

---

- Siehe User Task - Stilratgeber



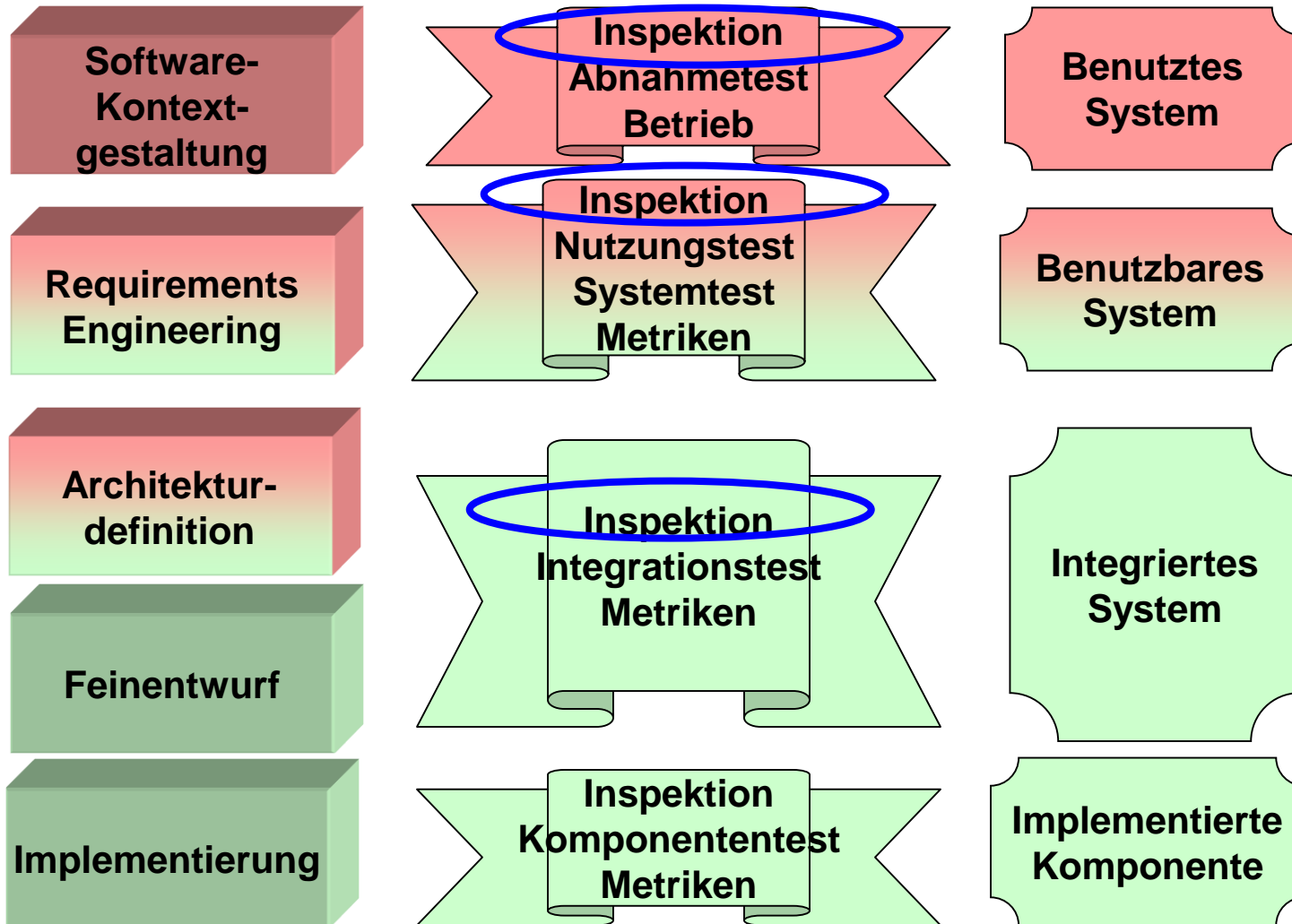
### 3.4. Dokumentenqualität

Einführung und Vorlage

Merkmale und Stilratgeber

Perspektivenbasierte Inspektion

# Aktivitäten und Ergebnisse der Entwicklung und Qualitätssicherung



# Wdh. Inspektionsverfahren

---

- Es gibt viele Verfahren, Software zu inspizieren.

1. die **Durchsicht** (der „Schreibtisch-Test“)
2. die **Stellungnahme**
3. das **Technische Review**
4. der **Walkthrough**
5. die **Design- and Code Inspections**

- **Gesprächsrunden** – vor allem zur Klärung der Anforderungen – werden ebenfalls als Reviews bezeichnet werden.

- Wir sprechen in diesem Fall von **Workshops**. Sie sind sehr sinnvoll, haben aber eine andere Zielsetzung als die Reviews.

## ■ Checklisten-basiertes Lesen

- schnellere, objektivere und wiederholbare Analyse
- wichtige Ressource eines Unternehmens, die den gegenwärtigen Stand von Erfahrungen reflektieren
- Für Entdeckung semantischer Mängel nicht so gut geeignet

## ■ Perspektiven-basiertes Lesen

- Familie von Techniken für unterschiedliche Ziele, unterschiedliche Artefakte, verschiedene Sichten auf Artefakte
- systematischer als Checkliste
- Besonders geeignet für Entdeckung semantischer Mängel
- *Siehe später*

- **Inspektions-Sichten** erlauben es, den InspektorInnen **unterschiedliche Aufgaben** zu geben, so dass deren Aufgabe **lebendiger** wird
- Inspektions-Sichten sollten **repräsentativ für die NutzerInnen der Dokumente** sein
- Sichten **ergänzen sich**
- **Beispiel für Inspektions-Sichten:**
  - Sicht der KundInnen
  - Sicht der EntwicklerInnen
  - Sicht der NutzerInnen

# Inspektions-Sichten ergänzen sich

## Fokussierung der InspektorInnen auf Stakeholder-Interessen

<b>Name</b>	Speichere Sitzposition
<b>Aktor</b>	FahrerIn, BeifahrerIn
<b>Ziel</b>	Aktor speichert Sitzposition für Fahrersitz
<b>Vorbedingungen</b>	keine
<b>Beschreibung</b>	Aktor betätigt Speichereingabe und gibt entsprechende Identifikation ein. System speichert aktuelle Fahrersitzposition unter entsprechender Identifikation Use Case beendet
<b>Ausnahmefälle</b>	keine
<b>Regeln</b>	keine
<b>Qualitätsanforderungen</b>	keine
<b>Eingänge</b>	Speichereingabe
<b>Ausgänge</b>	Gespeicherte Sitzposition
<b>Nachbedingungen</b>	Sitzposition gespeichert

Sicht KundIn:

nur FahrerIn darf  
Funktion ausführen

Sicht EntwicklerIn:

Was bedeutet  
„entsprechende  
Identifikation“

Sicht TesterIn:

Fehlende  
Eingangsvariable:  
aktuelle Position

- **Inspektions-Szenarien** sind eine weitere Hilfe für InspektorInnen
  - Nicht nur lesen, sondern auch gleich das Produkt „nutzen“
- Ein Inspektions-Szenario gibt die Erstellung einer **Abstraktion** anhand einer Anleitung vor
- Beantwortung einer Fragenliste während und nach Erstellung der **Abstraktion**
- **Z.B. geeignete Abstraktionen für Aufgabenbeschreibungen:**
  - z.B. Entwicklerperspektive: Domänendatendiagramm
  - z.B. UI-Entwicklerperspektive: UI-Struktur

- Inspektionsart, die
  - verschiedene **Inspektions-Sichten** mit einbezieht
  - Und **Inspektions-Szenarien** verwendet
- JedeR InspektorIn erhält ein Inspektions-Szenario für eine Inspektions-Sicht (beides zusammen wird **Perspektive** genannt)



# Beispiel: Perspektive Tester

---

- **Einführung:** Stellen Sie sich vor, Sie wollen das System testen. Deshalb entwickeln Sie einen Testplan und Testfälle auf Basis der Anforderungen. Für Sie ist ganz wichtig, dass die Anforderungen **verifizierbar** sind.
- **Anleitung:** Erstellen Sie zuerst eine **Matrix** der Testobjekte und zugeordneten Anforderungen
  - Erstellen Sie eine Hierarchie aller möglichen Testobjekte beginnend mit dem Gesamtsystem an der Spitze.
  - Listen Sie die Testobjekte in der obersten Zeile der Matrix.
  - Listen Sie die Anforderungen (Systemfunktionen und Qualitätsanforderungen) in der äußersten Spalte der Matrix.
  - Kreuzen Sie an, welche Anforderung sich auf welches Testobjekt bezieht.
- **Überprüfung**
  - Die Testobjekte ließen sich gut aus den Anforderungen ableiten
  - Die einzelnen Anforderungen ließen sich gut identifizieren aus dem Text
  - Die Zuordnung war immer klar.

- Qualität der Dokumente ist wichtig für die **NutzerInnen der Dokumente**
- Es sollten **klare Vorgaben** für Gliederung und Inhalte existieren
- Auf die **Formulierung** kommt es an!
- Die **Dokumentenqualität** muss dem Entwicklungskontext angemessen sein (Beteiligte, Umfang, Detailgrad...)
- **Inspektionen** sind auf alle Ergebnisse der Softwareentwicklung anwendbar => einfache Maßnahme, die auch Wissenstransfer unterstützt

# Literatur perspektivenbasierte Inspektion

---

- O. Laitenberger, Cost-effective Detection of Software Defects through Perspective-based Inspections; PhD Thesis in Experimental Software Engineering; Fraunhofer IRB Verlag; 2000.
- O. Laitenberger and JM DeBaud, An Encompassing Life-Cycle Centric Survey of Software Inspection, Journal of Systems and Software, 2000.
- T. Gilb and D. Graham, Software Inspection. Addison-Wesley Publishing Company, 1993
- V.R. Basili, S. Green, O. Laitenberger, F. Lanubile, F. Shull, S.Sorumgard and M.V.Zelkowitz, The Empirical Investigation of Perspective-Based Reading, Journal of Empirical Software Engineering, vol. 2, no. 1, pp.133-164, 1996

## 3.5. Analytische QS für SWE im Großen

- Analytische QS-Techniken für SWE im Kleinen sind genauso einsetzbar für SWE im Großen
  - Statische Analyse, Reviews, Testen
- Die Dokumente sind aber umfangreicher (d.h. mehr Dokumente und längere größere Dokumente).
- Mehr Planung und Dokumentation und Steuerung und Verwaltung von QS-Maßnahmen nötig, um Überblick zu behalten.

# Vorgehen Analytische QS

In größeren Teams  
pro Aktivität eigene Rolle

Ressourcen

Stabilität

**WER** führt  
Aktivitäten aus?  
**WAS** wird dafür  
benötigt?  
...

QS-Planung

Prüfspezifikation

**WAS** soll  
**WIE** (intensiv)  
**WIE** (mit welcher  
Prüfmethode)  
geprüft werden?

Prüfung,  
Protokollierung,  
Behebung

Welche  
**Qualität** hat  
der Prüfling?  
**Freigabe**  
möglich?  
Welche  
**Maßnahmen**  
bis zur  
Erreichung der  
geforderten  
Qualität nötig?

QS-Auswertung

QS-Verwaltung

- Ziel: Festlegung einer **Prüfstrategie**
- Prüfstrategie definiert **Prüfmethoden** und **Prüfaufwand** aufbauend auf einer **Kosten/Risiko-** Abschätzung (relativ zu **Qualitätszielen** )
  - **Kosten:**
    - **Direkte Fehlerfolgekosten** (bei KundIn durch Fehlerwirkung entstehende Kosten) +
    - **Indirekte Fehlerfolgekosten** (bei HerstellerIn durch Unzufriedenheit der KundIn entstehende Kosten) +
    - **Fehlerkorrekturkosten** (bei HerstellerIn durch Fehlerkorrektur entstehende Kosten, Korrektur = Suche+Behebung)
  - **Risiken:**
    - Reifegrad des Entwicklungsprozesses, Prüfbarkeit der Software (Dokumentation, Komplexität, Systemumgebung), Qualifikation der MitarbeiterInnen

# Fehlerkosten bei Hersteller

- Software Engineering strebt nach dem **Kostenminimum**. Dabei müssen auch **Folgekosten** und **Risiken** berücksichtigt werden.





## ■ Pro Prüfobjekt (Dokument oder Codeteil)

- Festlegung der Prüfmethode
- Festlegung des Abdeckungsgrades

## ■ Priorisierung der einzelnen Prüfungen

- Risiko (Eintrittswahrscheinlichkeit eines Fehlers) (z.B. häufig von KundIn genutztes Prüfobjekt, intern kritisches Prüfobjekt, komplexes Prüfobjekt)
- Fehlerkosten (Für KundIn: Schaden beim Auftreten bzw. Zufriedenheit; Für HerstellerIn: Größe des Fehlerbehebungsaufwandes)
- Priorität der Anforderungen

## ■ **QS-Konzept** [IEEE 829-1998]

- Einführung
- Qualitätsziele (z.B. zu erreichende Zuverlässigkeit)
- Prüfobjekte und Festlegung, welche Funktionen und Eigenschaften geprüft werden
- Prüfstrategien
- Kriterien für Prüfungsabnahme, Prüfungsabbruch (z.B. bei unreifem Prüfling), Prüfungsfortsetzung
- Prüfungsdokumentation (gewährleistet Reproduzierbarkeit)
- Prüfumgebung (gewährleistet Reproduzierbarkeit)
- Prüfaufgaben und Verantwortliche
- Zeitplanung, Risiken
- Freigabe des QS-Konzepts

- **Ausführung der Prüfung**
  - Ausführung der Prüfungen (manuell, automatisiert, semi-automatisiert)
  
- **Protokollierung**
  - Dokumentation der ausgeführten Schritte und der Ergebnisse
  - Vergleich von IST und SOLL-Ergebnissen
  
- **Behebung**
  - Behebung der Fehlerzustände und Prüfung, ob Fehlerwirkung nicht mehr auftritt

- Überwachung des **Fortschritts der Prüfkativitäten** anhand von Metriken
  - Fehlerfindungsrate: Nimmt die Fehlerfindungsrate zu oder ab?
  - Überdeckungsgrad
- Prüfzyklussteuerung: auf **Planabweichungen** reagieren
  - Anpassung der QS-Pläne
  - Einschränkung der durchzuführenden Prüfungen (z.B. eingeschränkte Testfallmenge, Einschränkung der Varianten eines Testfalls)
- **Dokumentation und Kommunikation** der Änderungen

- Sichert Verfolgbarkeit der Prüfergebnisse (inklusive Fehlermanagement, siehe Kapitel 2.2.)
- Stellt durchgeführte Prüfungen für Wartungsprüfungen sowie Weiterentwicklungsprüfungen zur Verfügung
- Benötigt **QS-Management-Werkzeuge** sowie **Dateivergleichs-Werkzeuge** (um Prüfergebnisse zu vergleichen)
- Ermöglicht damit **Regressionsprüfungen**
  - Prüfungen werden kontinuierlich nach Änderungen durchgeführt
  - Oft mit Continuous Build verbunden

- **Grundsatz 1**

Vollständiges Prüfen ist nicht möglich.

- **Grundsatz 2**

Mit Prüfen wird die Anwesenheit von Fehlerzuständen (statische Prüfung) und Fehlerwirkungen (Dynamische Prüfung = Testen) nachgewiesen. Dass keine Fehlerzustände im Prüfobjekt vorhanden sind, kann durch Prüfen nicht gezeigt werden.

»Program testing can be used to show the presence of bugs, but never to show their absence!«? Edsger Dijkstra

# Grundsätze der analytischen QS (2)

---

## ■ Grundsatz 3

Prüfen ist keine späte Phase in der Softwareentwicklung, es soll damit so früh wie möglich begonnen werden. Durch frühzeitiges Prüfen (z.B. Reviews) werden früher Fehler(zustände) erkannt und somit Kosten gesenkt.

- Siehe Ariane 5

## ■ Grundsatz 4

Fehlerzustände sind in einem Prüfobjekt nicht gleichmäßig verteilt, vielmehr treten sie gehäuft auf. Dort wo viele Fehlerzustände/-wirkungen nachgewiesen wurden, finden sich vermutlich auch noch weitere.

# Grundsätze der analytischen QS (3)

---

- **Grundsatz 5**

Prüfungen genau zu wiederholen, bringt keine neuen Erkenntnisse. Instrumente wie Testfälle sind zu prüfen, zu aktualisieren und zu modifizieren.

- **Grundsatz 6**

Prüfen ist abhängig vom Umfeld. Sicherheitskritische Systeme sind intensiver zu prüfen als beispielsweise der Internetauftritt einer Einrichtung.

- **Grundsatz 7**

Ein System ohne Fehlerwirkungen bedeutet nicht, dass das System auch den Vorstellungen der späteren NutzerInnen entspricht.

- Siehe Validierung vs. Verifikation



- Analytische QS muss **die ganze Softwareentwicklung begleiten**
- **Prüfplanung, insbesondere Testspezifikation** sollte parallel zu Requirements Engineering und Entwurf durchgeführt werden

- **Anforderungen** sind die **Grundlagen** für das ganze SWE-Projekt
- Anforderungen sind sehr gut zu dokumentieren
- **Aufgabenorientierte Anforderungsbeschreibung** hilft die Welt der NutzerInnen zu verstehen
- Dokumentenqualität wird durch **Templates und Stilratgeber** unterstützt
- **Frühe und kontinuierliche Prüfung** aller Artefakte wichtig