

Einführung in Software Engineering

Barbara Paech, Marcus Seiler

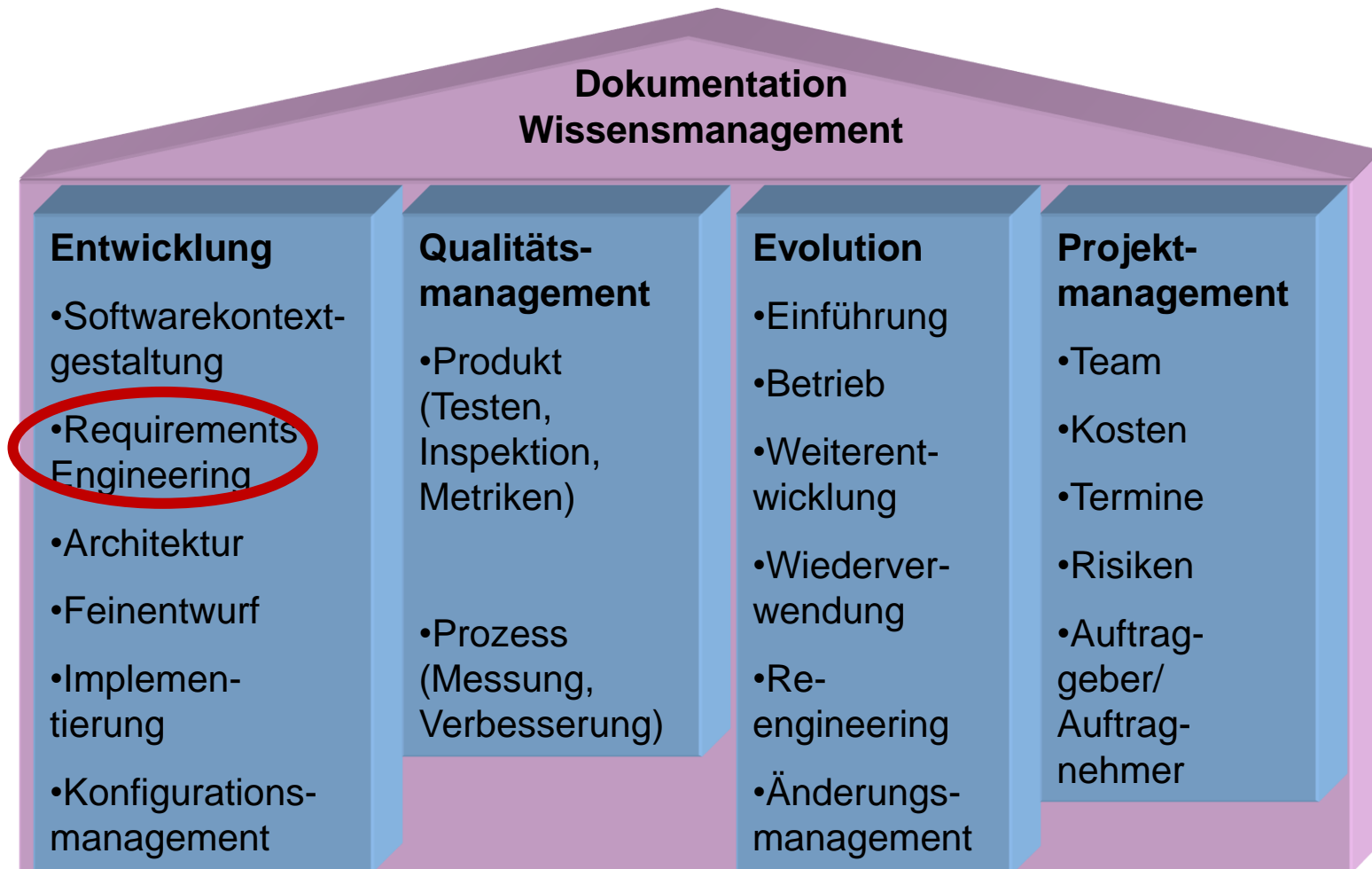
Institute of Computer Science
Im Neuenheimer Feld 326
69120 Heidelberg, Germany
<http://se.ifi.uni-heidelberg.de>
paech@informatik.uni-heidelberg.de



RUPRECHT-KARLS-UNIVERSITÄT HEIDELBERG



Aufgabenbereiche des Software Engineering



5. Ergänzungen zu Anforderungen (1. Teil)

Bei SWE im Großen gibt es viele Techniken, um Anforderungen zu erfassen und deren Qualität zu sichern. Die folgenden Techniken sind als Ergänzung hilfreich.

- 5.1. Gebrauchstauglichkeit (Usability)
- 5.2. Qualitätssicherung mit KundInnen
- 5.3. Use Cases
- 5.4 Qualitätsanforderungen (inkl. NFR-Test)
- 5.5. Anforderungserfassung und -spezifikation

5.1. Usability

Definition und Dialogprinzipien

Beschreibung

Was ist Usability?

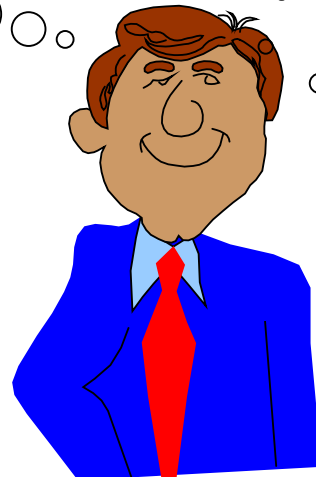
Ergonomie

Benutzungs-
freundlichkeit

MMI

(Man
Machine
Interface)

**Gebrauchs-
tauglichkeit**



HCI
(Human-
Computer-
Interaction)

Usability: Eine Summe von Faktoren

- Das Ausmaß, in dem ein Produkt durch bestimmte NutzerInnen in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele **effektiv**, **effizient** und **zufriedenstellend** zu erreichen (ISO 9241-110)

+ Effektivität

Welchen Teil der „Aufgabe“ unterstützt die Software?

-> Funktionalität, ...

+ Effizienz

Inwiefern wird der Zeitaufwand für die „Aufgabe“ durch die Software minimiert?

-> Performanz, Fehlertoleranz, Erlernbarkeit...

+ Zufriedenheit

Inwiefern werden die Bedürfnisse der NutzerInnen durch die Software befriedigt?

-> Attraktivität, Erwartungskonformität, ...

= Usability

Beurteilen Sie die Usability der folgenden Dosenöffner



Dosenöffner Nr. 1



Dosenöffner Nr. 2



Dosenöffner Nr. 4



Dosenöffner Nr. 5



Dosenöffner Nr. 3

Beurteilen Sie die Usability der verschiedenen Dosenöffner für folgende Personen.

Luca und Anna sind 21 Jahre alt. Sie wollen mit ihrem Rucksack per Interrail 6 Wochen durch ganz Europa fahren. Da ihr Geldbeutel nicht ausreicht, um immer Essen zu gehen, soll auch ein Dosenöffner mit ins Gepäck.

Frau Müller (43 Jahre alt) ist Köchin in der Kantine von SAP. Obwohl sie viel frisch zubereitet, muss sie ab und zu eine Dose öffnen. Sie ist auf der Suche nach einem geeigneten Dosenöffner.

Herr Freitag (72 Jahre alt) ist Linkshänder. Seit er Arthrose hat, fallen ihm feinmotorische Tätigkeiten immer schwerer, deshalb möchte er einen neuen Dosenöffner kaufen.

Usability: NutzerInnen, ihre Ziele, ihr Kontext

Das Ausmaß, in dem ein Produkt durch bestimmte **NutzerInnen** in einem bestimmten **Nutzungskontext** genutzt werden kann, um bestimmt **Ziele** effektiv, effizient und zufriedenstellend zu erreichen (ISO 9241-110)



**Kenntnis der
Ziele**




Fokus auf NutzerInnen



**Berücksichtigung
des Kontexts**



Welche Eigenschaften machen ein Software-Produkt „usable“? Nennen Sie zwei Eigenschaften.



**... wenn ich Software
auch ohne Training
bedienen kann!**

[DIN/EN ISO 9241.110]



Aufgabenangemessenheit:

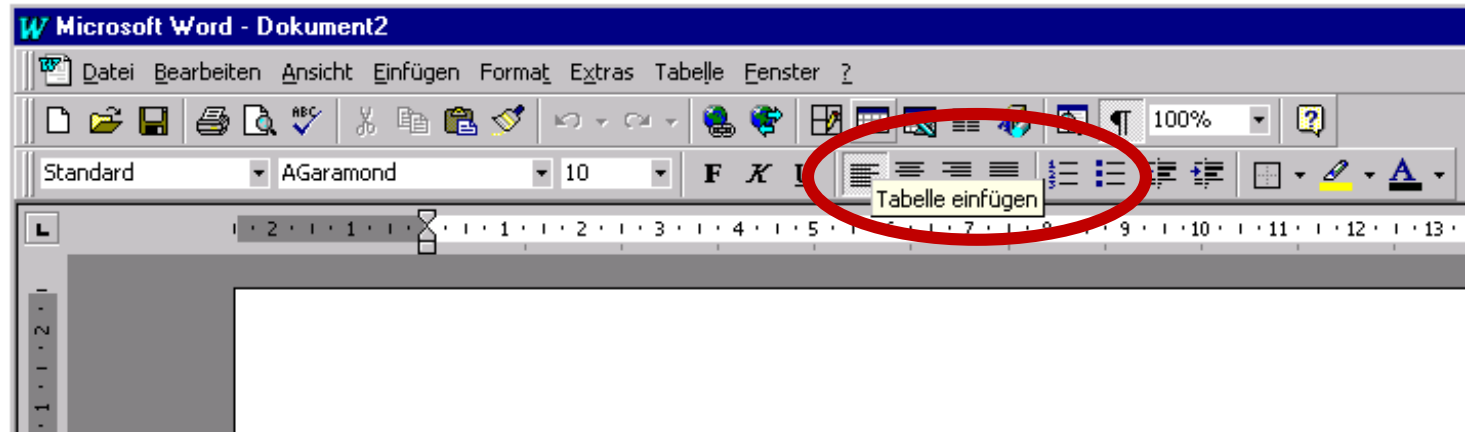
- Das Computerprogramm unterstützt die NutzerInnen bei der Erledigung ihrer Aufgaben.
- Es ist dabei eine Hilfe und kein unnötiges Übel, das die Arbeit erschwert oder umständlicher macht.

Gegenbeispiele:

- Texte und Daten müssen wiederholt eingegeben werden.
- NutzerIn muss Eingaben und Dialogschritte machen, die eigentlich unnötig sind.

Selbstbeschreibungsfähigkeit

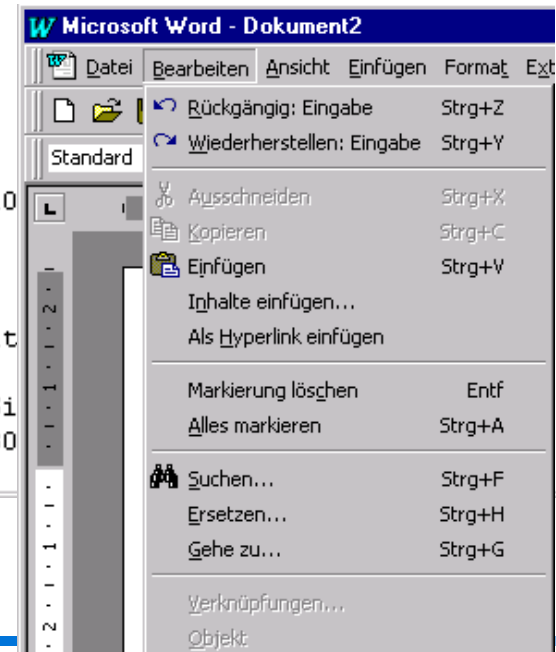
- NutzerInnen **wissen jederzeit**, was der Computer gerade macht, und welche Eingabe/Reaktion der Computer als nächstes erwartet.



Steuerbarkeit

- NutzerInnen können die Abfolge der Arbeitsschritte weitgehend **selbst bestimmen**.
- Wenn es die Arbeitssituation erfordert, können sie die Arbeit am Computer **unterbrechen** und ohne Verlust wieder aufnehmen.

```
baumert@knecht [~] >>lpq -Pbinac
Printer: lw-fl-ks-sek@info-f 'Laserwriter (HP4M+ with Duplex) FL10
Queue: 1 printable job
Server: pid 17515 active
Unspooler: pid 17548 active
Status: processing 'dfA5llinfo-f', size 138070, format 'f', IF filt
Filter_status: Rendering Page 1 ...
Rank  Owner/ID          Class Job Files          Si
active baumert@info-f+511      A    511 /tmp/BAUMERT..Daa3d 1380
baumert@knecht [~] >>
```



Erwartungskonformität

- Das System ist konsistent und entspricht den Merkmalen der NutzerInnen, z.B. den Kenntnissen aus dem Arbeitsgebiet, ihrer Ausbildung und ihrer Erfahrung sowie den allgemein anerkannten Konventionen.

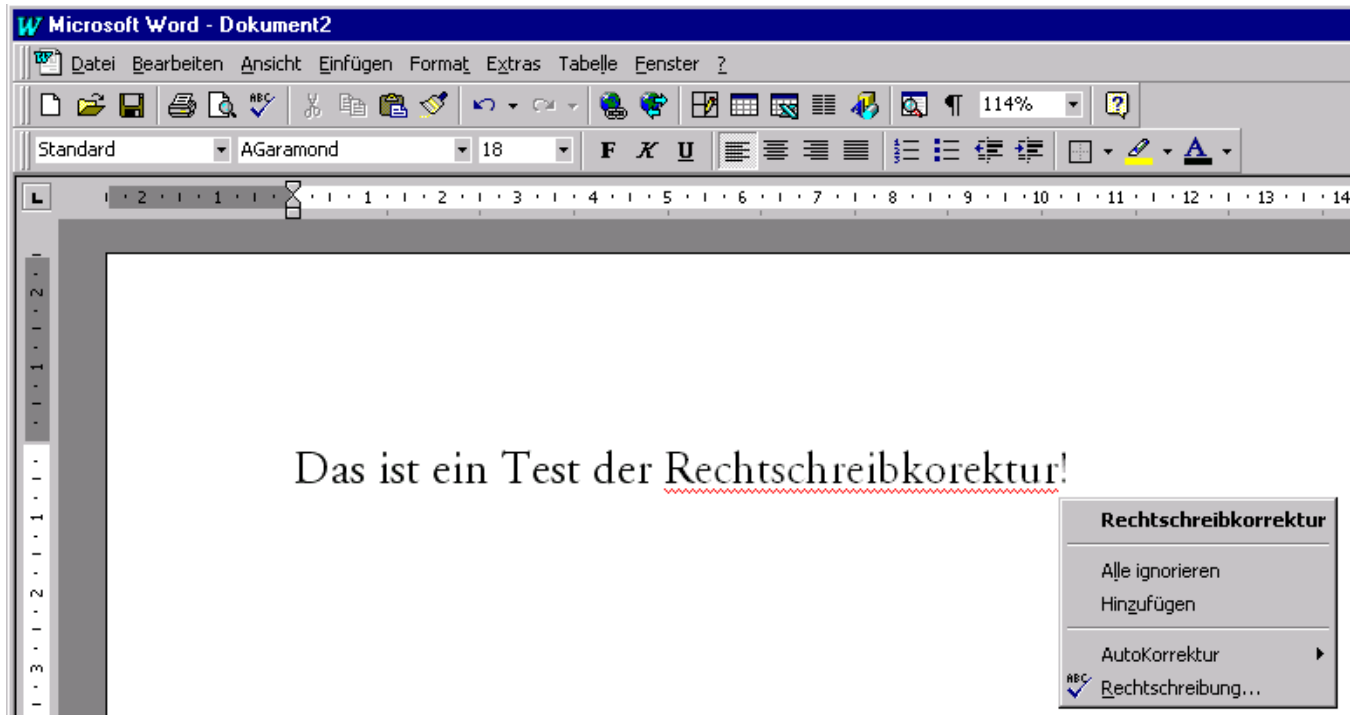


F1 = Hilfe

ESC = Abbrechen

Fehlertoleranz

- Das beabsichtigte Arbeitsergebnis kann trotz erkennbar fehlerhafter Eingaben mit keinem oder nur mit geringem Korrekturaufwand durch die NutzerInnen erreicht werden.



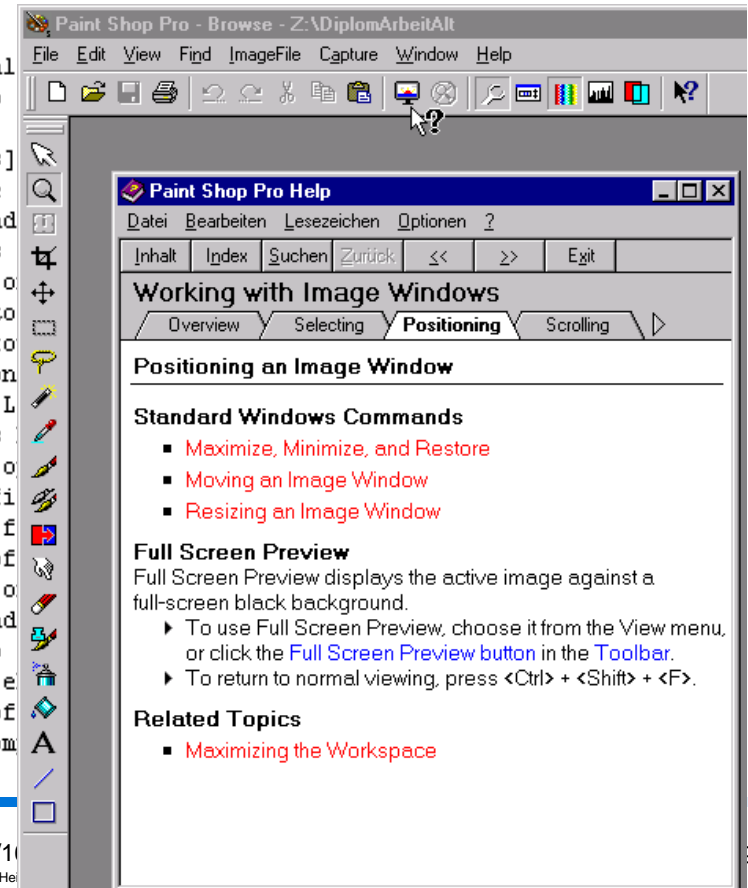
Individualisierbarkeit

- Das Dialogsystem lässt Anpassungen an die Erfordernisse der Arbeitsaufgabe, individuelle Vorlieben der NutzerInnen und deren Nutzungsfähigkeiten zu.

Lernförderlichkeit

- Die NutzerInnen werden beim Erlernen des Dialogsystems unterstützt und angeleitet.

```
baumert@knecht [~] >>zip -h
Copyright (C) 1990-1996 Mark Adler, Richard B. Wal
Onno van der Linden and Kai Uwe Rommel. Type 'zip
Zip 2.1 (April 27th 1996). Usage:
zip [-options] [-b path] [-t mmdyyy] [-n suffixes]
The default action is to add or replace zipfile
can include the special name - to compress stand
If zipfile and list are omitted, zip compresses
-f freshen: only changed files -u update: o
-d delete entries in zipfile -m move into
-k force MSDOS (8+3) file names -g allow gro
-r recurse into directories -j junk (don
-O store only -l convert L
-l compress faster -9 compress
-q quiet operation -v verbose o
-c add one-line comments -z add zipfi
-b use "path" for temp file -t only do f
-@ read names from stdin -o make zipf
-x exclude the following names -i include o
-F fix zipfile (-FF try harder) -D do not ad
-A adjust self-extracting exe -J junk zip
-T test zipfile integrity -X eXclude e
-y store symbolic links as the link instead of
-e encrypt -n don't com
```



- Lesen Sie die folgende kleine Geschichte. Welche der vorgestellten Dialogprinzipien scheint hier verletzt zu sein?
- Frau Neuhaus hat sich ein Haus gekauft und möchte eine kleinere Mülltonne als der Vorbesitzer. Deshalb ruft sie bei der Stadtverwaltung an, wo sie mit Herrn Meier, dem zuständigen Sachbearbeiter, verbunden wird. Was die neue Tonne koste und wann sie geliefert werde, will sie wissen. Freundlich weist Herr Meier sie darauf hin, dass er zuerst den laufenden Vorgang auf dem Computer abschließen muss. Erst dann kann er auf die benötigten Daten zugreifen. Auf Korrektheit kann der laufende Fall nicht mehr kontrolliert werden, denn zu lange will Herr Meier die Kundin nicht warten lassen, schließlich bemüht sich die Verwaltung intensiv um Bürgerfreundlichkeit. Seitdem steigt die Zahl der Widersprüche gegen Bescheide. Nun möchte Herr Meier die Fragen von Frau Neuhaus beantworten. "Ihre Kundennummer, bitte." Die hat Frau Neuhaus nicht parat; sie sucht in ihren Unterlagen. Als sie ihre Kundennummer findet und durchgibt, erkennt Herr Meier sofort, dass sie diese Nummer nicht beibehalten kann, weil das neue Haus in einem anderen Stadtteil liegt. Also muss er einen "neuen Kunden" anlegen. Alle erforderlichen Daten über Frau Neuhaus müssen nochmals in die neue "elektronische Fallakte" eingegeben werden. Dazu muss Herr Meier zuerst das Programm wechseln. Anschließend diktiert Frau Neuhaus und Herr Meier tippt ein. Nachdem alle Daten eingegeben sind, bestätigt Herr Meier diese und erhält die Fehlermeldung "Falsche Eingabe!", die er mit "OK" quittiert. Jetzt kann er den Fehler korrigieren.

- Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität, Fehlertoleranz, Individualisierbarkeit, Lernförderlichkeit
- Frau Neuhaus hat sich ein Haus gekauft und möchte eine kleinere Mülltonne als der Vorbesitzer. Deshalb ruft sie bei der Stadtverwaltung an, wo sie mit Herrn Meier, dem zuständigen Sachbearbeiter, verbunden wird. Was die neue Tonne koste und wann sie geliefert werde, will sie wissen. Freundlich weist Herr Meier sie darauf hin, dass er zuerst den laufenden Vorgang auf dem Computer abschließen muss. Erst dann kann er auf die benötigten Daten zugreifen. Auf Korrektheit kann der laufende Fall nicht mehr kontrolliert werden, denn zu lange will Herr Meier die Kundin nicht warten lassen, schließlich bemüht sich die Verwaltung intensiv um Bürgerfreundlichkeit. Seitdem steigt die Zahl der Widersprüche gegen Bescheide. Nun möchte Herr Meier die Fragen von Frau Neuhaus beantworten. "Ihre Kundennummer, bitte." Die hat Frau Neuhaus nicht parat; sie sucht in ihren Unterlagen. Als sie ihre Kundennummer findet und durchgibt, erkennt Herr Meier sofort, dass sie diese Nummer nicht beibehalten kann, weil das neue Haus in einem anderen Stadtteil liegt. Also muss er einen "neuen Kunden" anlegen. Alle erforderlichen Daten über Frau Neuhaus müssen nochmals in die neue "elektronische Fallakte" eingegeben werden. Dazu muss Herr Meier zuerst das Programm wechseln. Anschließend diktiert Frau Neuhaus und Herr Meier tippt ein. Nachdem alle Daten eingegeben sind, bestätigt Herr Meier diese und erhält die Fehlermeldung "Falsche Eingabe", die er mit "OK" quittiert. Jetzt kann er den Fehler korrigieren.

5.1. Usability

Definition und Dialogprinzipien

Beschreibung

- Qualität des GUI wird durch **Nutzbarkeitsanforderungen (usability requirements)** beschrieben
- Kann aus Sicht des Auftraggebers oder aus Sicht des Auftragnehmers formuliert werden.
 - Je nachdem ist das Risiko für **Auftraggeber (G)** oder **Auftragnehmer (N)** höher
- Anforderung muss messbar sein (siehe auch Qualitätsanforderungen später)

Wie beschreibe ich Usability?

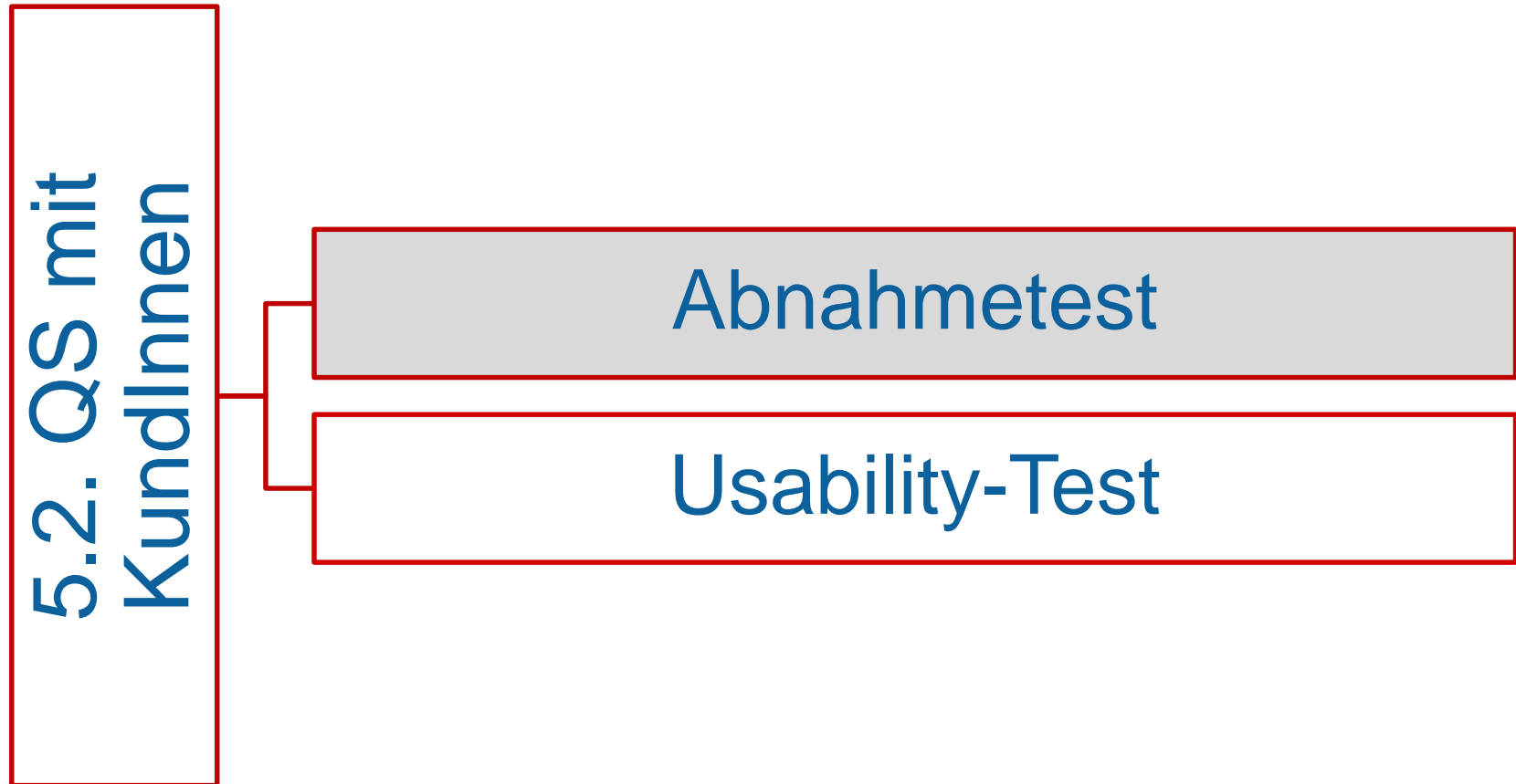
- **Performanzorientiert** **Höheres Risiko für [N]**
 - Unerfahrene NutzerInnen sollen Aufgabe X in 15 Minuten durchführen können, erfahrene NutzerInnen sollen das in 2 Minuten können
- **Fehlerorientiert** **[N]**
 - 80% der NutzerInnen sollen Aufgabe X fehlerfrei durchführen können
- **Prozessorientiert** **[G]**
 - Es sind 3 Prototypen zu entwickeln, jeder muss auf Usability getestet werden
- **Subjektorientiert** **[N]**
 - 80% der NutzerInnen sollen das System leicht erlernbar einschätzen
- **Eingabeorientiert** **[G]**
 - Eingabe des Kundenauftrags muss mit 5 Tastenanschlägen möglich sein (keine Verwendung der Maus)
- **Entwurfsorientiert** **[G]**
 - Die in X gezeigten Bildschirmmasken sind zu verwenden
- **Richtlinienorientiert** **[G]**
 - Der MS-StyleGuide ist zu verwenden. Menüs dürfen höchstens eine Tiefe von 3 haben

- Die Nutzersicht fokussiert darauf, das Produkt
 - Für **bestimmte Ziele**
 - In einem **bestimmten Nutzungskontext**
 - Effektiv, Effizient, und zufriedenstellend (also **subjektiv!**) nutzen zu können.

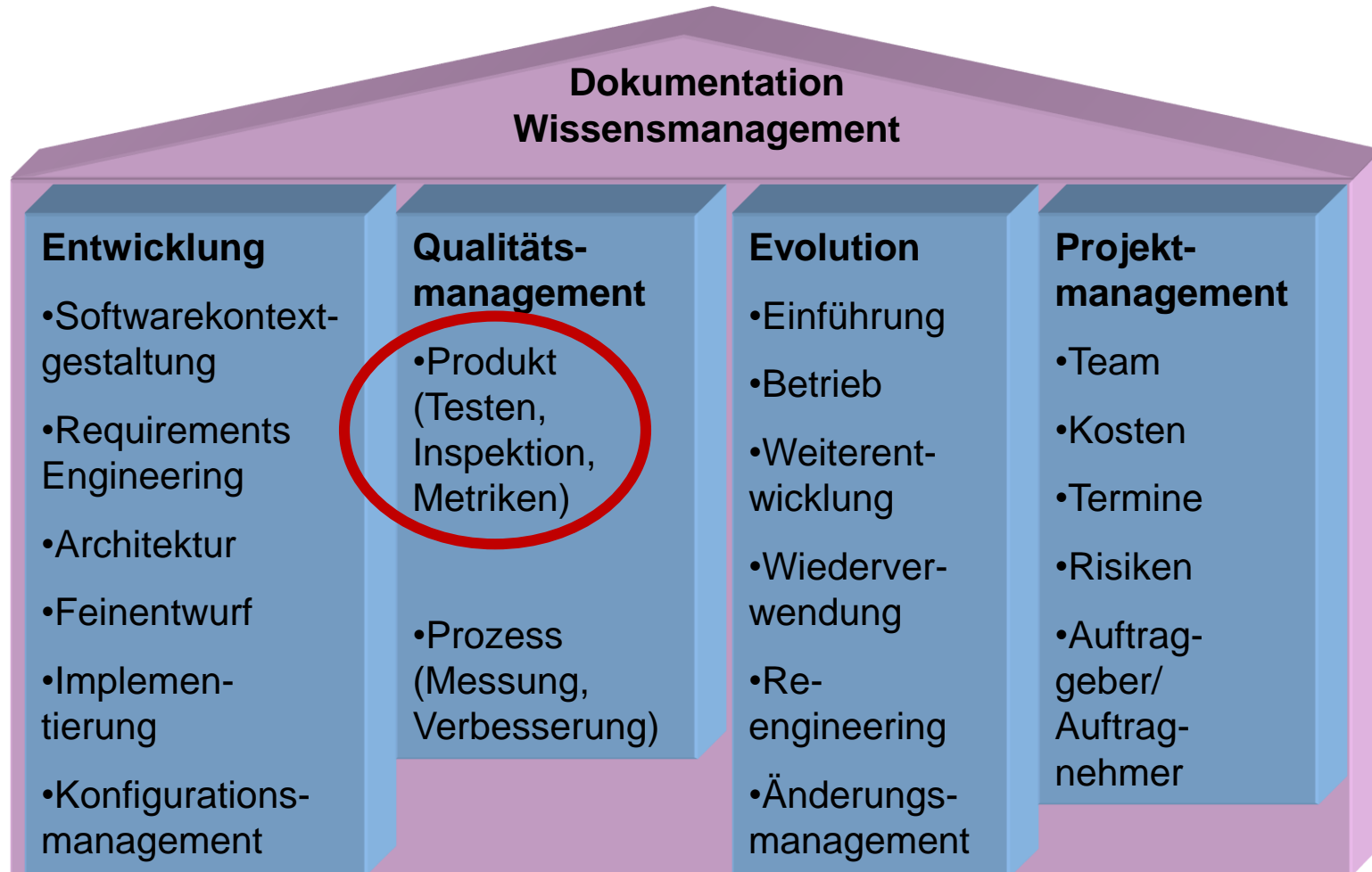
- **Dialogprinzipien** sind die Grundlage für gute Usability

- Usability-Anforderungen müssen **messbar** formuliert sein

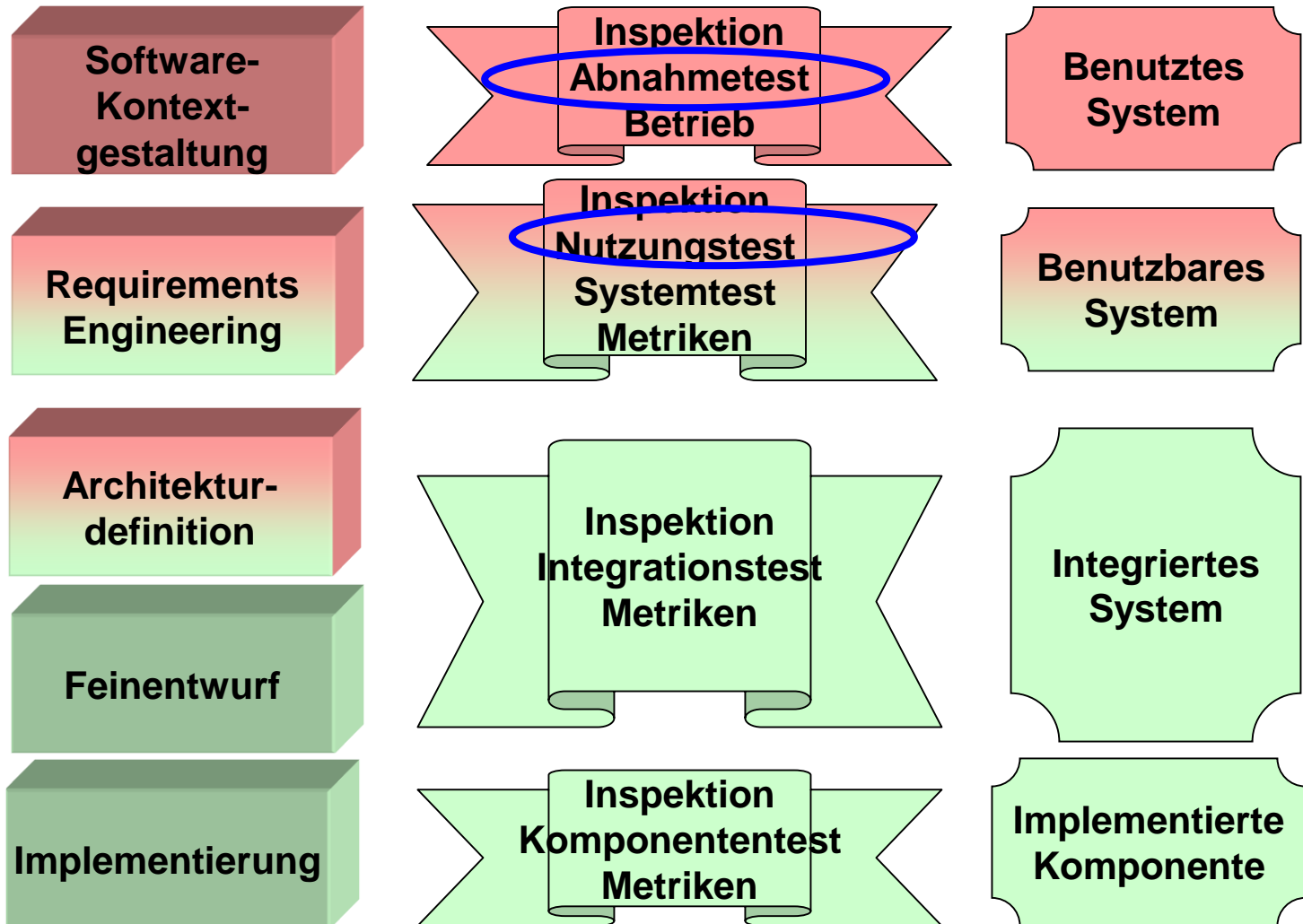
- S. Lauesen, *Requirements Engineering*, Addison-Wesley, 2002
- DIN/EN ISO 9241, *Ergonomie der Mensch-System-Interaktion*, 2011 (mehrere Teile Neubearbeitet)
- L. Constantine and L. Lookwood, *Software For Use*, ACM Press, 1999
- W. Dzida et al., *Gebrauchstauglichkeit von Software*, Schriftreihe der Bundesanstalt für Arbeitsschutz und Arbeitsmedizin, 2001
- S. Lauesen, *User Interface Design – A software engineering perspective*, Addison Wesley, 2005



Aufgabenbereiche des Software Engineering

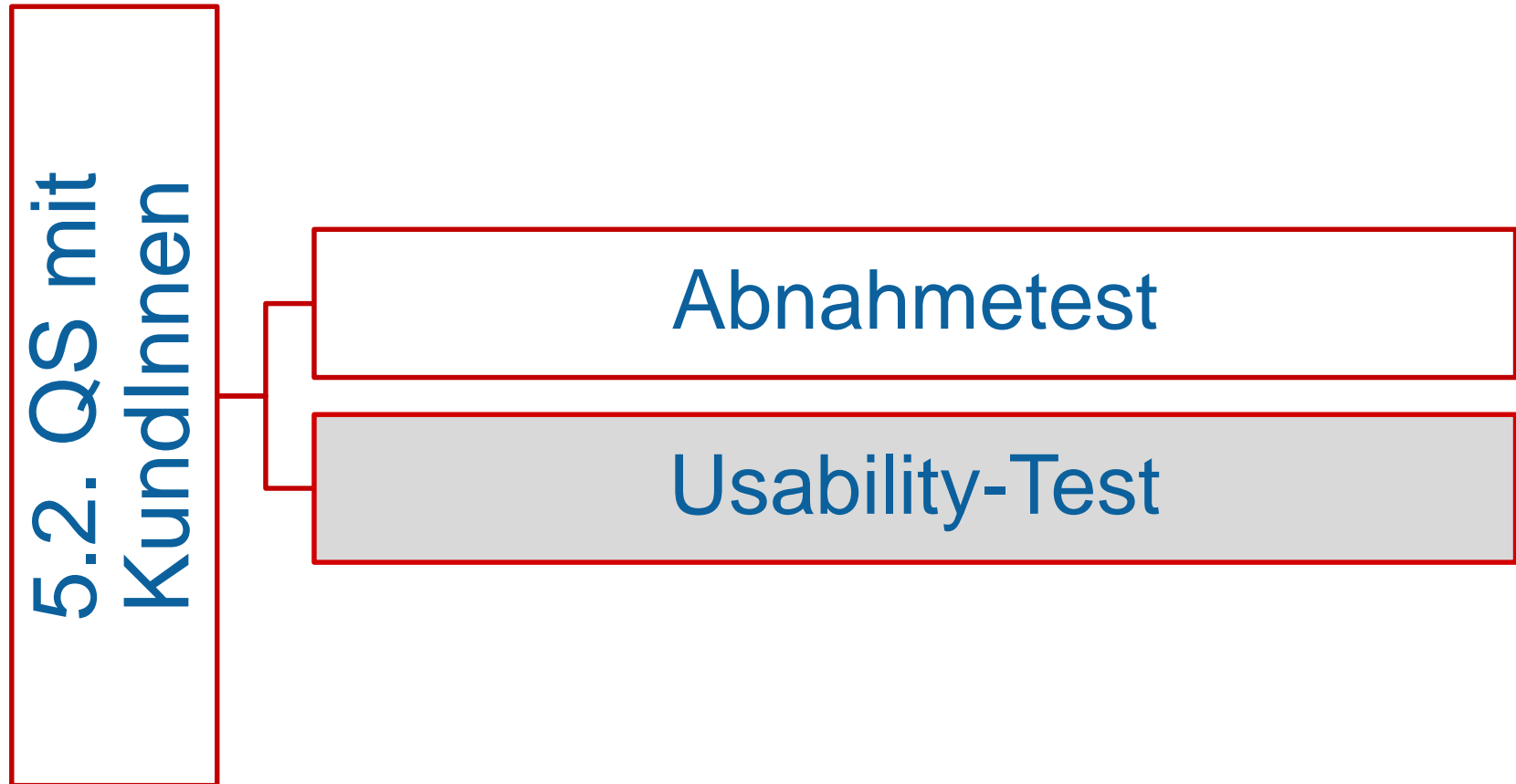


Wdh. Aktivitäten und Ergebnisse der Entwicklung und Qualitätssicherung

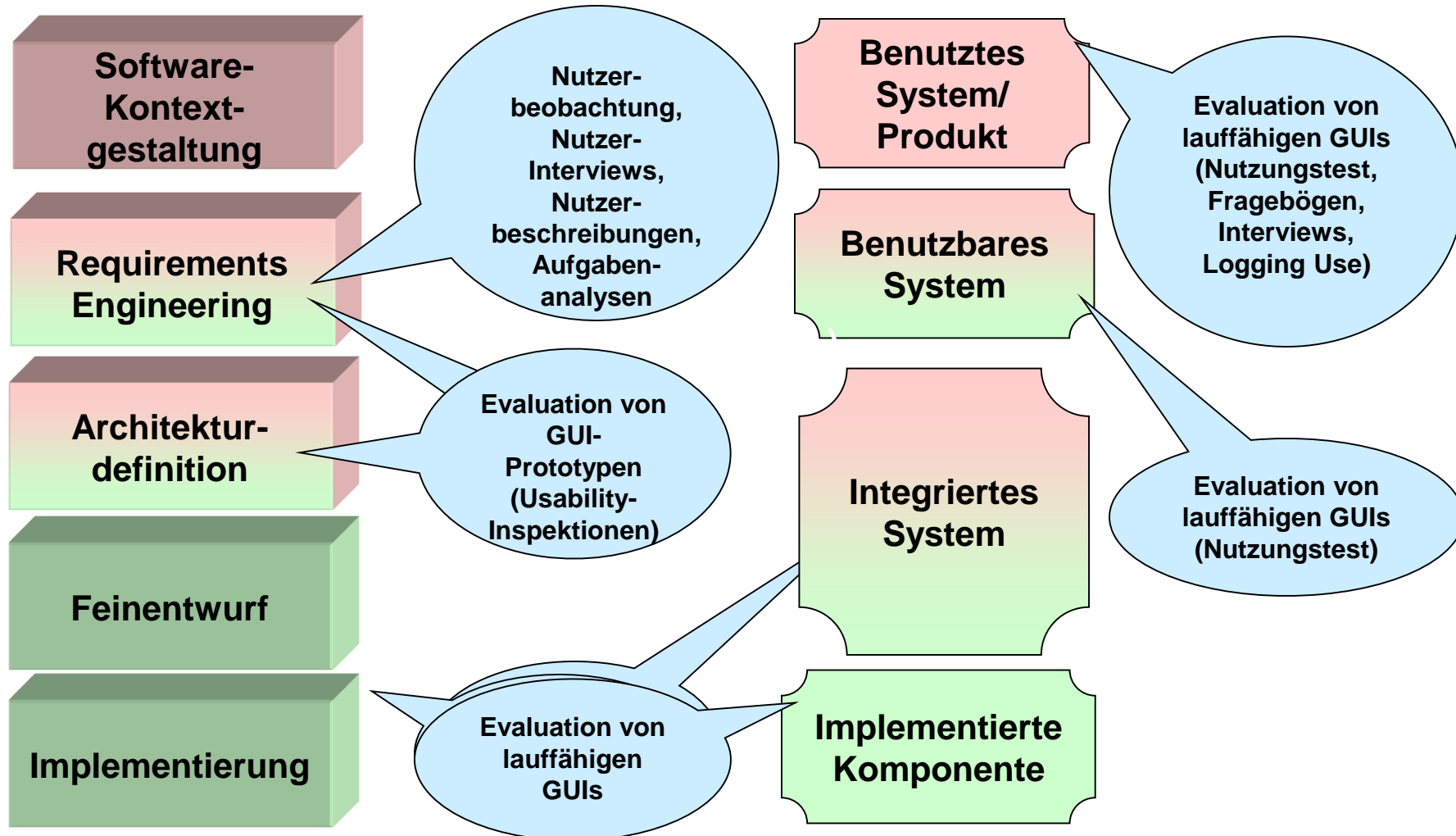


- KundInnen (NutzerInnen) geben die Anforderungen vor. Diese werden in der Modellierung erhoben.
- Es ist aber wichtig, **kontinuierlich** nachzufragen, ob die Zwischenergebnisse der Entwicklung den Vorstellungen der KundInnen (NutzerInnen) entsprechen.
 - **Verifikation** gegenüber den Anforderungsdokumenten
 - Die Dokumente müssen eine geeignete **Qualität** als Kommunikationsgrundlage haben.
 - Dies kann durch **Dokumenteninspektionen** überprüft werden
 - **Validation** gegenüber den Vorstellungen der KundInnen (NutzerInnen)
 - **Nutzungstest** überprüfen Gebrauchstauglichkeit der Software
 - **Abnahmetest** stellt Einsatzfähigkeit der Software im Kundenkontext sicher

- Abnahmetest setzt Systemtest voraus (inkl. Fehlerkorrektur)
- Testet, ob **Kunde System akzeptiert (Validierung)**.
- Verwendet von KundIn gestellte Systemtestfälle
- Umfasst insbesondere auch Nutzungstests (siehe später)
- Testet in **Produktiv-Umgebung**
- Gibt es viele verschiedene Produktiv-Umgebungen, so wird ein systematischer **Feldtest** gemacht (Beta-Versionen an Auswahl von Kunden)

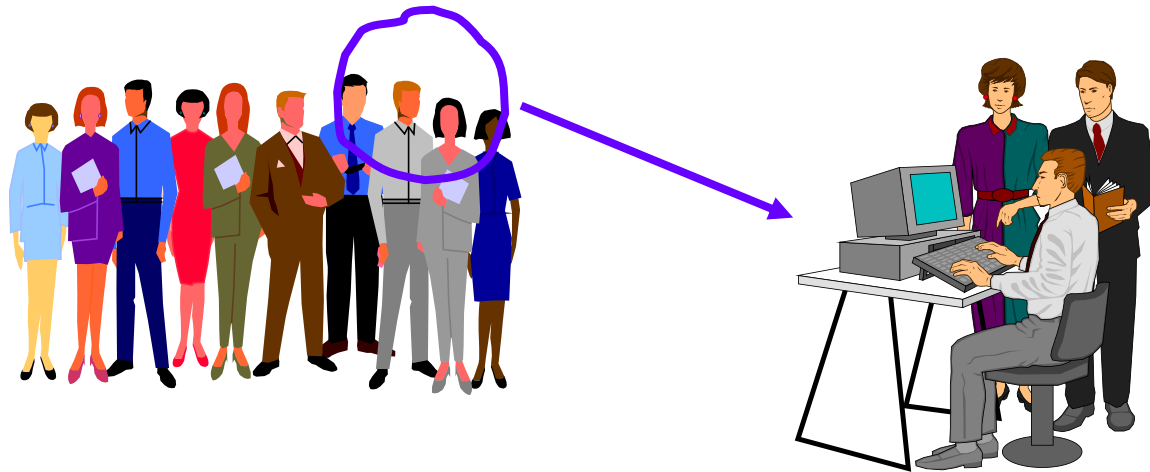


Wie stellt man Usability sicher?



Was ist ein Nutzungstest?

Ein Prozess, in dem für die Zielgruppe des Produktes **repräsentative** NutzerInnen hinzugezogen werden, um zu evaluieren, zu welchem Grad die spezifizierten Usability-Ziele erfüllt werden.



NutzerInnen arbeiten

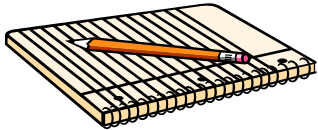
- in möglichst realen Situationen
- an typischen Aufgaben

BeobachterInnen

- beobachten die Interaktion der NutzerInnen mit der Software
- Mitschreiben von Auffälligkeiten
- Videoanalyse
- Methode des Lauten Denkens

Wie läuft ein Usability Test ab?

Ähnlich wie
Empirische
Studie, z.B.
in der Soziologie



**Testplan
entwickeln**

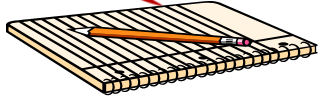
**Test
vor-
bereiten**

**Test
durch-
führen**

**Ergebnisse
analy-
sieren**



Testplan entwickeln (1)



Beschreibt **warum, wann, wo, wie** und **mit wem** der Test durchgeführt wird

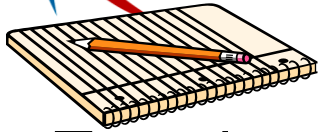
- Ziel
- Problembeschreibung
- NutzerInnenbeschreibung
- Testentwurf
- Liste mit Aufgaben
- Testumgebung
- Evaluations-Metriken

...Mit Hilfe des Tests, sollen Probleme identifiziert werden, die gehäuft von der Hotline gemeldet wurden ...

*...Führt die Antwortzeit des Systems zur Frustration der BenutzerInnen?
Entsprechen die Sichten dem Konzeptionellen Modell der BenutzerInnen?*

Ausbildung, Alter, Lernstil, Erfahrung mit Computern,

Testplan entwickeln (2)



Beschreibt **warum, wann, wo, wie** und **mit wem** der Test durchgeführt wird

- Ziel
- Problembeschreibung
- Nutzerbeschreibung
- Testentwurf
- Liste mit Aufgaben
- Testumgebung
- Evaluations-Metriken

Drucke Text doppelseitig

Vorbedingung: Text geladen, Drucker angeschlossen

Erfolg: Proband findet das richtige Setup und druckt Test doppelseitig

Zeit um die Aufgabe zu erledigen

Anzahl der falschen Menüauswahl

- Rekrutierung von Test-NutzerInnen (NutzerInnen müssen **repräsentativ** sein!)
- BeobachterInnen auswählen
- Verteilung und Review des Testplans
- Szenarien für Aufgaben vorbereiten
 - Information für Test-NutzerInnen vorbereiten
- (Background-, Pretest-, Posttest-) Fragebogen vorbereiten
- Listen zum Protokollieren vorbereiten
- „**Geheimhaltungsvereinbarung**“, Erlaubnis für Video
- Testumgebung vorbereiten



Test durchführen (1)

zuerst **Pilot Test** durchführen
dann **eigentlicher Test**

Vor dem Test:

- Begrüßung
- Information über Test
- Pretest-Fragebogen und Vereinbarungen ausfüllen

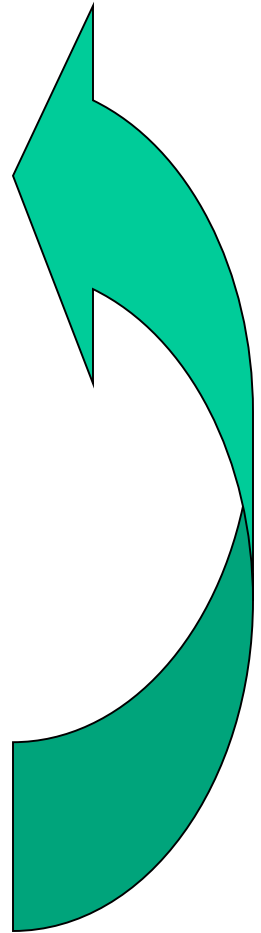


Der Test:

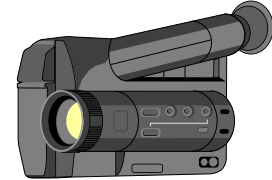
- Aufgabenbeschreibung aushändigen
- NutzerIn beginnt mit der Bearbeitung der Aufgabe
- NutzerIn: „lautes Denken“
- BeobachterIn notiert Auffälligkeiten/ keine Hilfestellung

Nach dem Test:

- Post-Test Fragebogen ausfüllen lassen
- Debriefing (Nachbesprechung)

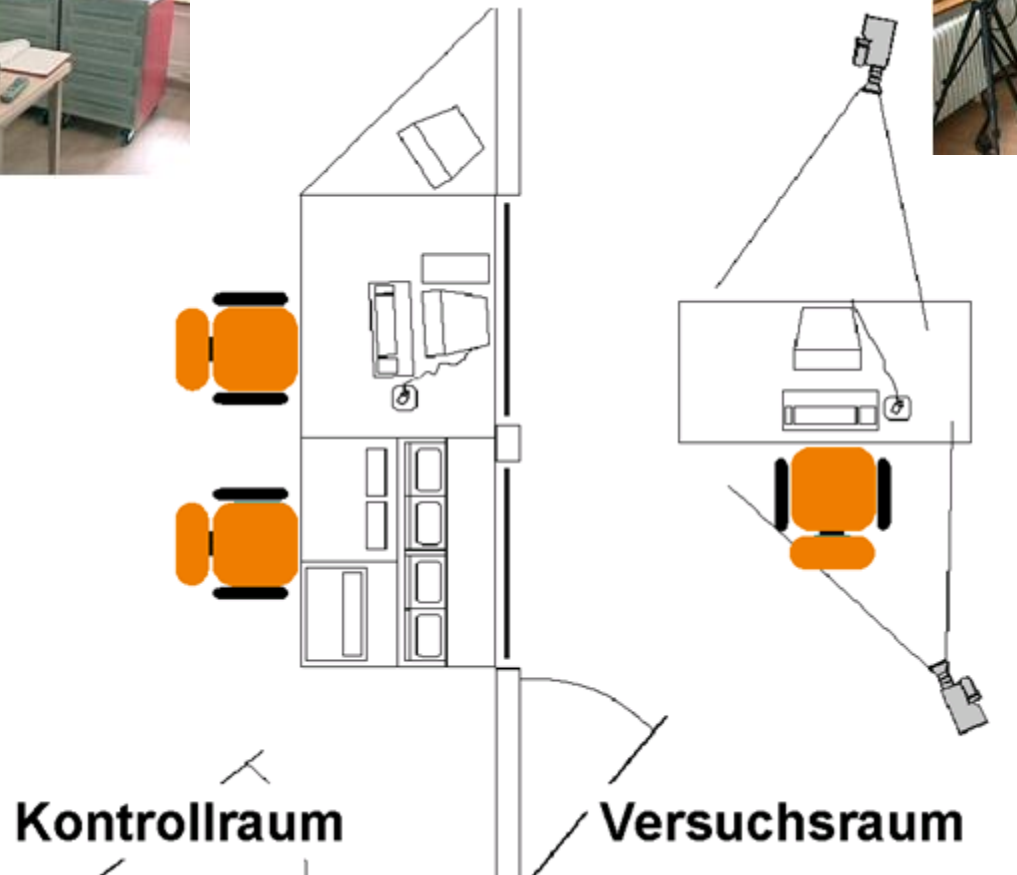


Beobachtung und Aufzeichnung mit Video



- Video enthält Daten, die man übersehen hat
- Video sollte den EntwicklerInnen gezeigt werden
 - EntwicklerInnen lernen, wer die NutzerInnen sind
 - Sie sehen, wie diese die Software bedienen
 - Bilder überzeugen
- Auswertung der Videodaten ist sehr zeitaufwändig (ca. 3mal Aufnahmedauer)

Usability-Labor





Transformation der Daten in Verbesserungsvorschläge

- Schnelle verbale Kommunikation der “Hot Spots”
- Ausarbeitung eines umfassenden Berichtes
 - (Statistische) Auswertung der Beobachtung (z.B. Zeitverhalten)
 - Auswertung der Fragebögen und Debriefings
 - Identifikation der “Aufgaben”, für die die gesetzten Ziele nicht erreicht wurden
 - Beschreibung der Schwierigkeiten, die die NutzerInnen hatten
 - Identifikation von Gründe für diese Schwierigkeiten
 - Empfehlungen diese zu beheben

Zusammenfassung QS für Usability

- Abnahmetest erfolgt in Produktivumgebung.
- Usability muss **kontinuierlich** im Softwareentwicklungsprozess sichergestellt werden.
- **Nutzungstest** erlauben systematisch Usability zu erheben, nachdem die GUI implementiert wurde

- J. Nielsen, *Usability Engineering*, Morgan Kaufman, 1994
- J. Rubin, *Handbook of Usability Testing*, John Wiley & Sons, 1994
- <https://uxpa.org> (usability professionals)
- <http://www.usabilitynet.org>
- <https://listserv.uni-siegen.de/cgi-bin/mailman/listinfo/mci> (deutschsprachige Mailingliste)

5.3. Use Cases

Motivation

UC-Beschreibung

Ableitung über Aktivitäten

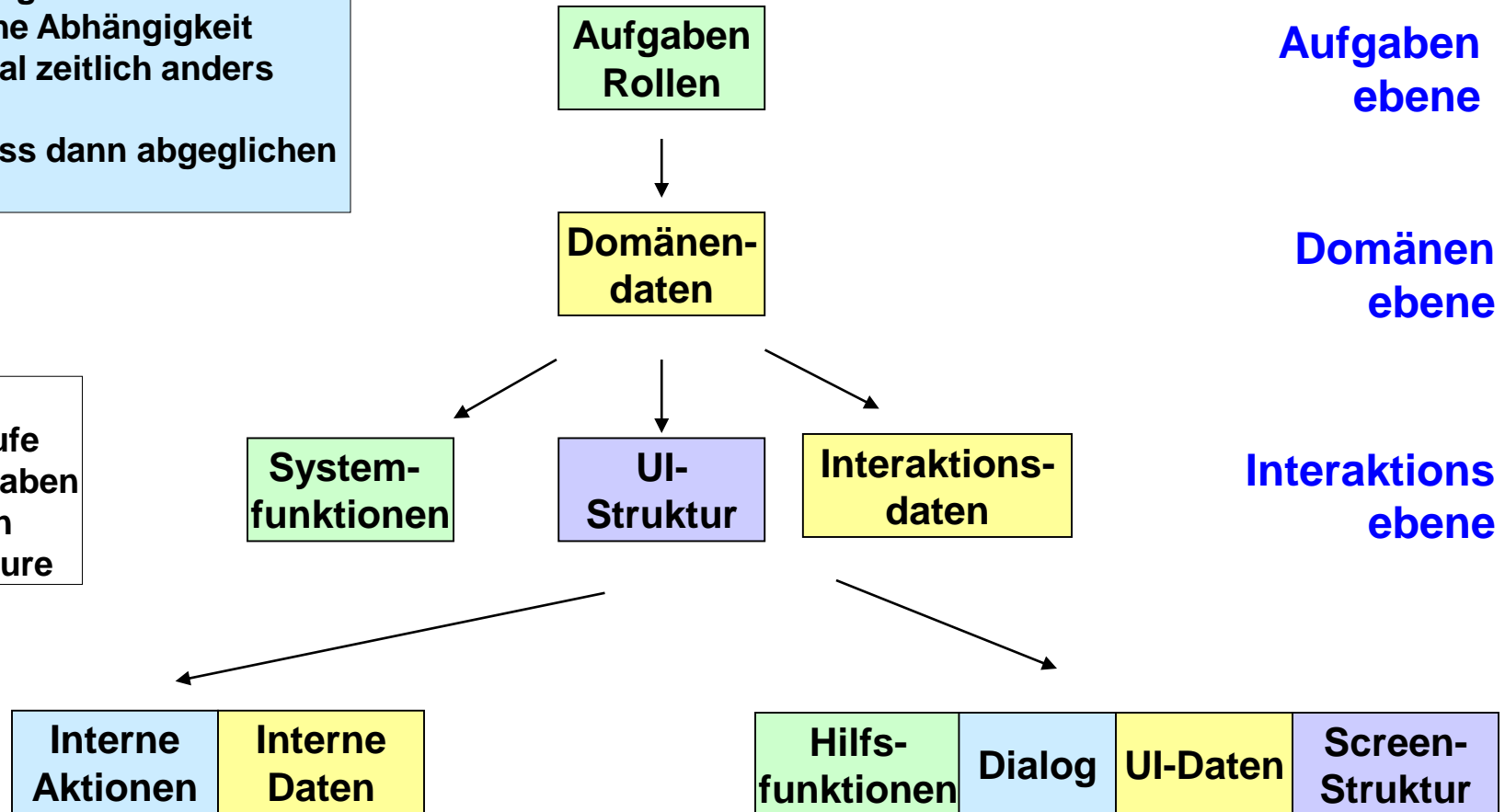
Verwendung für Systemtest

Ergänzungen

Wdh. Anforderungs- und Entwurfsgestaltung

Reihenfolge bedeutet
inhaltliche Abhängigkeit
Manchmal zeitlich anders
erstellt
Aber muss dann abgeglichen
werden

Legende:
Blau: Abläufe
Grün: Aufgaben
Gelb: Daten
Lila: Strukturen



**Systemebene: Anwendungskern
(siehe OOAD)**

Systemebene: GUI

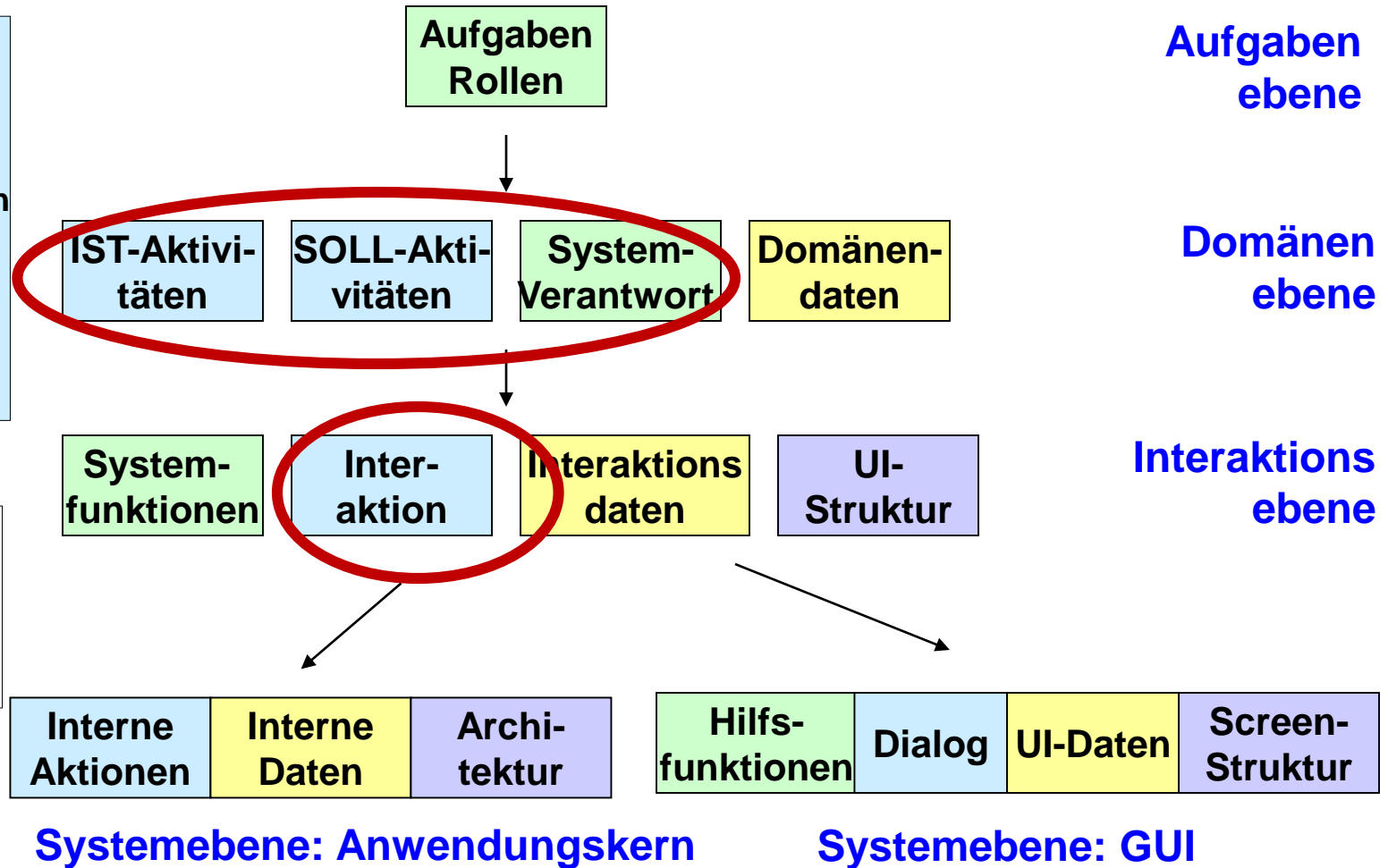
Erweiterte Nutzungsmodellierung

- Wir haben bisher **Anforderungen** durch
 - Aufgaben, Domänendaten
 - Systemfunktionen, Domänendaten, UI-Struktur beschrieben.
Das ist gut für Ideenfindung, was zu tun ist.
- Im Entwurf wurde das durch konkrete Sichten und Dialogmodelle verfeinert.
- Dadurch wird aber nur indirekt beschrieben, **wie (in welchen Abläufen) die NutzerInnen das System nutzen werden**.
 - z.B. wie genau wird eine Aufgabe mit dem System ausgeführt, welches Verhalten genau sollte im Usability-Test getestet werden?
- Das kann durch **Use Cases** beschrieben werden, die aus Aufgaben über **Aktivitäten** abgeleitet werden.

Gesamtbild Beschreibungsebenen

Reihenfolge bedeutet inhaltliche Abhängigkeit
Manchmal zeitlich anders Erstellt.
Aber muss dann abgeglichen werden

Legende:
Blau: Abläufe
Grün: Aufgaben
Gelb: Daten
Lila: Struktur



5.3. Use Cases

Motivation

UC-Beschreibung

Aktivitäten

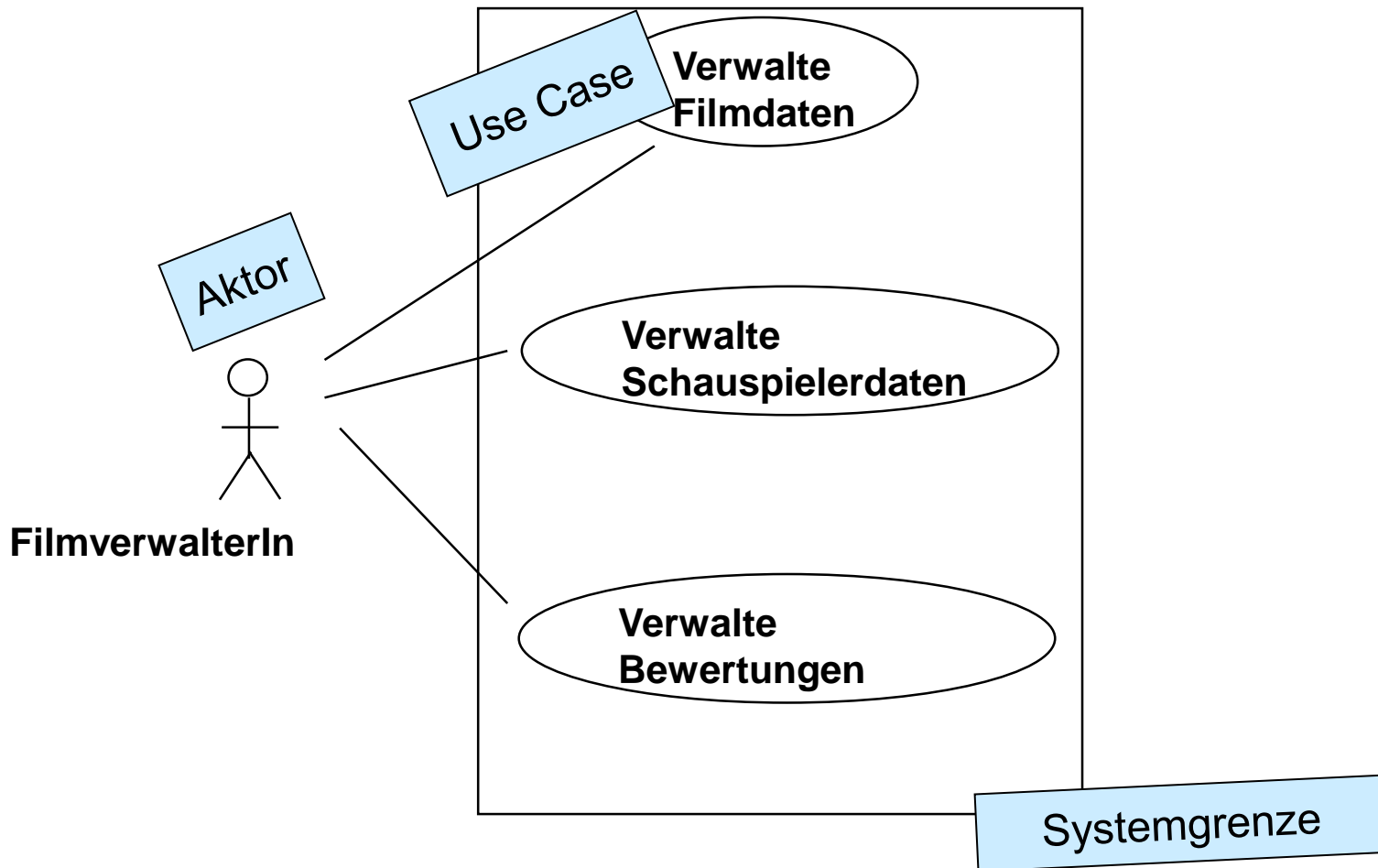
Verwendung für Systemtest

Ergänzungen

- Fokus auf der Gestaltung der Grenze/Schnittstelle zwischen Mensch und Maschine bei der aufgabengerechten Umsetzung der Systemverantwortlichkeiten
 - Welche Funktionen bietet das System an?
 - **Funktionsbeschreibung**
 - Wie interagieren / kommunizieren NutzerInnen mit dem System?
 - **Use Cases (Nutzungsfälle)**
 - Welche Daten tauschen NutzerInnen und System aus?
 - **Verfeinertes Datenmodell**
 - In welchen Zusammenhängen können NutzerInnen welche Funktionen aufrufen? Welche Daten sehen Sie dabei? Wie gliedert sich die Gesamtfunktionalität in Teilbereiche?
 - **User Interface Struktur (UI-Struktur)**
 - **Achtung: Nicht GUI-Struktur, d.h. konkretes Layout noch nicht wichtig**

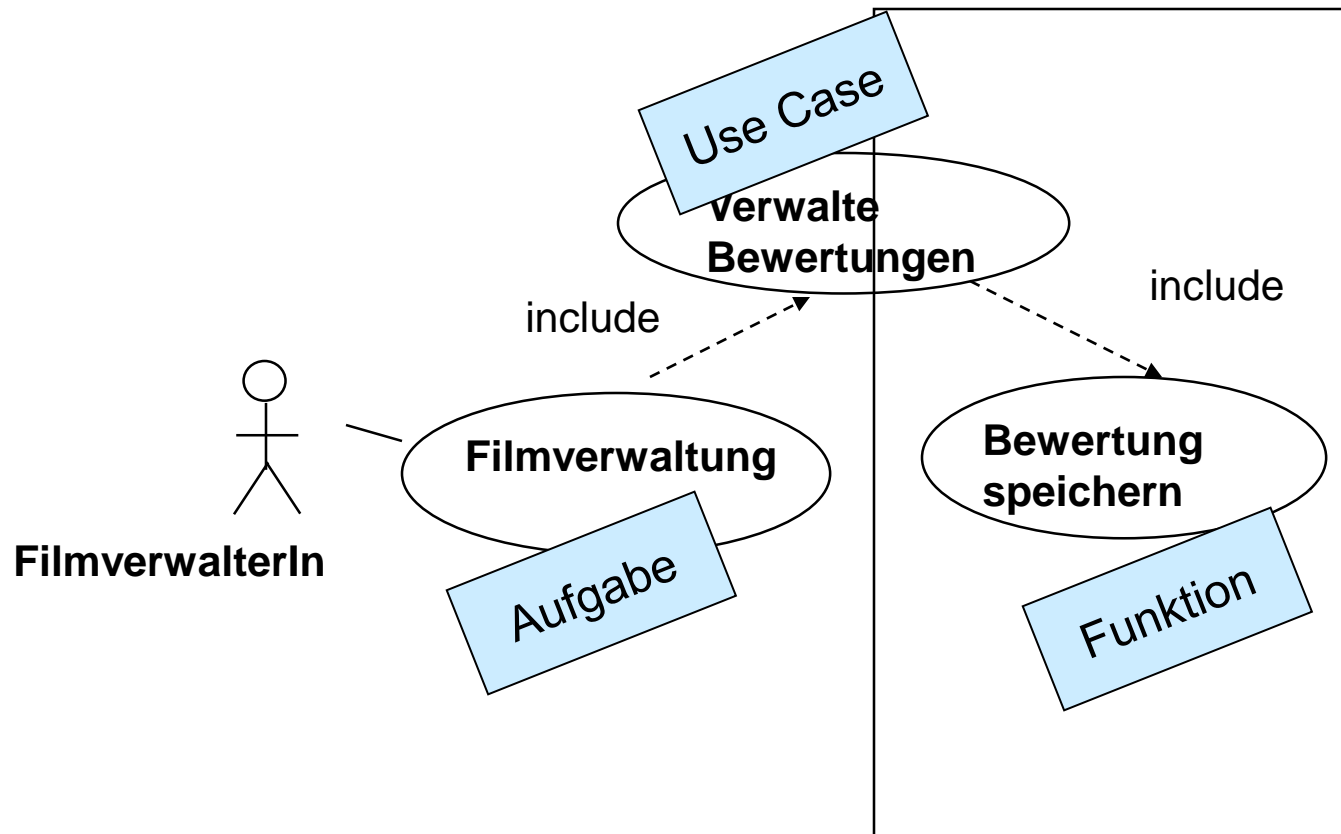
- Ein **Use Case** (Nutzungsfall, UC) fokussiert auf die **Interaktion** zwischen NutzerIn und Software bei der Ausführung einer (Folge von) Systemfunktion(en) zur Erreichung eines bestimmten **Ziels**
- Beschreibung von Interaktionsfolgen
 - gut geeignet für Kommunikation mit BenutzerInnen

Use Case Diagramm



- Wir erweitern das UC Diagramm, um eine Übersicht über Aufgaben, Use Cases und Systemfunktionen zu geben. Das erweiterte Diagramm nennen wir **Nutzungsdiagramm**.
- Im Unterschied zum üblichen UC Diagramm werden die „Bubbles“ für die Use Cases **AUF** der Systemgrenze gezeichnet, um deutlich zu machen, dass diese Aufgaben von Aktor und System gemeinsam durchgeführt werden.
- Bubbles **IM** System beschreiben dann Systemfunktionen

Nutzungsdiagramm (2)



- Der Akteur befindet sich **außerhalb** der Software
- Er interagiert mit dem System (aktiv oder passiv)
- Ein Akteur repräsentiert einen Menschen, der eine **Rolle** inne hat, oder er repräsentiert ein externes System, z.B.:
 - Richtig: Systemadministrator
 - **Falsch: Heinz Müller (was ist seine Rolle?)**
 - Richtig: Datenbank
- Akteure können im UC unterschiedliche Funktionen haben
 - Initiator
 - Externer Server
 - Empfänger
 - Zwischenstufe für Interaktion zwischen System und anderem Akteur

Wie beschreibt man den UC näher?

- Welche Informationen sind **wichtig**,
 - damit **NutzerInnen** entscheiden können, ob das System so funktioniert, wie sie es gerne hätten?
 - damit **EntwicklerInnen** wissen, was sie implementieren sollen?

Kurzes Beispiel: UC Verwalte Bewertungen

- **Aktor:** FilmverwalterIn
- **Ziel:** Bewertungen für Filme und SchauspielerInnen abgegeben
- **Vorbedingungen:** Film oder SchauspielerInnendaten liegen vor
- **Beschreibung des Ablaufs:** Aktor sucht Film bzw. SchauspielerIn und ruft dann Bewertungsfunktion auf. Das System ermöglicht Eingabe der Bewertung. Der Aktor gibt Bewertung ein. Das System prüft und speichert die Bewertung, und berechnet die Gesamtbewertung neu.
- **Regeln:** Bewertung muss in vorgegebener Skala sein.
- **Ausnahmefälle:** Film oder SchauspielerIn existiert nicht im System, Bewertung falsche Skala
- **Qualitätsanforderungen:** einfache Eingabe
- **Daten:** Film, SchauspielerIn Bewertung
- **Funktion:** Bewertung speichern
- **Nachbedingungen:** Bewertung von Film bzw. SchauspielerIn ist im System

Beschreibungsvorlage eines Use Cases

Name

Kurzbezeichnung des Use Cases

Aktor

Welcher Aktor löst den Use Case aus?

Ziel

Was soll durch den Use Case erreicht werden?

Vorbedingung

Zustand **von System und Umgebung aus Sicht des Aktors** bevor der Use Case eintritt

Ablaufbeschreibung

Was soll **normalerweise** passieren?

Ausnahmefälle

Welche Ausnahmen gibt es? Was soll dann passieren?

Regeln

Komplexe funktionale oder kausale Zusammenhänge

Qualitätsanforderungen
Daten, Funktionen

Welche übergreifenden Eigenschaften sind wichtig?
Welche Daten und Funktionen werden verwendet ?

Nachbedingung

Zustand **des Systems aus Sicht des Aktors** nachdem der Use Case erfolgreich beendet ist

Langes Beispiel: Use Case Sitzplatzzuweisung

- **Aktor:** PassagierIn, evtl. SchalterangestellteR.
- **Ziel:** Aktor hat Sitzplatz für gebuchten Flug.
- **Vorbedingung:** Aktor hat eine gültige Flugbuchung der Fluglinie.
- **Ablaufbeschreibung:**
 - **A1:** Aktor verlangt einen Sitzplatz für den Flug XY123. Dies kann Teil des Eincheckens oder eine Vorreservierung sein.
 - **S1:** Das System (evtl. mithilfe der/des Schalterangestellten) fragt nach dem Flugdatum, der Flugnummer, dem Flughafen und dem Namen.
 - **A2:** Aktor gibt diese Informationen an. Statt des Namens kann er/sie auch die Vielfliegernummer angeben. **[Ausnahmefall: Es ist zu früh, um Sitzplatzzuweisungen durchzuführen.]**
 - **S2.1:** Das System findet eine Flugbuchung. **[Ausnahmefall: Keine Buchung gefunden.]** Falls der Flugbuchung bereits ein Sitzplatz zugewiesen wurde, wird diese dem/der Aktor gezeigt und die Möglichkeit gegeben, den Platz zu ändern.

- **S2.2:** Falls keine Sitzplatzzuweisung existiert oder Aktor diese ändern möchte, erfragt das System bevorzugte Plätze: (1) Fenster oder Gang, (2) Nichtraucher oder Raucher.
- **A3:** Aktor gibt die Daten an.
- **S3:** Das System nutzt diese Informationen und die Vielfliegernummer (falls angegeben), um einen passenden Sitzplatz zuzuweisen, abhängig von einer früheren Sitzplatzzuweisung und der Geschäftspolitik der Fluggesellschaft. **Falls nötig**, fragt das System zusätzlich nach, ob Aktor auch folgendes akzeptieren würde: (1) einen Sitzplatz mit Blick auf die Tragfläche, (2) einen Sitz am Notausgang?
- **A4:** Aktor gibt die Daten an.
- **S4:** Das System schlägt eine Sitzplatzzuweisung vor. Falls nicht alle Wünsche des/der Aktor berücksichtigt werden können, versucht das System so viele wie möglich zu erfüllen. **[Ausnahmefall: Keine Sitzplatzzuweisung möglich.]**.
- **A5:** Aktor akzeptiert den Platz oder bittet um Änderungen.....

- **Ausnahmefälle:**

A2.Zu früh: Falls das aktuelle Datum zu früh ist, teilt das System Aktor mit, ab wann Sitzplatzzuweisungen möglich sind.

S2.1.Keine Buchung gefunden: Falls keine Flugbuchung gefunden wurde, prüft das System die vorhandenen Daten gegen die Buchung nach teilweisen Übereinstimmungen. Falls auch jetzt keine Buchung gefunden werden kann, empfiehlt das System dem/der Aktor, eine Buchung durchführen zu lassen.

S5.Keine Sitzplatzzuweisung möglich: Falls die Zuweisung eines Sitzplatzes unmöglich, empfiehlt das System dem/der Aktor, eine Sitzplatzzuweisung zum Zeitpunkt des Check-Ins durchführen zu lassen. Falls dies bereits der Check-In ist, setzt das System den/die Aktor auf die Standby-Liste.

Use Case Sitzplatzzuweisung (cont.)

Regeln:

<Aussagen zu Definition für frühesten Zeitpunkt der Reservierung, Definition für Sitzplatzzuweisung, Definition für Standby-Listenreihung>

Qualitätsanforderungen:

starke Benutzerführung, da Laien als Nutzer

Daten:

Passagierdaten, Fludaten, Flugbuchung, Sitzplatz-zuweisung, Standby-Liste

Funktionen:

Suche einer Flugbuchung, Durchführen einer Sitzplatzzuweisung

Nachbedingungen:

Aktor hat eine Sitzplatzzuweisung. Diese ist im System gespeichert

Typische Fehler bei der UC Erstellung

- **Namen der UC nicht einheitlich:** am besten immer <Objekt Prädikat>
- **Ziel nicht klar vom Namen getrennt:** Ziel sollte in Vergangenheit formuliert werden
- **Vorbedingung vs. Ausnahmefall:** Vorbedingung muss für Aktor sichtbar sein
- **Nachbedingung vs. Ziel:** Nachbedingung nur Systemzustand, Ziel ist Aktorsicht
- **Passivstil:** es muss immer klar sein, wer etwas tut (Aktor, System oder weitere Beteiligte?)
- **Berechnungsdetails mit Ablauf gemischt:** Details in den Regeln beschreiben

Siehe
UC-
Stilratgeber

- UC werden in der Literatur **in unterschiedlichen Detaillierungsgraden** beschrieben
 - Geschäftsprozesse (hier nicht behandelt)
 - Kundenanforderungen
 - Softwarespezifikation
 - Benutzungsschnittstelle

- *Aktor kann mit dem System Einzelbewertungen zu Filmen und SchauspielerInnen abgeben.*
- Abstrahiert weitgehend von Ausnahmen
- Entscheidungsebene Arbeitsgestaltung: Eingabe jederzeit möglich? Oder nur gebündelt pro Film?
- UC sehr grob: => Besser als Systemverantwortlichkeiten (siehe später) beschreiben (Domänenebene), siehe auch user stories bei agiler Entwicklung
- Ist hilfreich insbesondere für Planung, grobe Abstimmung zwischen Auftraggeber und Auftragnehmer

Beispiel: Softwareanforderungsebene

- *A1: Akteur sucht Film und ruft dann Bewertungsfunktion auf. **[Ausnahme: Film nicht bekannt].***
S1: Das System ermöglicht Eingabe der Bewertung.
A2: Der Akteur gibt Bewertung ein.
S2: Das System prüft und speichert die Bewertung und berechnet die Gesamtbewertung neu.
- **Ausnahmen** aufgrund der **internen Verarbeitung**
- **Entscheidung Gestaltung der Systemfunktionen:** z.B. Prüfung nicht bei jedem Eintrag
- **Passt am besten zu UI-Struktur-Granularität (Interaktionsebene)**
- Beachte: Detail der **internen Aktionen sind in Systemfunktionsbeschreibung** zu definieren (z.B. was genau geprüft)!

Beispiel: Benutzungsschnittstellen-Ebene

- *A1: Akteur öffnet Suchfenster*
S1: System zeigt Maske zur Eingabe von Filmnamen.
A2: Akteur gibt Namen ein [Ausnahme: Film nicht bekannt].
S2: Das System zeigt den Film an .
A3: Der Akteur ruft Bewertungsfunktion auf.
S3: Das System ermöglicht Eingabe der Bewertung.
A4: Der Akteur gibt Bewertung ein.
*S4: Das System prüft und speichert die Bewertung **und bestätigt dies.***
[Ausnahme: Bewertung entspricht nicht Skala]
- *S5: Das System berechnet die Gesamtbewertung neu.*

- **Ausnahmen** aufgrund von **NutzerInneneingaben oder Systemausgaben**
- **Entscheidung Paradigma:** Alternative, z.B. zuerst Objektauswahl dann Funktionsauswahl
- **Entscheidung Ergonomie:** Bestätigungsabfrage
- **Sehr detailliert: => Kann besser bei Dialogen beschrieben werden (GUI-Ebene)**

Interaktionsbeschreibung durch UC

Use Case Template	Kundenanforderungen	Softwareanforderungen	Benutzungsschnittstellenbeschreibung
Aktor	Rolle	Rolle	Rolle
Ziel	Aufgabe der Rolle	Interne Aufgliederung einer komplexen Systemfunktionsfolge	Einzelheiten der Interaktion zwischen System und Actor
Ablaufbeschreibung	Informationsfluss zwischen Actor, System u. anderen Aktoren	Eingaben zu Systemfunktionen, deren Verarbeitung durch Änderungen auf Systemdaten u. Ausgaben	Benutzungsschnittstellenkommandos, Systemausgaben auf dem Bildschirm
Ausnahmefälle	Nur sehr wichtige, die Arbeitsgestaltung beeinflussen	Bzgl. der internen Verarbeitung	Bzgl. Benutzereingaben, Systemausgaben
Regeln	Zusammenarbeit mit anderen Aktoren	Geschäftsregeln, die bei der Abarbeitung der Systemfunktionen zu beachten sind	Formatvorgaben, Gültigkeitsbereich für Ein/Ausgaben)
Qualitätsanforderungen	Für die Arbeit (z.B. Humane Arbeit)	Für die interne Durchführung der Systemfunktion (z.B. Volumen, Performanz)	Für die Interaktion (z.B. Selbstbeschreibungsfähigkeit)
Daten	Informationen, Produkte	Systemdaten	Sichten
Funktionen	Aktivitäten/ Systemfunktionen	Systemfunktionen	Interaktionsteilschritte
Vorbedingung	Bzgl. Aufgabe	Bzgl. Systemzustand	Bzgl. Systemzustand, Ein/Ausgabegeräte
Nachbedingung	Ergebnis der Aufgabe	Systemzustand	Bzgl. Systemzustand, Ein/Ausgabegeräte

- Wir beschreiben UC auf **Softwareanforderungsebene**.
- Dabei verwenden wir auch **abstrakte Benutzungsschnittstellenbeschreibung** in Form von Arbeitsbereichen (siehe UI-Struktur)
- Der Arbeitsbereich charakterisiert für Aktor sichtbare Daten und Funktionen
 - Wird **in Vorbedingung angegeben**, um deutlich zu machen, mit welcher Ausgangsinformation Aktor startet
 - Wird **in Systemaktion angegeben**, um deutlich zu machen, welche Informationen Aktor nun neu sieht
- Beispiel UC Manage Watched Movies

5.3. Use Cases

Motivation

UC-Beschreibung

Ableitung über Aktivitäten

Verwendung für Systemtest

Ergänzungen

- **Syntaktische Unterscheidung:** Aufgabe durch substantiviertes Verb beschreiben, z.B. Notenvergabe (vs. UC Bewertung abgeben)
- **Semantische Unterscheidung:**
 - **Aufgabe:** größere Handlung, Verantwortlichkeit gegenüber anderen; Unteraufgaben entsprechen Teilziel, Akteur hat „Belohnung verdient“
 - **UC:** ähnlich wie Unteraufgaben, aber im Fokus stehen zusammenhängende Datenänderung im System, die mehrere Schritte umfassen.
- => Bei großen Aufgaben (insbesondere bei Beteiligung mehrerer Akteure): Überbrückung der Granularitätsstufe zwischen Aufgaben und UC durch **Aktivitäten**

- **Syntaktische Unterscheidung:** Bei Systemfunktionen beschreiben Verben Aktionen des Systems, nicht des Aktors
- **Semantische Unterscheidung:**
 - **UC:** zusammenhängende Datenänderung im System, Akteur hat ein Teilziel einer Aufgabe erreicht, Akteur hat „Belohnung verdient“ (z.B. „Bewertung speichern“ ist zu klein, beinhaltet nur Datenänderung, nicht Gesamtbewertung)
 - **Systemfunktion:** Datenänderung im System, die typischerweise an mehreren Stellen (in verschiedenen UC) sinnvoll ist; hat allein aber noch keinen besonderen Wert

- Zuerst Fokus auf Arbeitsplatzgestaltung (Verständnis des Kontexts der NutzerInnen)
 - Welche **Rollen** soll das System unterstützen?
 - Welche **Aufgaben** haben diese Rollen?
 - **Domänendaten**: Welche Informationen sind dabei wichtig?

- Erweitert um Betrachtung der **IST- und SOLL-Aktivitäten (IT-Innovation)**
 - **IST**: Aus welchen **Aktivitäten** bestehen die Aufgaben?
 - **SOLL**: Wie **ändern** sich die Aufgaben und Aktivitäten durch das System?
 - **SOLL**: Welche Aktivitäten soll das System unterstützen (**Systemverantwortlichkeiten**)?

Abstrahiert vom
Softwaresystem!

- IST- und SOLL-Aktivitäten werden wie Aufgaben beschrieben. Sie entsprechen einzelnen Unteraufgaben oder bündeln sie.
- Folgen von Aktivitäten können durch **UML-Aktivitätsdiagramme** beschrieben werden
 - Gut für den Überblick über das Zusammenwirken mehrerer Aufgaben
=> Geschäftsprozesse
 - Ist aber oft unnötig detailliert (die Vielfalt der Abläufe ist so groß, dass man Reihenfolgen nicht übersichtlich aufschreiben kann)

Beispiel Aktivitätsdiagramm

Kundenaufgabe:
Bestellung

Unternehmensaufgabe:
Bestellabwicklung

Lieferantenaufgabe:
Lieferung

Legende:

Kasten = Daten

Abgerundeter

Kasten = Aktivität

Pfeil = Daten-

Fluss

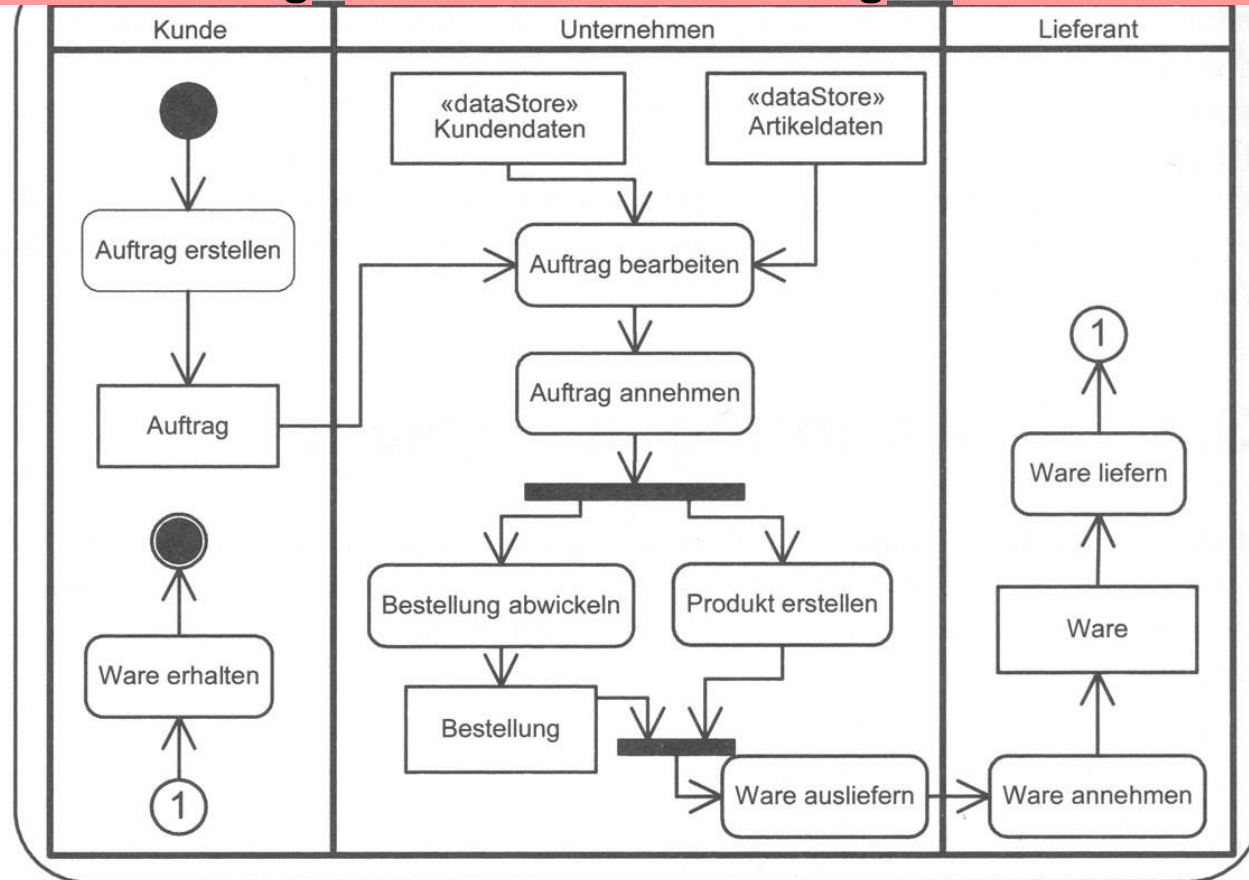
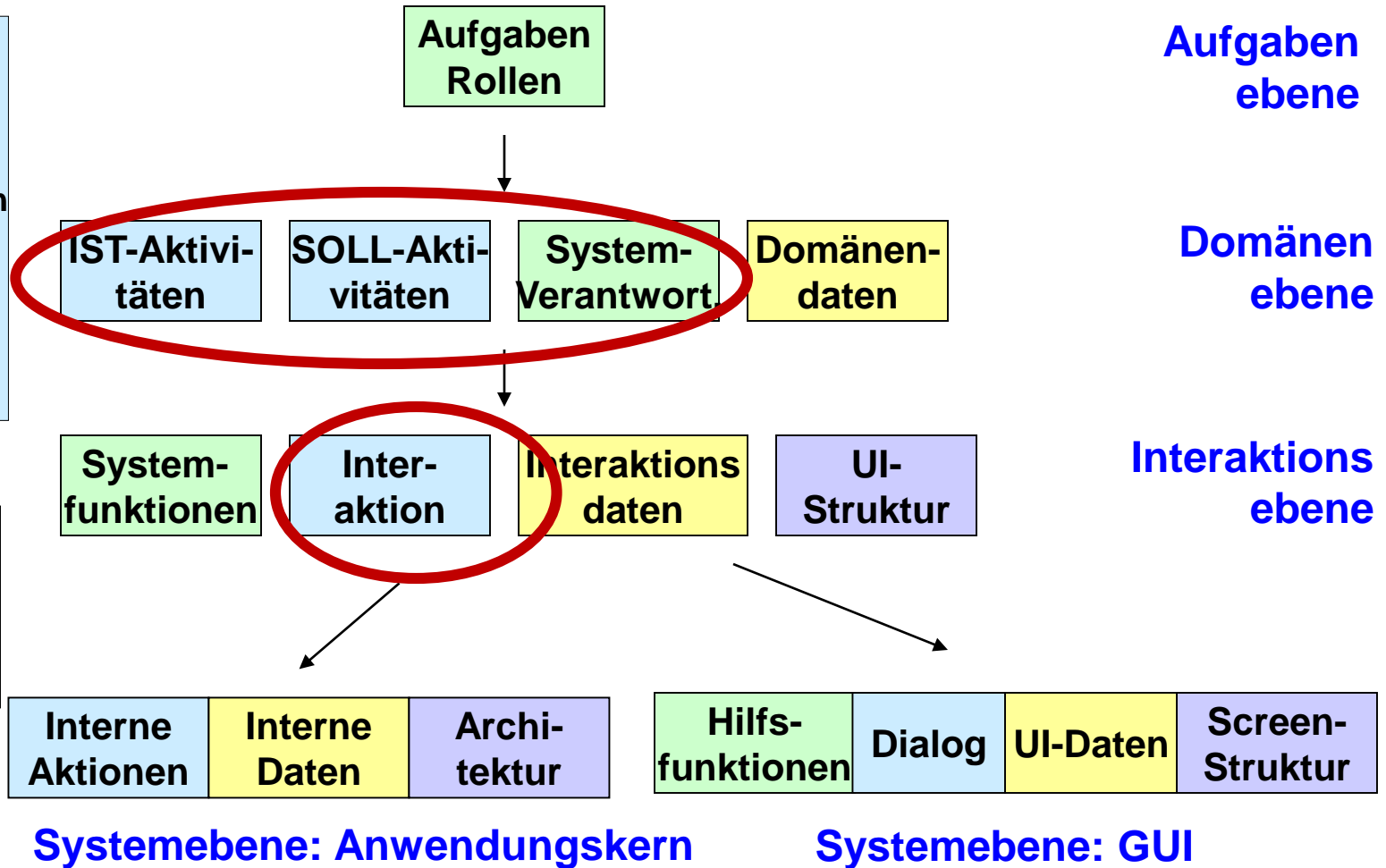


Abbildung 13.10: Vereinfachte Darstellung einer Auftragsbearbeitung

- Diejenigen **Aktivitäten, die das Softwaresystem unterstützen soll**, werden Systemverantwortlichkeiten genannt.
- Bei agilen Methoden werden Systemverantwortlichkeiten durch kurze **User Stories (in einem Satz)** beschrieben.
- Im Release Management werden sie oft **Features** genannt
- Die Systemverantwortlichkeiten eignen sich gut für die Planung, **aber reichen nicht aus, um die Interaktion der NutzerInnen mit dem System zu beschreiben/verstehen.**

Gesamtbild Beschreibungsebenen

Reihenfolge bedeutet inhaltliche Abhängigkeit
Manchmal zeitlich anders Erstellt.
Aber muss dann abgeglichen werden



Legende:
Blau: Abläufe
Grün: Aufgaben
Gelb: Daten
Lila: Struktur

Vorgehen bzgl. Funktionalität insgesamt

- **Identifiziere Aufgabe und Unteraufgaben einer Rolle**
 - **Ggf. Nutzung von Aktivitätsdiagrammen:** Wenn die Rolle dabei mit anderen Rollen zusammenarbeitet, erstelle Aktivitätsdiagramme und identifiziere daraus die Unteraufgaben
 - Ansonsten: bündele wesentliche Teilziele als **Unteraufgaben**
 - **Identifiziere Systemunterstützung**
 - Identifiziere die von der Software zu unterstützenden Unteraufgaben (= **Systemverantwortlichkeiten**)
 - Identifiziere mögliche Systemfunktionen für diese Unteraufgaben
 - Erstelle Nutzungsdiagramm mit einem UC pro Unteraufgabe
 - **Ggf. Nutzung von UC-Text:** Falls eine Unteraufgabe viele Systemfunktionen umfasst oder falls bestimmte Systemfunktionen in bestimmter Reihenfolge auszuführen sind (auch über mehrere Unteraufgaben hinweg), erstelle UC-Texte zur Beschreibung der möglichen Abfolgen (und ggf. weitere UC in Nutzungsdiagramm)
 - Ggf. Nutzung von Systemfunktionsbeschreibung (falls komplexe Berechnung)
 - **Parallel dazu:** Datendiagramme und UI-Struktur
-

5.3. Use Cases

Motivation

UC-Beschreibung

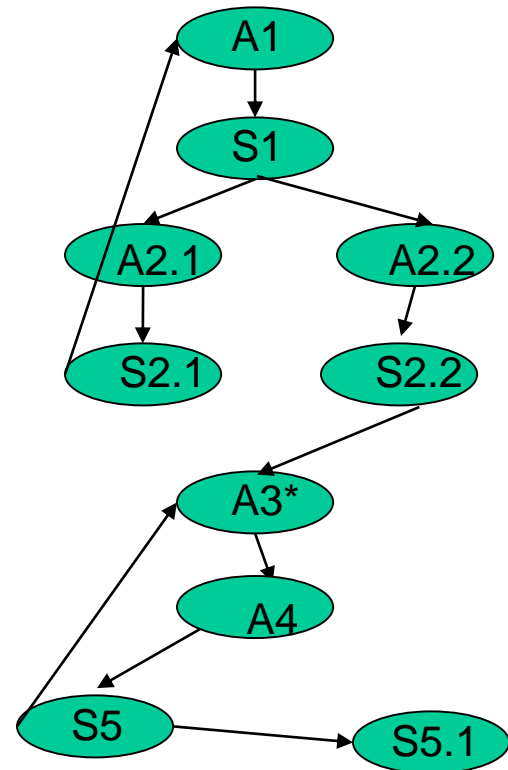
Ableitung über Aktivitäten

Verwendung für Systemtest

Ergänzungen

- UC-Notation für komplexe Abläufe oft nicht ausreichend
- Weitere Notationselements hilfreich
 - **optional**: für Akteur- oder Systemschritt, der auch weggelassen werden kann
 - **optional***: wenn der Schritt mehrfach optional ausgeführt werden kann
 - **VARi**: Akteur oder System können einen Schritt auf unterschiedliche Weise ausführen
 - **weiter mit**: Wiederholung mehrerer Schritte oder Sprung zu einem anderen Zweig

- A1
- S1
- A2: Var1, Var2
- S2: Var1 weiter mit A1, Var2
- Optional* A3
- A4
- S5: [Ausnahme: S5.1], weiter mit A3



- Ein **Szenario**
 - ist ein spezifischer Interaktionsablauf.

- Ein Use Case
 - ist eine abstrakte Beschreibung einer **endlichen Menge** von Szenarien.

- Eine Auswahl von typischen Szenarien kann **Vorlage** für die Erstellung eines UC sein.

- Ein UC kann durch die **nachträgliche** Erstellung von Szenarien validiert werden => *Siehe Systemtest*

- Testet ob Kunden-Anforderungen richtig umgesetzt wurden (Verifikation)
- Testumgebung sollte Produktiv-Umgebung mglst. nahe kommen (also keine Stubs und Testtreiber)
- Produktiv-Umgebung oft selbst nicht geeignet, wegen Schadensrisiko und mangelnder Kontrolle
- Kann einzelne Funktionen, aber auch Funktionssequenzen (für Geschäftsprozesse) testen (siehe Use case basierter Test später)
- Sollte Test von Qualitätsmerkmalen beinhalten wie Performanz, Sicherheit (siehe später)

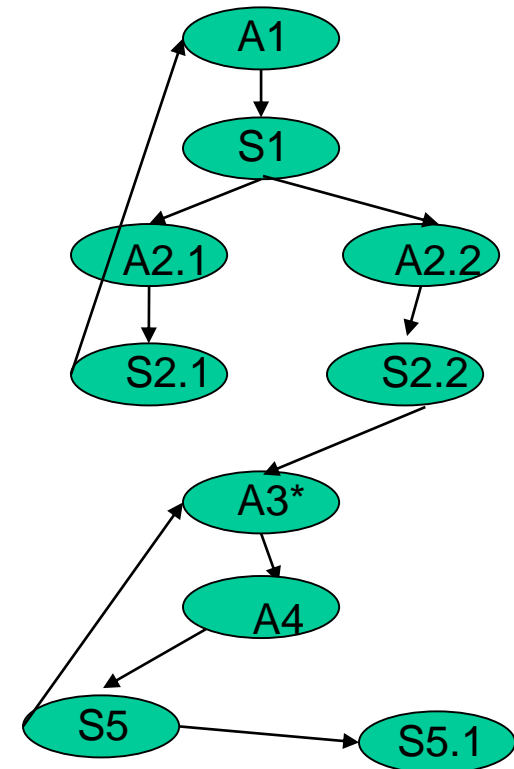
Wdh. 2.6. Zusammenfassung Teststufen

Teststufe	Komponenten	Integration	System
Zu entdeckende Fehler	Fehler in einzelnen Komponenten	Fehler beim Zusammenspiel bestimmter Komponenten	Fehler in Systemfunktionen
Testrahmen	Komponente, Stubs, Treiber	Komponenten, Stubs, Treiber, Monitore	Integriertes System
Teststrategie	Abdeckung der Operationen	Abdeckung der Abhängigkeiten (Top-Down, Bottom-Up)	Abdeckung der Systemfunktionen und UC
Typische Fehler	Falscher Kontrollfluss in den Operationen. Falsche Datenstrukturen	Probleme bei Empfänger, Daten, Vorbedingung, Reihenfolge, Synchronisation, Deadlock	Falsches Zusammenspiel Nutzungsschnittstelle und Systemkern

Es gibt noch weitere Teststufen: Abnahme-Test, Usability-Test => siehe später

- Abdeckung der Systemfunktionen und Abdeckung der Use Cases
- Abdeckung der Systemfunktionen
 - Blackbox
 - White-Box in Bezug auf detailliertere Beschreibung der Funktion (Pseudo-Code, Test)
- Abdeckung von Szenarien, Use Cases
 - White-Box angewendet auf Use Case Beschreibung

- Pfad ist „gültig“, wenn keine Ausnahme vorkommt
- Typische Abdeckung von UC durch Testfälle:
 - Alle Pfade
 - oder alle gültigen bzw. ungültigen Pfade
 - Hier gültig: Alle Pfade, die in S5 enden
 - Hier ungültig: alle Pfade, die in S5.1. enden
 - Oder ein gültiger Pfad



5.3. Use Cases

Motivation

UC-Beschreibung

Ableitung über Aktivitäten

Verwendung für Systemtest

Zusammenfassung

- **Beschreibung der funktionalen Anforderungen**
 - verständlich für die NutzerInnen
 - aus Sicht der NutzerInnen
 - guter Übergang zur Benutzungsschnittstellengestaltung und zum Test
 - auch als Einheit zur Projektplanung

- **Als Mittel zur Anforderungsermittlung**
 - Modellcharakter
 - Detail, Vollständigkeit, Management (Änderbarkeit) **nicht** so wichtig

- **Als Mittel zur Anforderungsspezifikation**
 - Vorgabe für Entwurf
 - Detail, Vollständigkeit, Management (Änderbarkeit) **sehr** wichtig

- Ermittlung vs. Spezifikation
 - Abdeckung des Gesamtverhaltens durch UC ist nicht immer möglich
- Granularität und Inhalt oft nicht ganz klar
 - unzulässige Vereinfachung
 - keine klare Trennung von
 - Akteur - System
 - Typ (UC) – Instanz (Szenario)
 - Normalfall - Ausnahme
 - Kundenanforderungen, Softwarespezifikation, Benutzungsschnittstellengestaltung
 - IST-SOLL
- Pflege aufwändig (Verwaltung, Änderbarkeit)
- => Use Case – Text gezielt einsetzen