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

Kapitel 4: SWE im Großen – Kommunikation der EntwicklerInnen

4. Kommunikation der EntwicklerInnen (3.Teil)

- 4.1. Einführung Modellierung
- 4.2. Klassendiagramme
- 4.3. Interaktionsdiagramme
- 4.4. Zustandsdiagramme
- 4.5. Klassenentwurf mit OOAD
- 4.6. Kommunikation von Erfahrungswissen (Entwurfsmuster)
- 4.7. Kommunikation von Entscheidungen (Rationale)
- 4.8. Zusammenfassung Modellierungstechniken

4.7. Kommunikation von Entscheidungen (Rationale)

Motivation Rationale (1)

Warum der ?

- Dokumente enthalten meist nur die letzte Entscheidung
- Verworfene Optionen und Kriterien sind daraus nicht ablesbar

Motivation Rationale (2)

- Rationale ist die Begründungen für Gestaltungsentscheidungen (während der Softwareentwicklung)
- Fehlendes Rationale führt dazu, dass
  - Entscheidungen oft nicht alles berücksichtigen, weil Kriterien und Optionen nicht systematisch untersucht wurden
  - Entscheidungen oft nicht überzeugend sind für Leute, die nicht dabei waren
  - Entscheidungen nachträglich (z.B. bei Änderungen) völlig umgeworfen werden, weil die Leute die Gründe nicht verstehen
  - Verworfene Optionen (Sackgassen) bei Änderungen noch einmal durchgegangen werden

Wissensbereiche in der Softwareentwicklung

	Wissen über System	Wissen über Prozess (Rollen, Aktivitäten, Dokumente)
Wissen auf Produkt-ebene	**Inhalte:** Spezifikation, Entwurf, Kode, Testpläne, etc	**Inhalte:** Projektplan, Kostenplan, Aufgaben, Richtlinien
Wissen auf Organisa-tions-ebene	**Inhalte:** Domänenmodell, Systemarchitektur, Entwurfsmuster	**Inhalte:** Prozessmodell, Best Practices, Erfahrungen

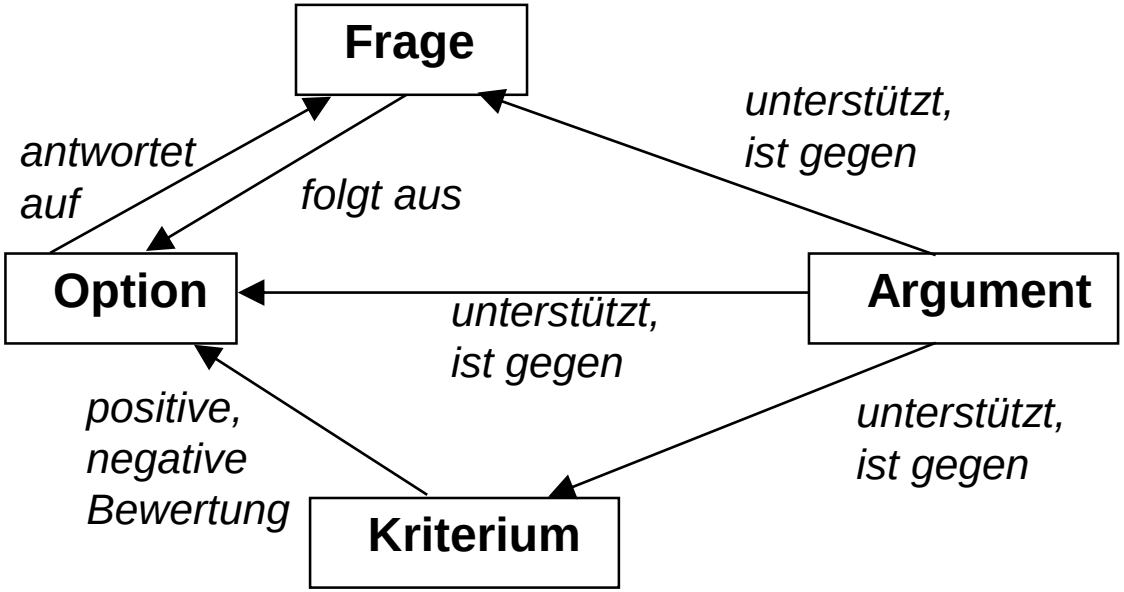
Wissensbereiche in der Softwareentwicklung

	Wissen über System	Wissen über Prozess (Rollen, Aktivitäten, Dokumente)
Wissen auf Produkt-ebene	**Inhalte:** Spezifikation, Entwurf, Kode, Testpläne, etc **Rationale:** Entwicklungsziele, Kriterien, Alternativen, Bewertungen	**Inhalte:** Projektplan, Kostenplan, Aufgaben, Richtlinien **Rationale:** Planungsziele, Risikobewertungen, Kriterien, Alternativen
Wissen auf Organisa-tions-ebene	**Inhalte:** Domänenmodell, Systemarchitektur, Entwurfsmuster **Rationale:** auf Ebene der generalisierten Modelle (z.B. Vor/Nachteile bei Muster)	**Inhalte:** Prozessmodell, Best Practices, Erfahrungen **Rationale:** auf Ebene der generalisierten Modelle (z.B. Erfolgsfaktoren bei Best Practices)

Wie beschreibe ich Rationale?

- Fragen (Issues)
- Optionen
- Kriterien
- Argumente (Diskussionen)
- Entscheidungen

Beispiel Beschreibungstechnik für Rationale: Question Option Criteria (QOC)



[MacLean et al. 1991]

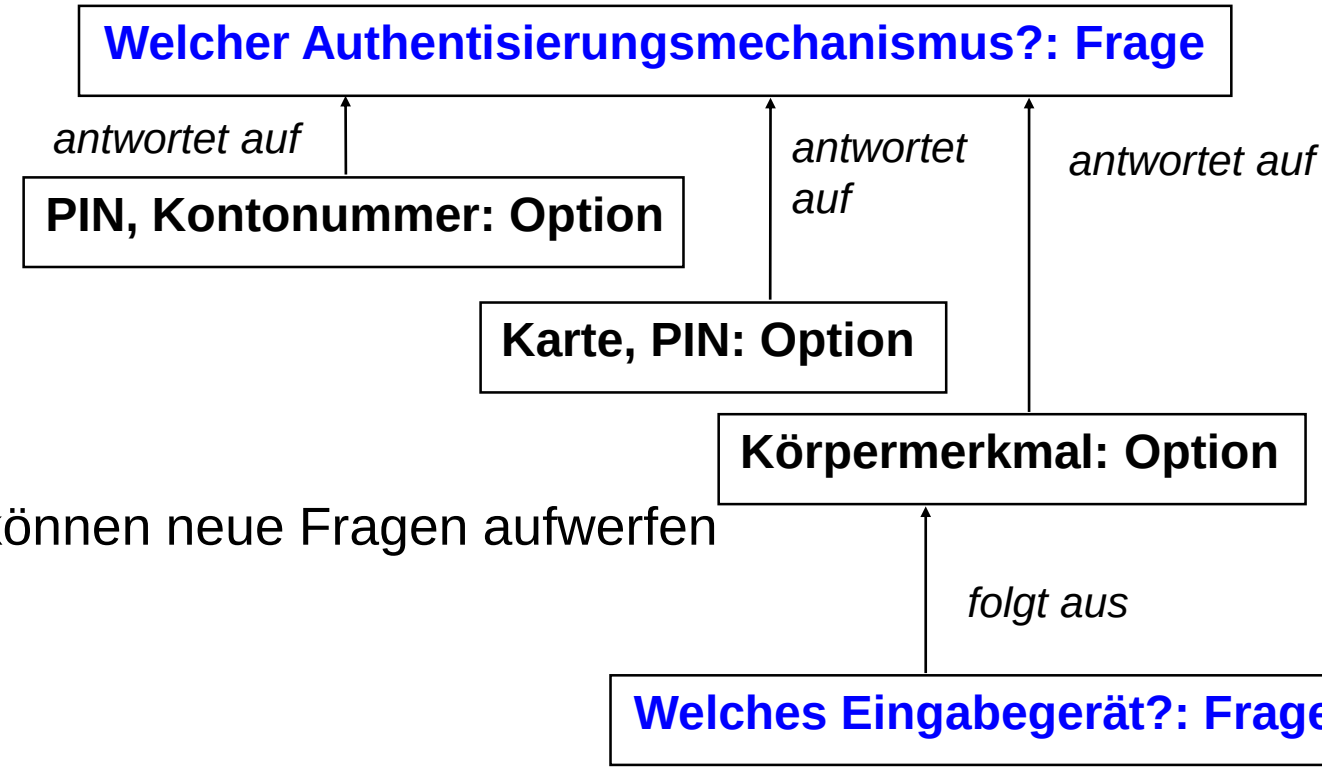
Fragen

- Fragen sind konkrete Probleme, die keine eindeutige Lösung haben
- Typische Fragen bei Dokumentationselementen:
  - Frage bzgl. der Form
  - Klärungsfrage
  - Übersehenes
  - Inkonsistenz
  - Begründung
  - **Frage bzgl. des Inhalts, die andere Optionen aufzeigt**

Welcher Authentisierungsmechanismus?: Frage

Optionen

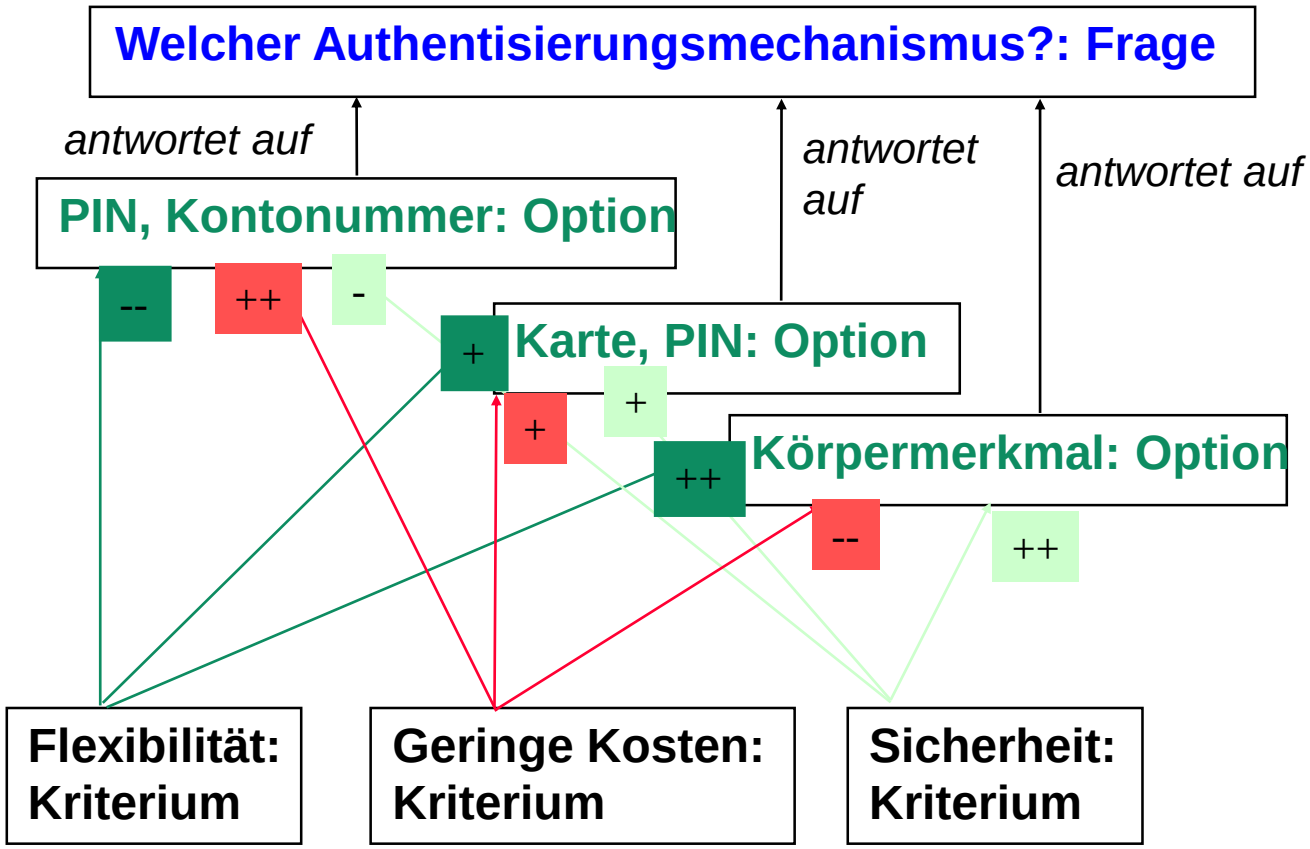
- Optionen beschreiben Alternativen zur Lösung der Probleme
- eine Option kann sich auf mehrere Probleme beziehen



- können neue Fragen aufwerfen

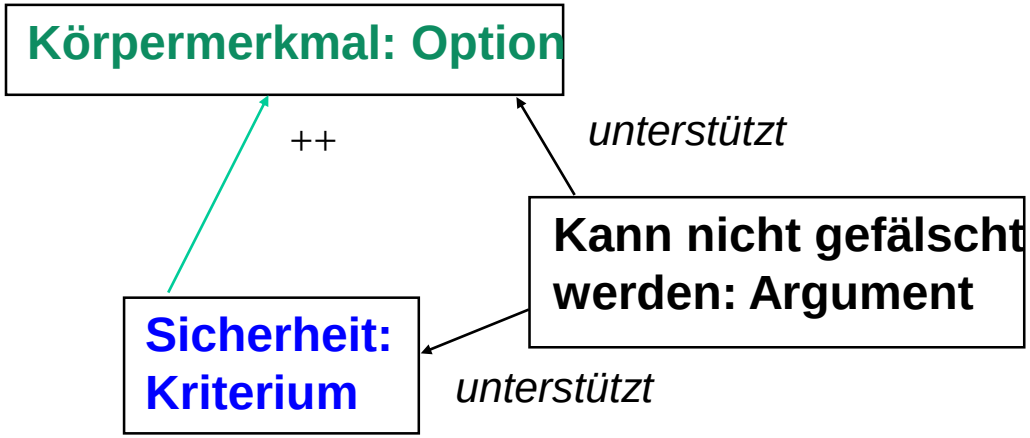
Kriterien und Bewertung

- Kriterien sind oft Qualitätsanforderungen



Argumente

- Argumentente kondensieren die Diskussionen
- Bilden den umfangreichsten Teil des Rationale



Entscheidungen

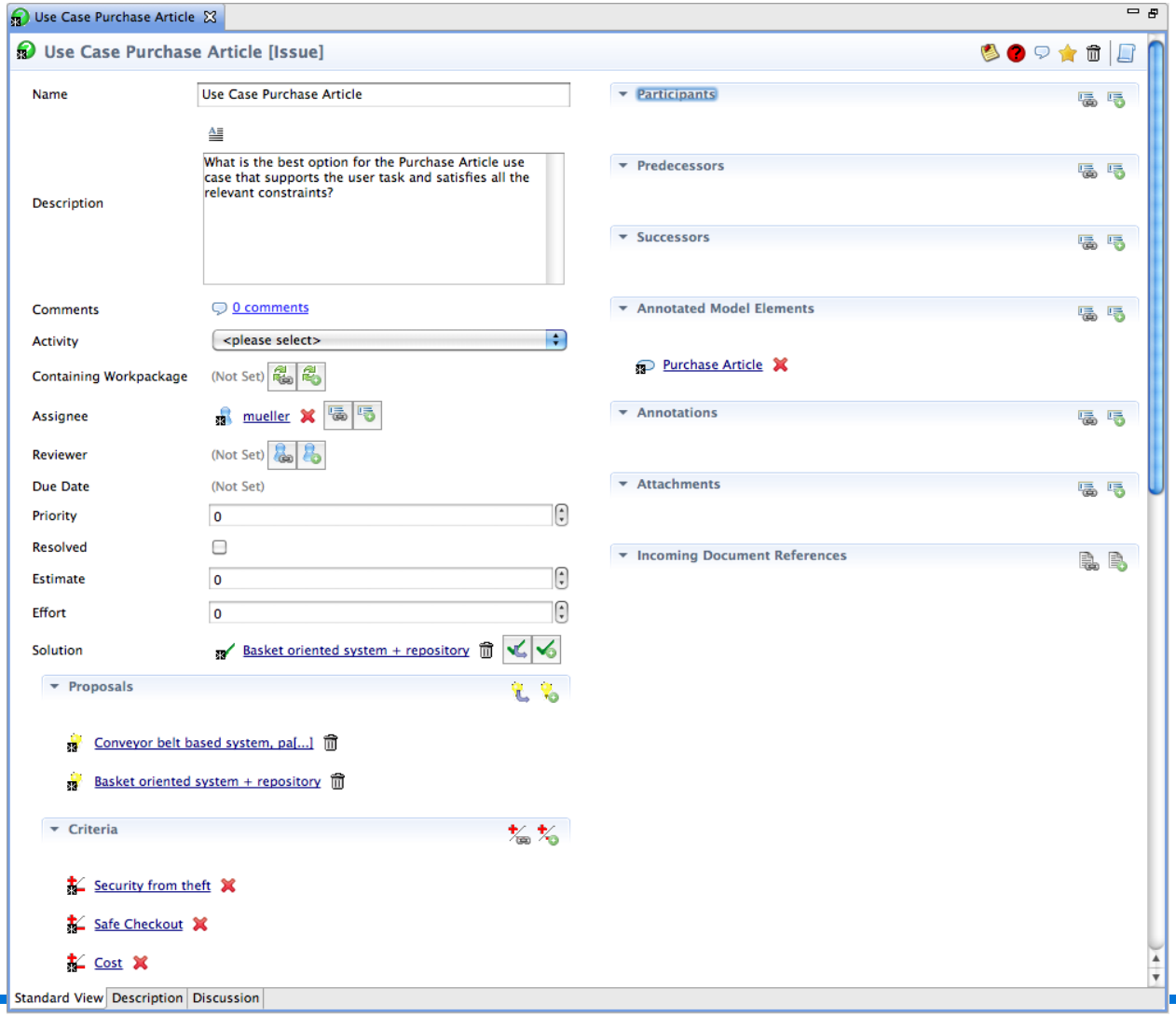
- Eine Entscheidung bezieht sich auf ein oder mehrere *offene* Fragen
- Fasst die gewählten Optionen und die unterstützenden Argumente zusammen
- Daraufhin gelten die Fragen als *geschlossen*
- Kann auch revidiert werden. Dann sind die entsprechenden Fragen wieder geöffnet.

Rationale-Darstellung

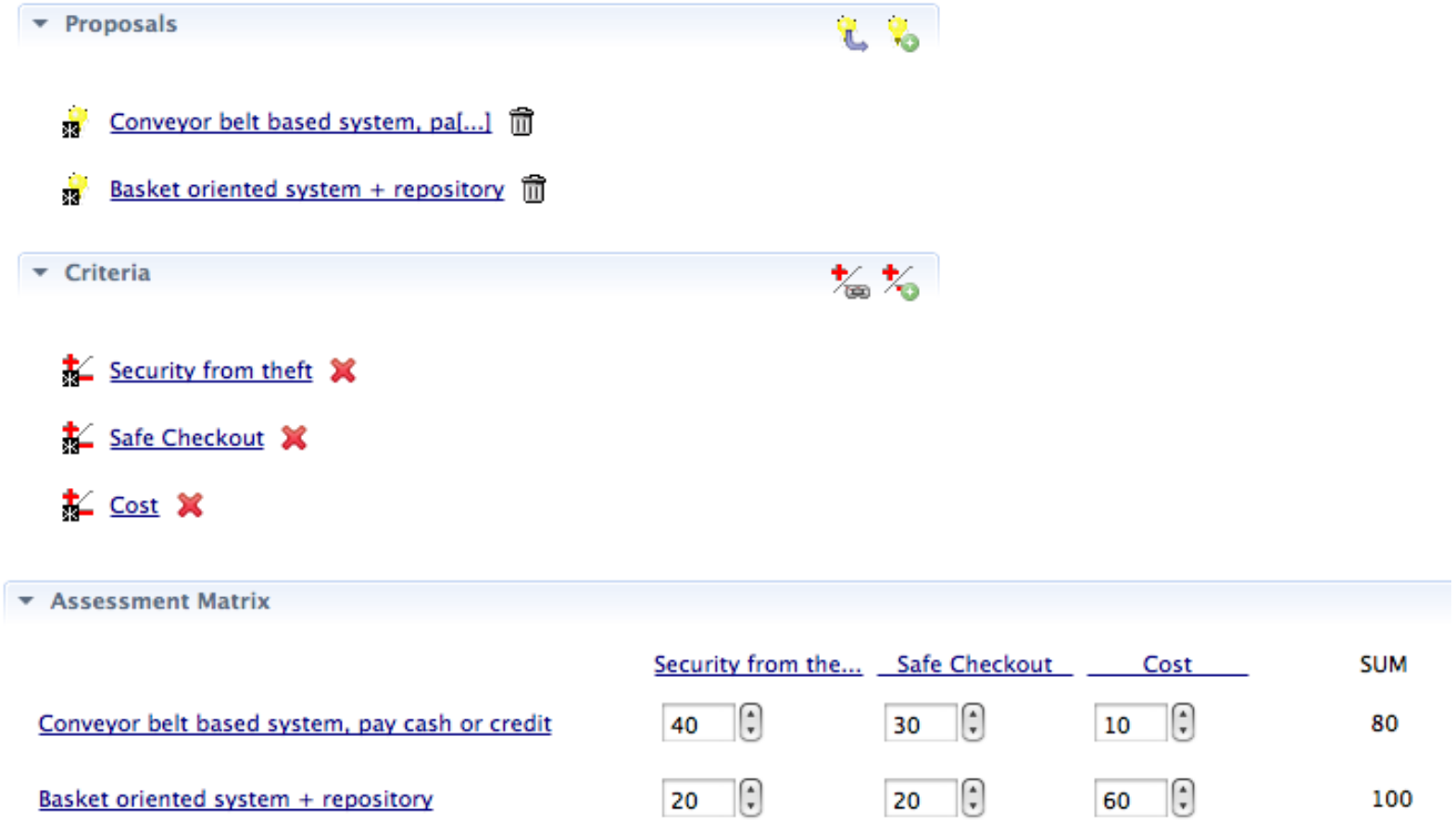
- Rationale gut als Tabelle darstellbar
  - Argumente mit Bewertung verlinken
  - Entscheidung durch gewählte Option sichtbar machen

	Flexibilität	Geringe Kosten	Sicherheit
PIN / Kontonummer	--	++	--
Karte/ PIN	+	+	-
Körpermerkmal	++	--	++

Rationale in Unicase (1)



Rationale in Unicase (2)



Wie erfasse ich Rationale?

- implizit: Rationale in Gesprächsnotizen, Protokollen etc. versteckt
- später: wird nach eigentlicher Entwicklung zusammengestellt, enthält keine Alternativen
- kontinuierlich: während der Entwicklung, Überarbeitung nach der eigentlich Entwicklung
- integriert: auch Überarbeitung während der Entwicklung

Probleme des Rationale

- Aufwand zur Erfassung ist hoch, Rationale-Nutzen nicht sofort sichtbar
  - muss gut motiviert werden
- Aufwand zur Konsolidierung ist noch höher
  - eigene Rolle für Rationale-Wartung
- vollkommen freie Erfassung nicht ausreichend
  - Vorgaben nicht zu streng, aber auch nicht zu locker
- Werkzeuge zur Erfassung nicht gut in den Entwicklungsprozess integriert
  - neue Werkzeuge entwickeln (siehe Unicase)
- Interaktion über Werkzeug ist oft kompliziert
  - Groupware-Elemente einbauen
- Information, Nutzung ist komplex
  - gute Sichten, Filter- und Suchmöglichkeiten

Vorteile des Rationale (1)

- Unterstützt Zusammenarbeit
  - Koordinierung, da Entscheidungen untereinander transparent
  - Fokussierte Diskussion, da Optionen und Kriterien verschiedene Sichtweisen transparent machen
  - Mitarbeit, da gezielt Fragen gestellt werden können
  - Konsens, da nachvollziehbare Entscheidungen getroffen werden

- Unterstützt Wiederverwendung / Änderungen
  - Folgen von Änderungen besser abzuschätzen
  - Folgen von Wiederverwendung besser abzuschätzen

Vorteile des Rationale (2)

- Unterstützt Qualität
  - Konsistenz, da Kriterien explizit werden
  - Nachvollziehbarkeit von Verfeinerungen (z.B. Verbindung von Anforderungen zu Entwurfselementen)
  - Wartung, da Folgen besser abschätzbar

- Unterstützt Wissenstransfer
  - Aus der Vergangenheit lernen, da auch Fehlentscheidungen besser erkennbar und nachvollziehbar
  - Konsolidierung von Begründungen zu Mustern
  - Einarbeitung neuer MitarbeiterInnen
  - Langzeitgedächtnis

Wichtige Einsatzgebiete

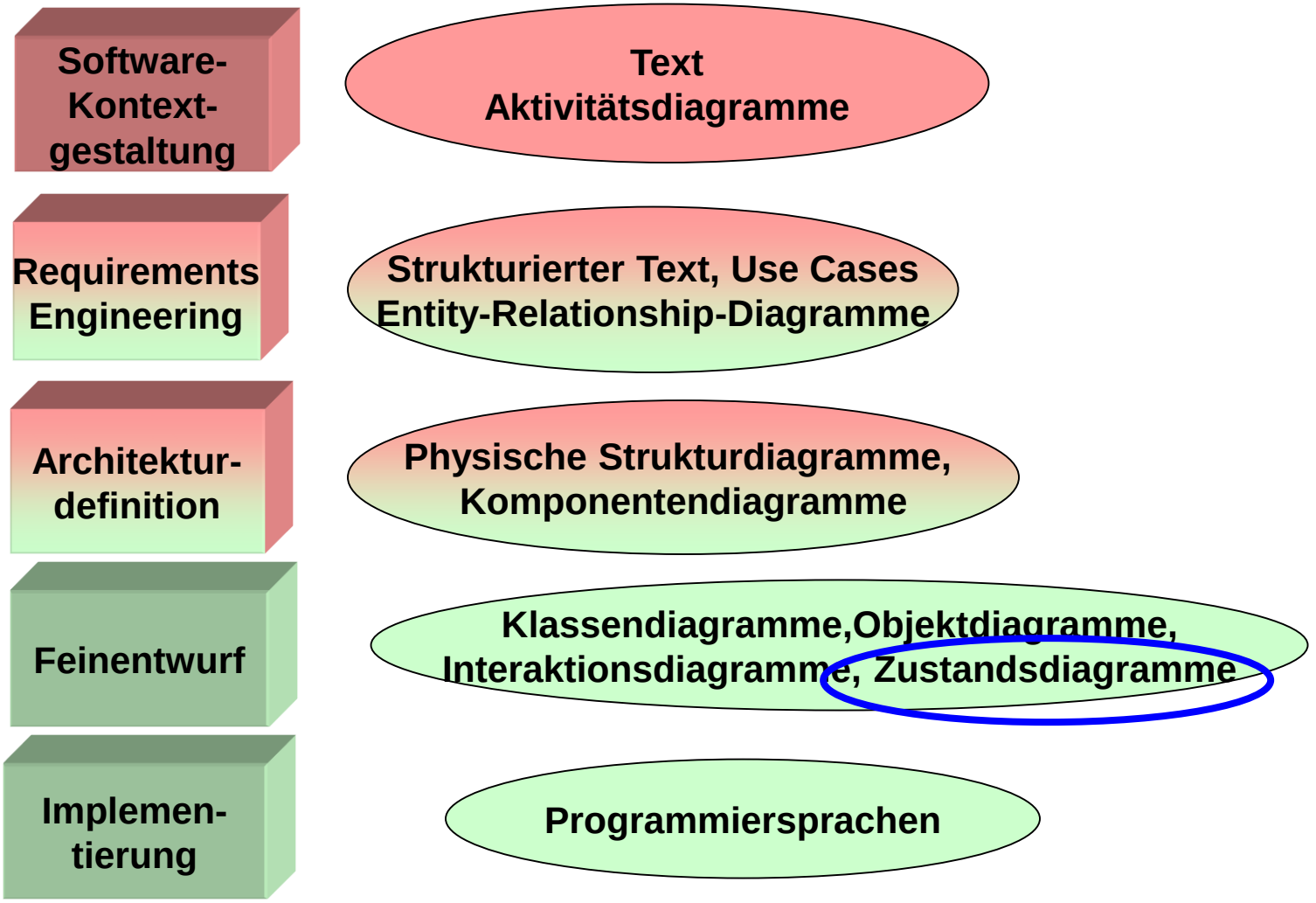
- Besonders wichtig in
  - Verteilten Projekten
  - Projekten zur Erstellung von langfristig wiederverwendbaren Komponenten (Produktlinien, COTS, Services)
  - Sicherheitskritischen Systemen

Literatur Rationale

- Dutoit A, Paech B (2002) Rationale Management, in Handbook of Software and Knowledge Engineering, World Scientific Publishing

- Dutoi, A, McCall R, Mistrik I, Paech B (2006) Rationale Management in Software Engineering, Springer Verlag

Wdh. Beschreibungstechniken



4.4. Zustands-diagramme

- UML Zustandsdiagramm
- Dialogmodell

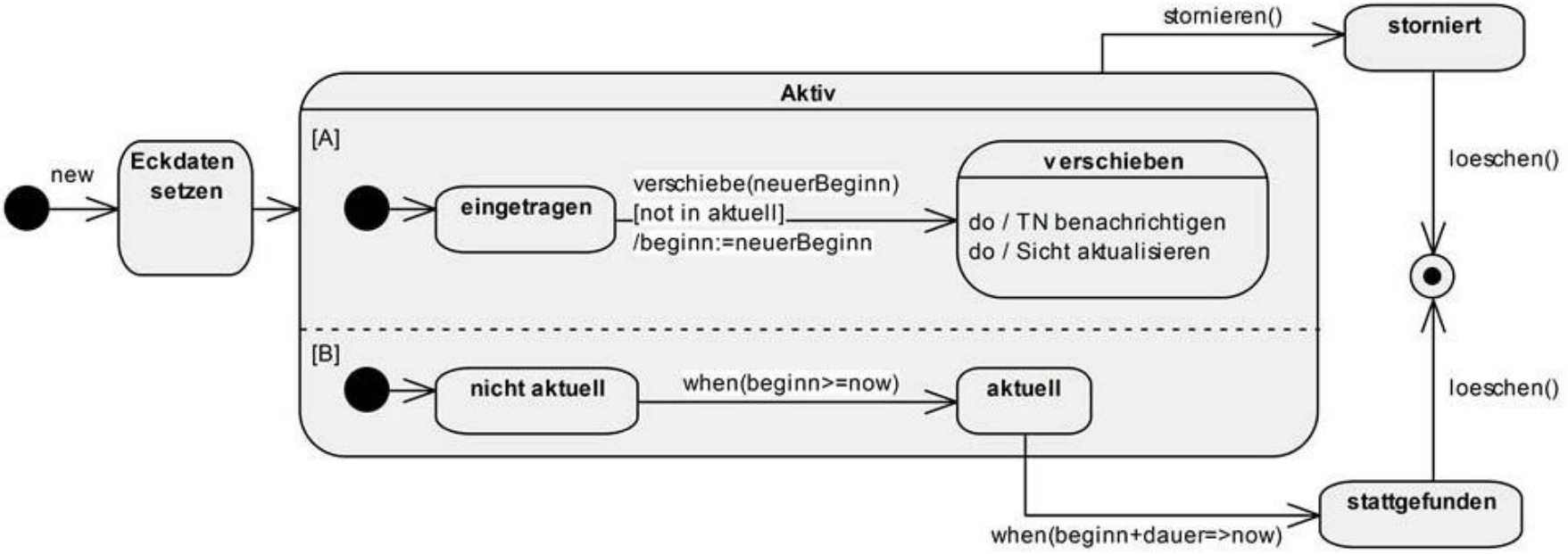
Wichtige Elemente

- Zustandsübergangsdiagramme beschreiben Verhalten
- Bestehen aus
  - Definiertem Anfangszustand
  - Ein oder mehreren Endzuständen
  - Zuständen zur Beschreibung des Verhaltens zu einem bestimmten Zeitpunkt
  - Transitionen (zeitloser Übergang von einem Zustand zum anderen)
- Wichtige Elemente
  - Zustände, Start/Endzustand , Transitionen (Ereignis, Guard, Aktion), interne Transitionen, komplexe Zustände

- Siehe Folien TU Wien
- http://www.uml.ac.at/de/lernen

[Folien Kappel, S.25]

Beispiel mit komplexem Zustand



Welche Elemente?

Welche Abläufe, z.B: Ablauf mit Möglichst viel verschiedenen Zuständen?

Hausaufgabe Zustandsdiagramm

- Was sind die wichtigsten Elemente der Zustandsdiagramme?
  - **Zustand**: Normaler Zustand, Endzustand
  - **Zustandsübergang**: Ereignis, Bedingung, Aktion
  - **Pseudozustände**: Startzustand, Entscheidung, Parallelisierung, Synchronisierung
  - **Transitionen**: hängen von Art der Ereignisse ab (SignalEvent, CallEvent, TimeEvent, ChangeEvent)
- Welche 3 Arten von internen Aktivitäten gibt es? Unterschiede?
  - **Innere Transition**: Eintrittsaktivität (Nach Betreten eines Zustands), Austrittsaktivität (Nach Verlassen des Zustands), Andauernde Aktivität (nach Eintrittsaktivität ausgeführt)
- Welche Konzepte erlauben die Beschreibung von Zustandshierarchie?
  - **zusammengesetzte Zustände**: Einführung von Subzuständen, Einheitliche Behandlung einer ganzen Gruppe

4.4. Zustands-diagramme

- UML Zustandsdiagramm
- Dialogmodell

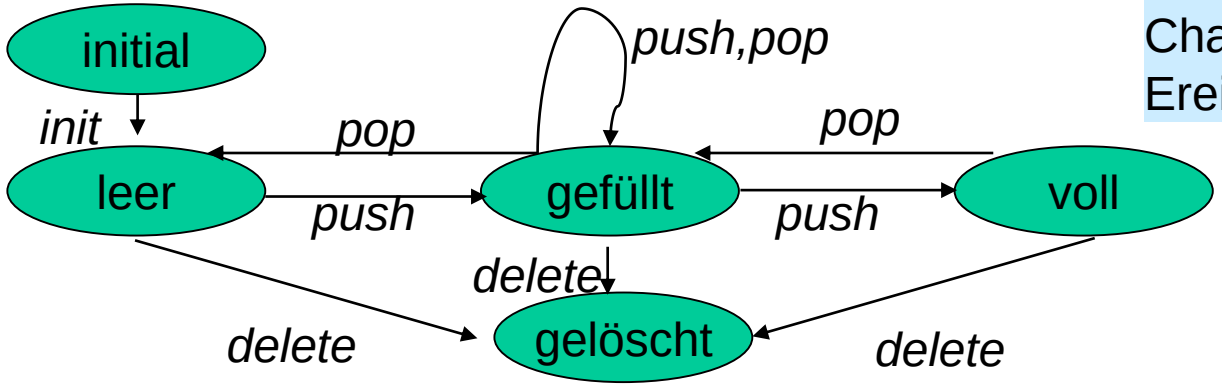
Zustandsdiagramme im SWE

- Zustandsdiagramme können Verhalten auf unterschiedlichen Ebenen beschreiben
  - Verhalten eines Objektes / einer Klasse
    - Siehe Zustandsbezogener Test in Kapitel 2.6.
  - Verhalten einer Komponente/ eines Systems
  - **Verhalten der Benutzungsschnittstelle**
    - **Siehe Dialogmodell (nachfolgend)**

Wdh. Zustandsbezogener Test

- Berücksichtigt neben Ein/Ausgaben auch die Historie (den erreichten Zustand)
- Zustände beschreiben die Vor/Nachbedingungen der Operationen / Ereignisse
- Typisches Beispiel: Stapel
- Zustände: initial, leer, gefüllt, voll, gelöscht
  Ereignisse: init, pop, push, delete

Kontrollzustände:
Zustände sind durch Attributwerte gekennzeichnet
Charakterisieren mögliche Ereignisse



Beschreibung von Dialogen

- Ein **Dialog** beschreibt die Abfolge von Sichten bei der Durchführung einer Aufgabe durch die/den NutzerIn

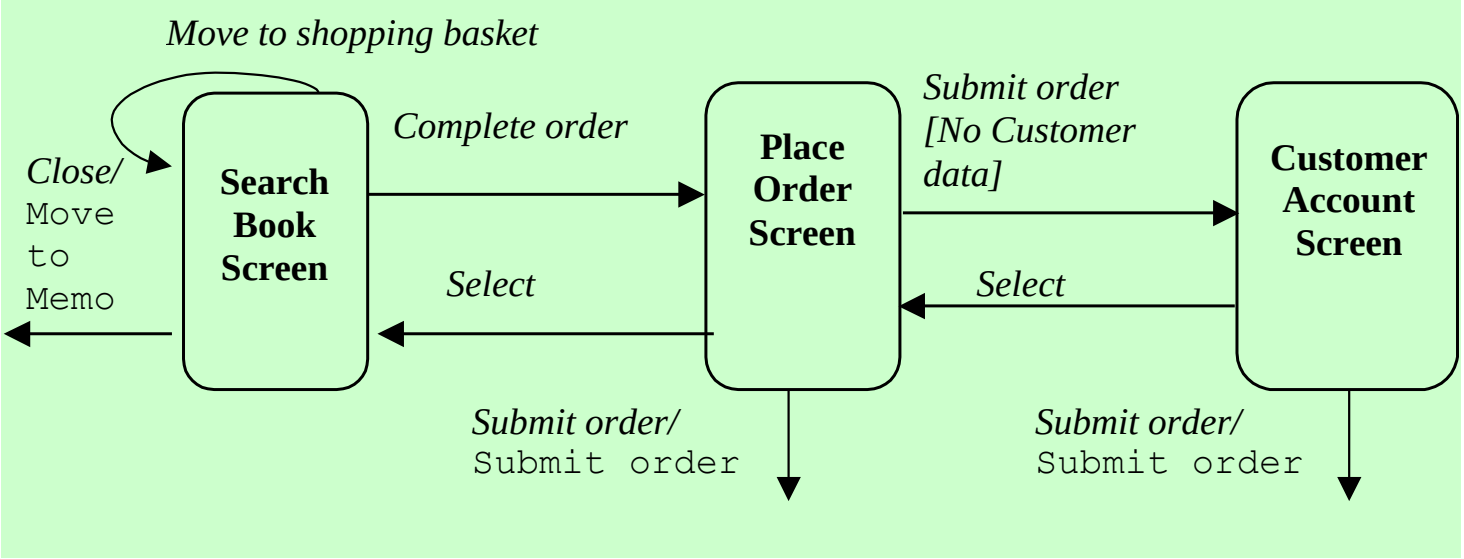
- Dabei sind alle Dialoge, die einen Arbeitsbereich betreffen, zusammen zu betrachten (Menge aller möglichen Abläufe durch die Sichten des Arbeitsbereichs)

- BenutzerInnen sollen für eine Aufgabe möglichst wenig Sichten benötigen
- Betreibe so viel **Wiederverwendung** wie möglich
  - Fasse ähnliche Sichten zusammen
  - Unterteile Sichten (falls notwendig), um „Teilsichten“ besser wiederverwenden zu können

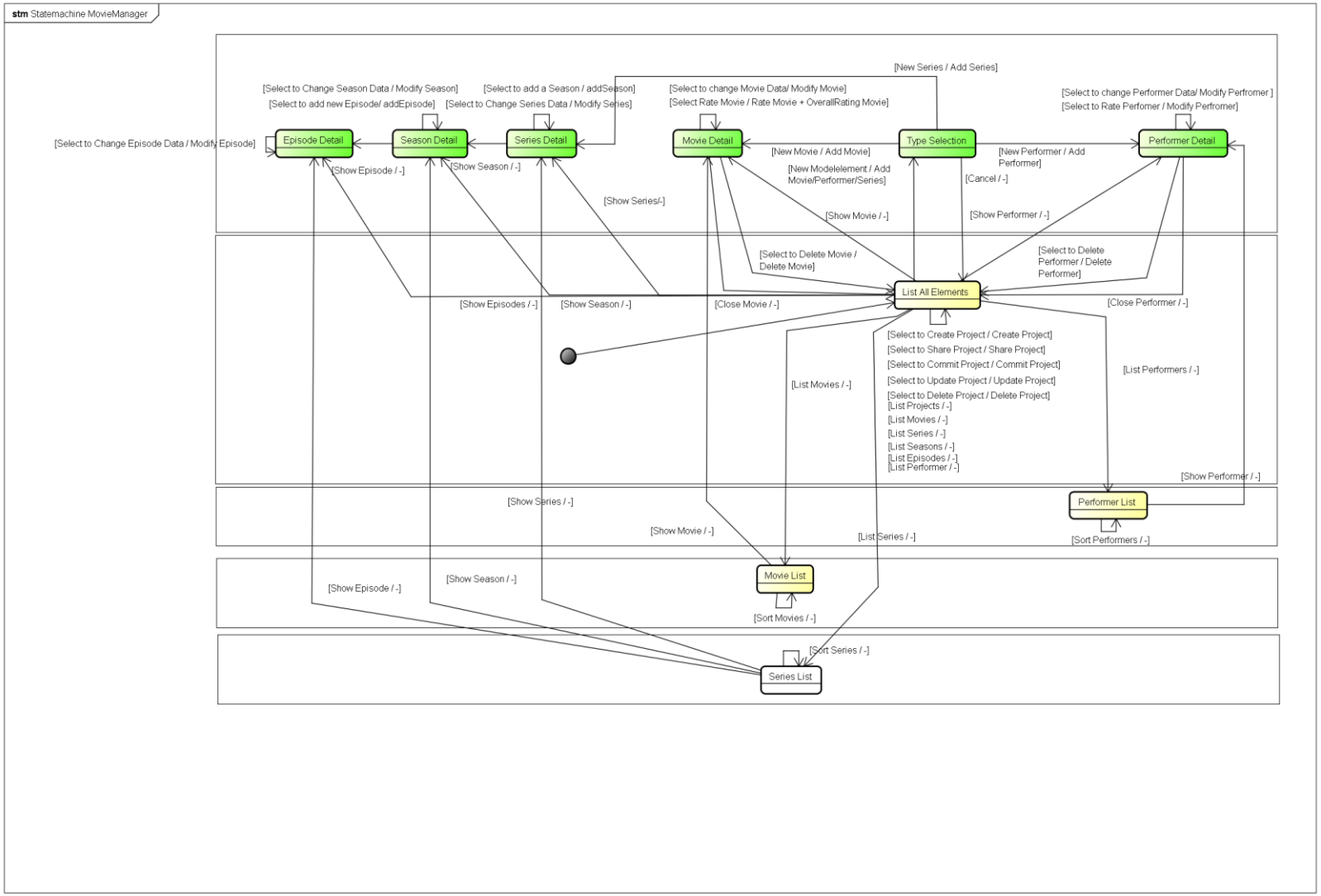
Dialogmodell als Zustandsdiagramm

- Zustand ist Arbeitsbereich oder Sicht
- Transitionen beschreiben Funktionsübergänge
  - *Ereignis = Aktion auf Benutzungsschnittstelle*
  - *Aktion* = semantische Funktion
  - Insgesamt: *Aktion auf Benutzungsschnittstelle [evtl.Bedingung] / semantische Funktion*

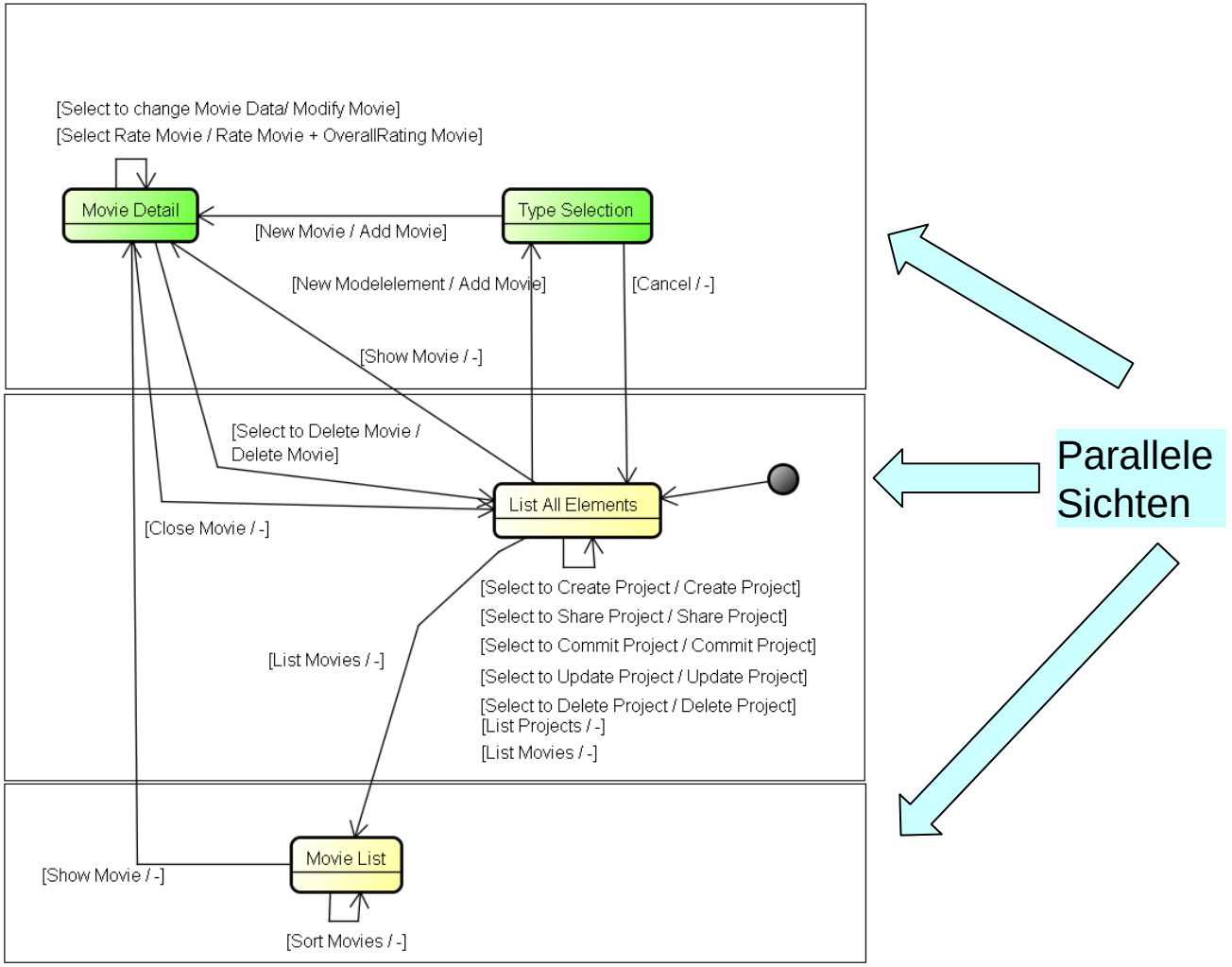
Auch hierarchische Zustände für Parallelität möglich



Beispiel: Dialogmodell der Movie Management Anwendung



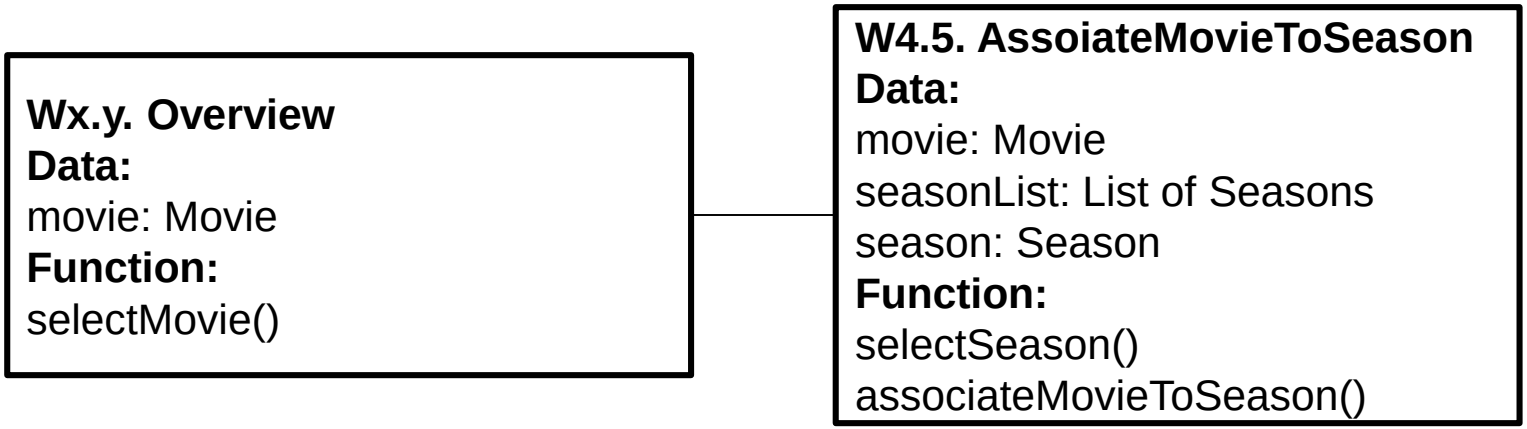
Beispielausschnitt Dialogmodell



Vorgehen zur Ableitung eines Dialogmodells

- Jede Sicht mit wesentlichen Daten entspricht einem Zustand (Benennung analog zu Arbeitsbereich)
  - Sichten, die nur zur Auswahl einer Funktion dienen, können im Dialogmodell weggelassen werden
- Für jede Funktion, die in der Sicht ausgeführt werden kann (siehe auch UI-Struktur), wird eine Transition erstellt zur Sicht, die nach der Ausführung zu sehen ist.
  - Dabei Unterscheidung, ob semantische Funktion oder nur Hilfsfunktionen an der Oberfläche
  - Hilfsfunktionen, die die anzuzeigenden Daten bei der Rückkehr in eine Ausgangssicht beschreiben, können weggelassen werden (z.B. Anzeige der Staffel mit einer neuen Episode nach Hinzufügen der Episode)

Beispiel: Dialogmodell für AssociateMovieToSeason



- Arbeitsbereich *W4.5. AssociateMovieToSeason*
- Systemfunktion *AssociateMovieToSeason():* erstellt aus dem Film eine Episode, verlinkt diese mit der Staffel und löscht den Film
- Sichten siehe Arbeitsblatt 9

Zusammenfassung Zustandsmodellierung

- Wichtig zur Beschreibung von Folgen von Zuständen (Kontroll- oder Datenzustände)

- Im SWE vielfältig einsetzbar:
  - Klassenentwurf (Verhalten einer Klasse)
  - UI-Entwurf (Dialogmodell)
  - Testen (zustandsbasierte Testfallableitung)
  - Insbesondere auch im Bereich eingebetteter Systeme (Steuergeräte)
  - Dabei oft auch spezielle Notationen/Erweiterungen

4.8. Zusammenfassung Modellierung und Kommunikation

Modelle und Kommunikation

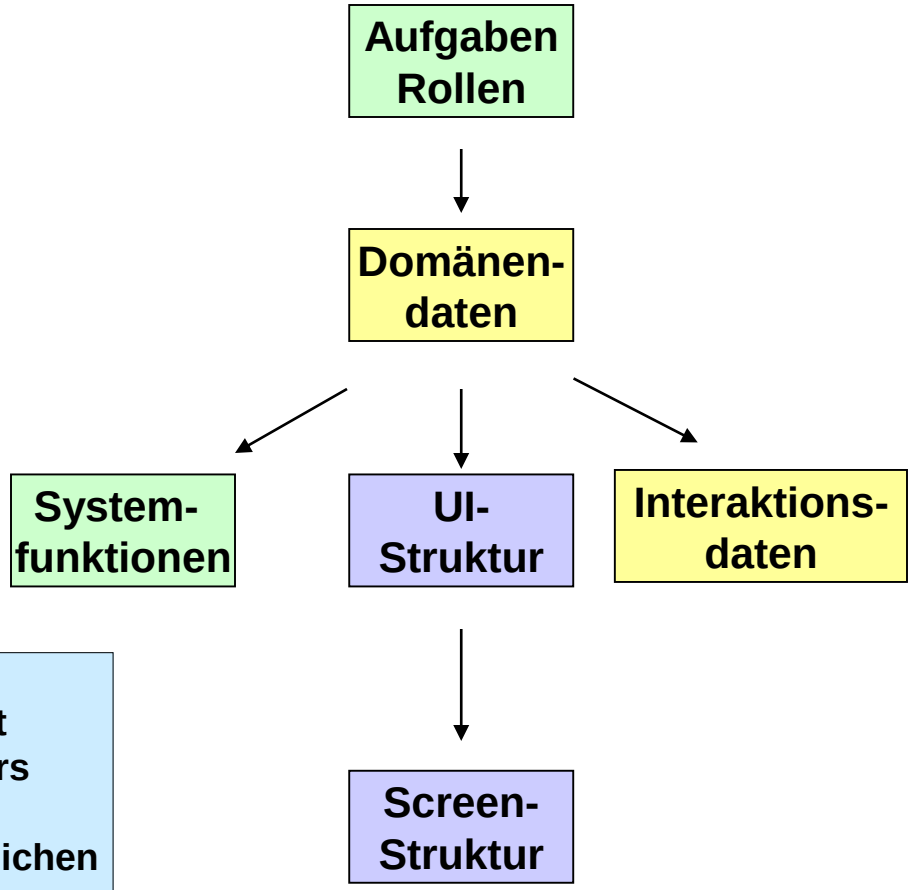
- UML-Diagramme ermöglichen Beschreibung des Systems und Kommunikation von EntwicklerInnen darüber

- Modelle machen wichtige statische (Klassendiagramm) und dynamische (Interaktionsdiagramm, Zustandsdiagramm) Aspekte deutlich, ohne den Code festzulegen

Kommunikation von Anforderungen und Entwurf

- Der Übergang zwischen Kommunikation zu den KundInnen bzw. Kommunikation unter den EntwicklerInnen ist fließend.
- Insbesondere UI-bezogene Artefakte sind
  - Einerseits Entwurfsartefakte, die die EntwicklerInnen gestalten
  - Andererseits Anforderungsartefakte, da die KundInnen am UI gut sehen und bewerten können, wie die EntwicklerInnen die Anforderungen umsetzen wollen.
- Von der Interaktionsebene ist der Übergang zum Entwurf (Systemebene) gut möglich
  - OOAD überführt Systemfunktionen, Interaktionsdaten und UI-Struktur in einen Klassenentwurf
  - Dialogmodelle verfeinern die UI-Struktur und zeigen den Zusammenhang der Sichten

Wdh. Anforderungsgestaltung



Aufgaben ebene

Domänen ebene

Interaktions ebene

Systemebene: GUI

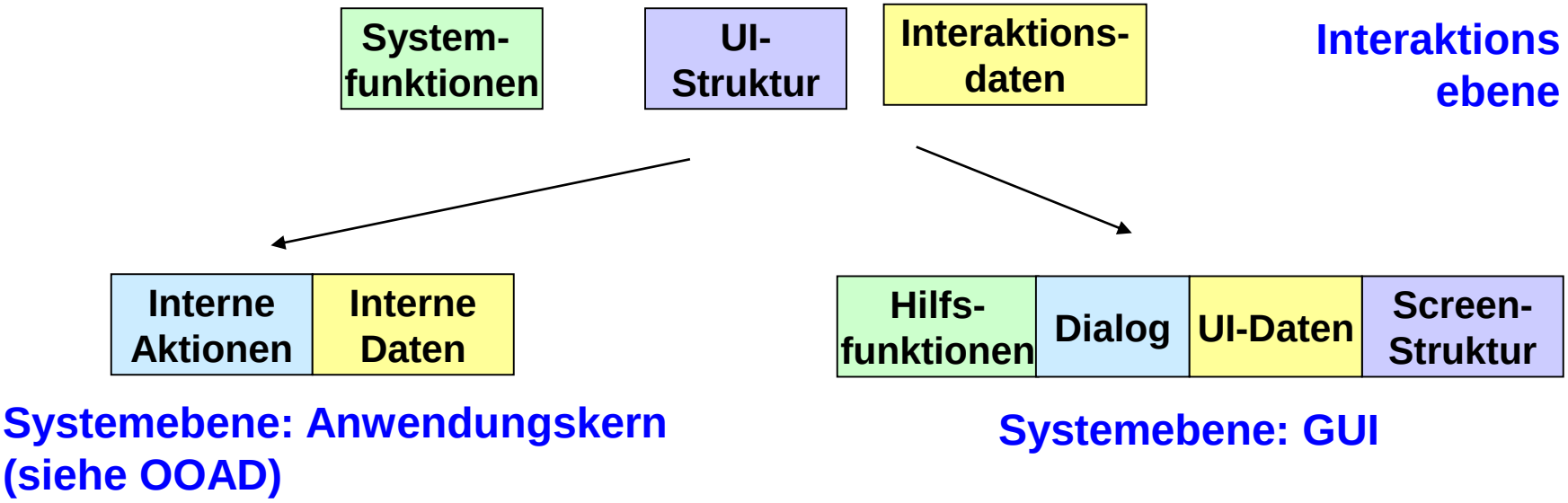
Legende:
Grün: Aufgaben
Gelb: Daten
Lila: Struktur

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

Entwurfsgestaltung

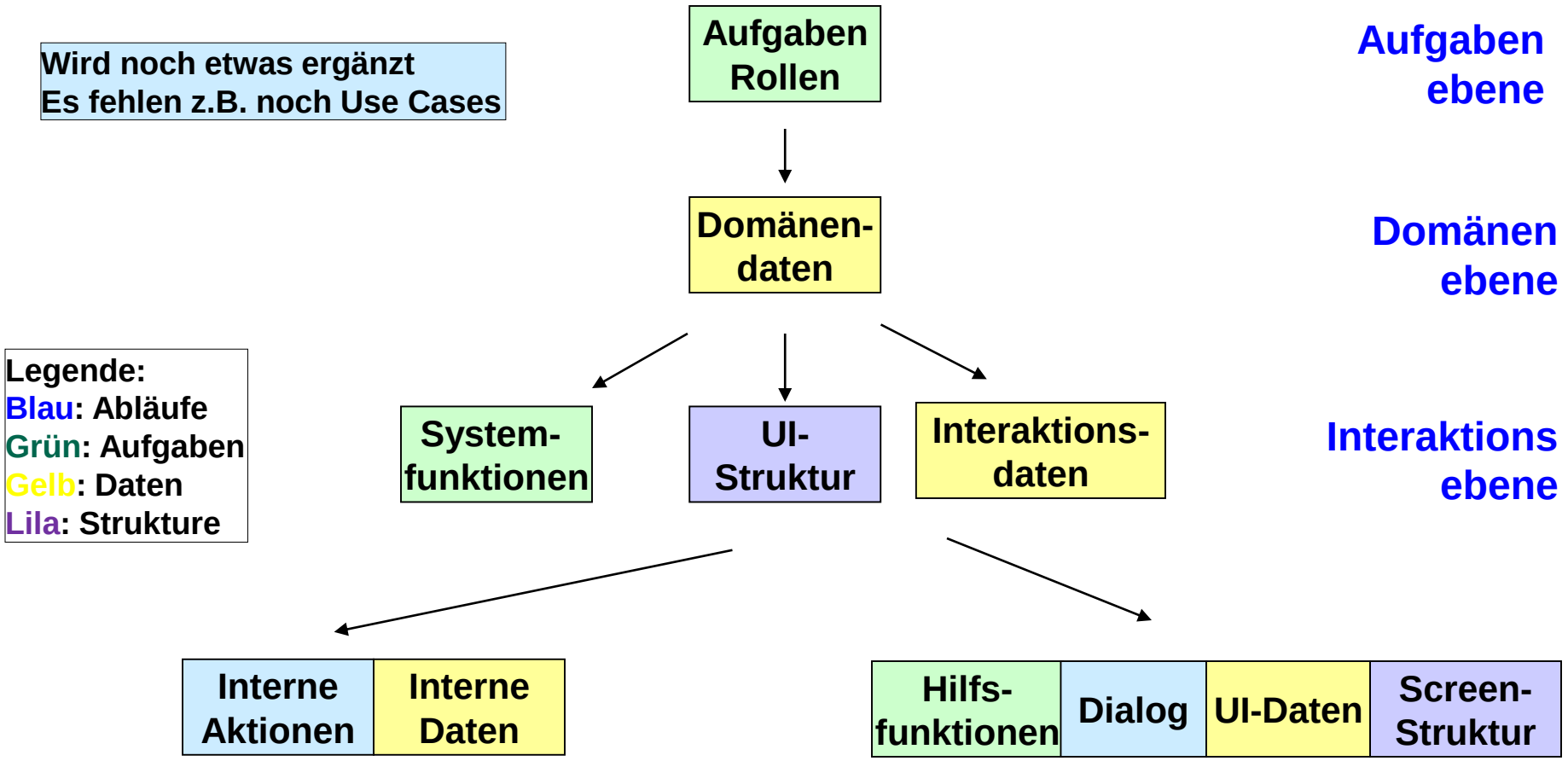
Legende:
Blau: Abläufe
Grün: Aufgaben
Gelb: Daten
Lila: Strukture

- Auf der Systemebene wird der Entwurf beschrieben



Insgesamt: Anforderungs- und Entwurfsgestaltung

Wird noch etwas ergänzt
Es fehlen z.B. noch Use Cases



Aufgaben ebene

Domänen ebene

Interaktions ebene

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

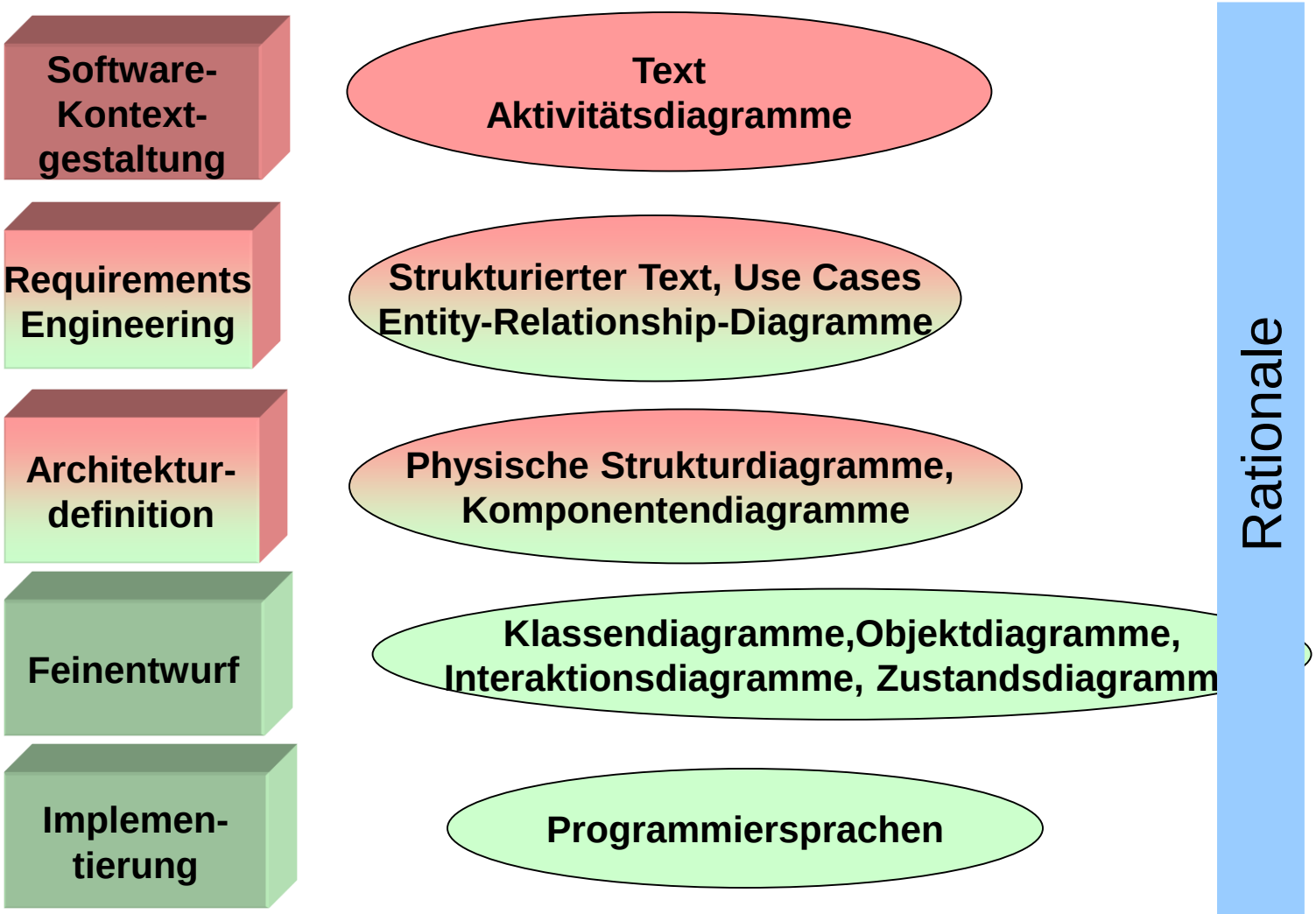
Systemebene: Anwendungskern (siehe OOAD)

Systemebene: GUI

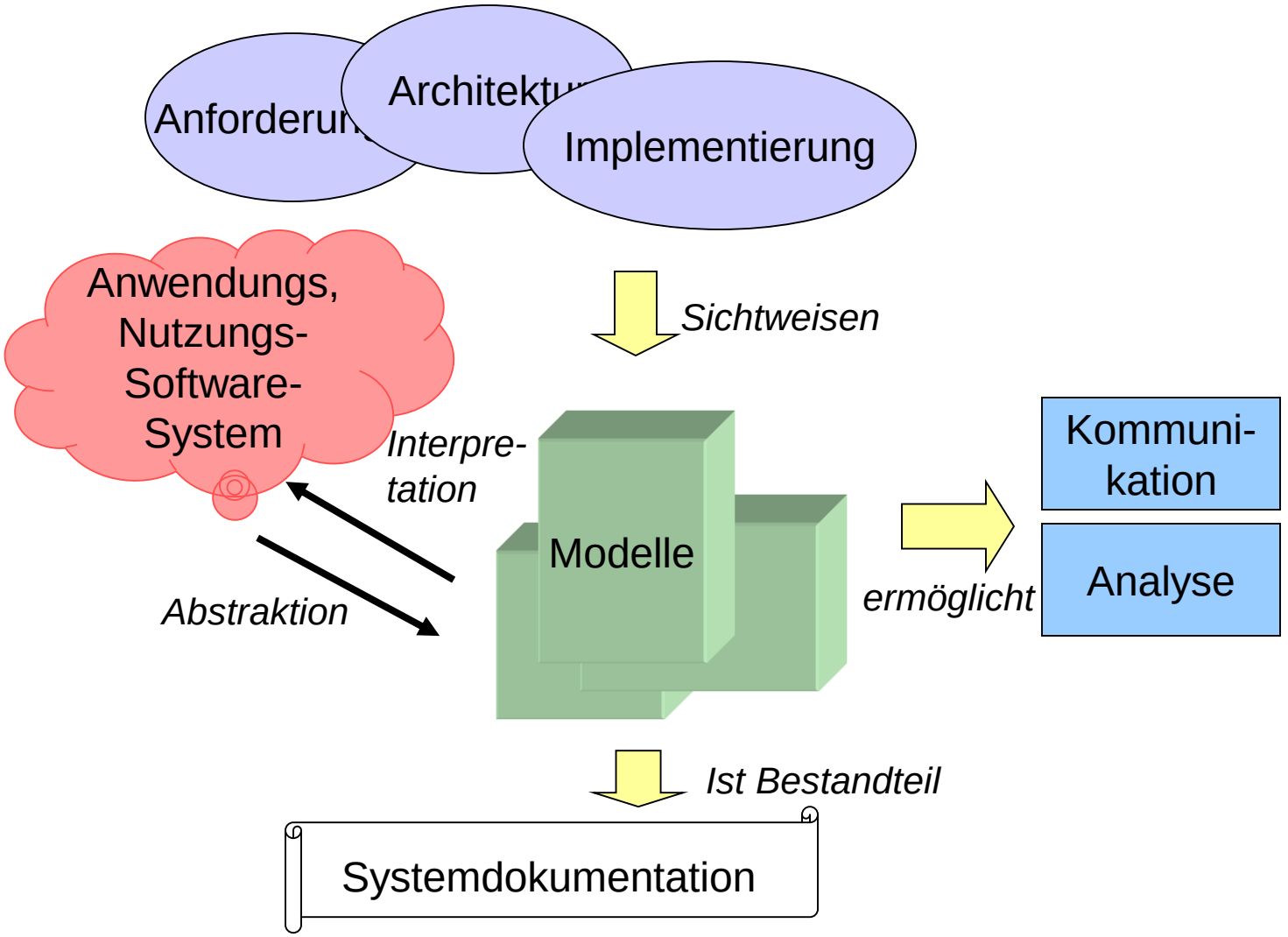
Überblick Modellierung

- Modellierungstechiken stellen eine Notation zur Verfügung, um Ausschnitte der (IST- oder SOLL-) Welt zu beschreiben.
- Die Notation repräsentiert eine bestimmte Auswahl von **Modellierungskonzepten**.
- Die Auswahl einer Modellierungstechnik sollte **systematisch** erfolgen.

Wdh. Beschreibungstechniken



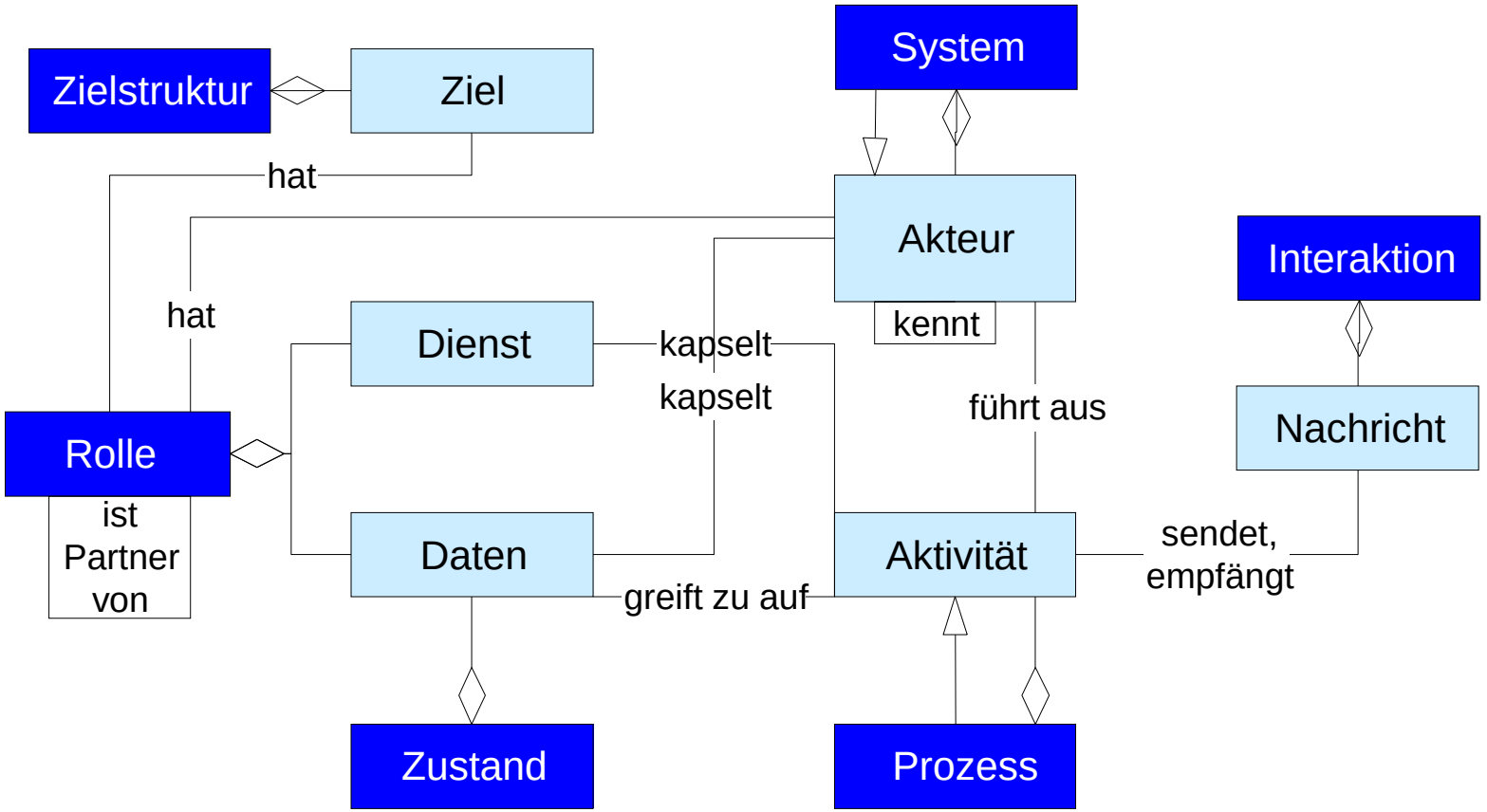
Wdh. Modelle in der Systementwicklung



Welche Technik wofür?

- Jede Technik fokussiert auf bestimmte Systemkonzepte
  - Akteur
  - Aktivität (intern, Teil eines Dienstes)
  - Nachricht (ausgetauscht zwischen Akteuren)
  - Daten (verwaltet von Akteur)
  - Dienst (angeboten nach außen von Akteur)
  - Ziel

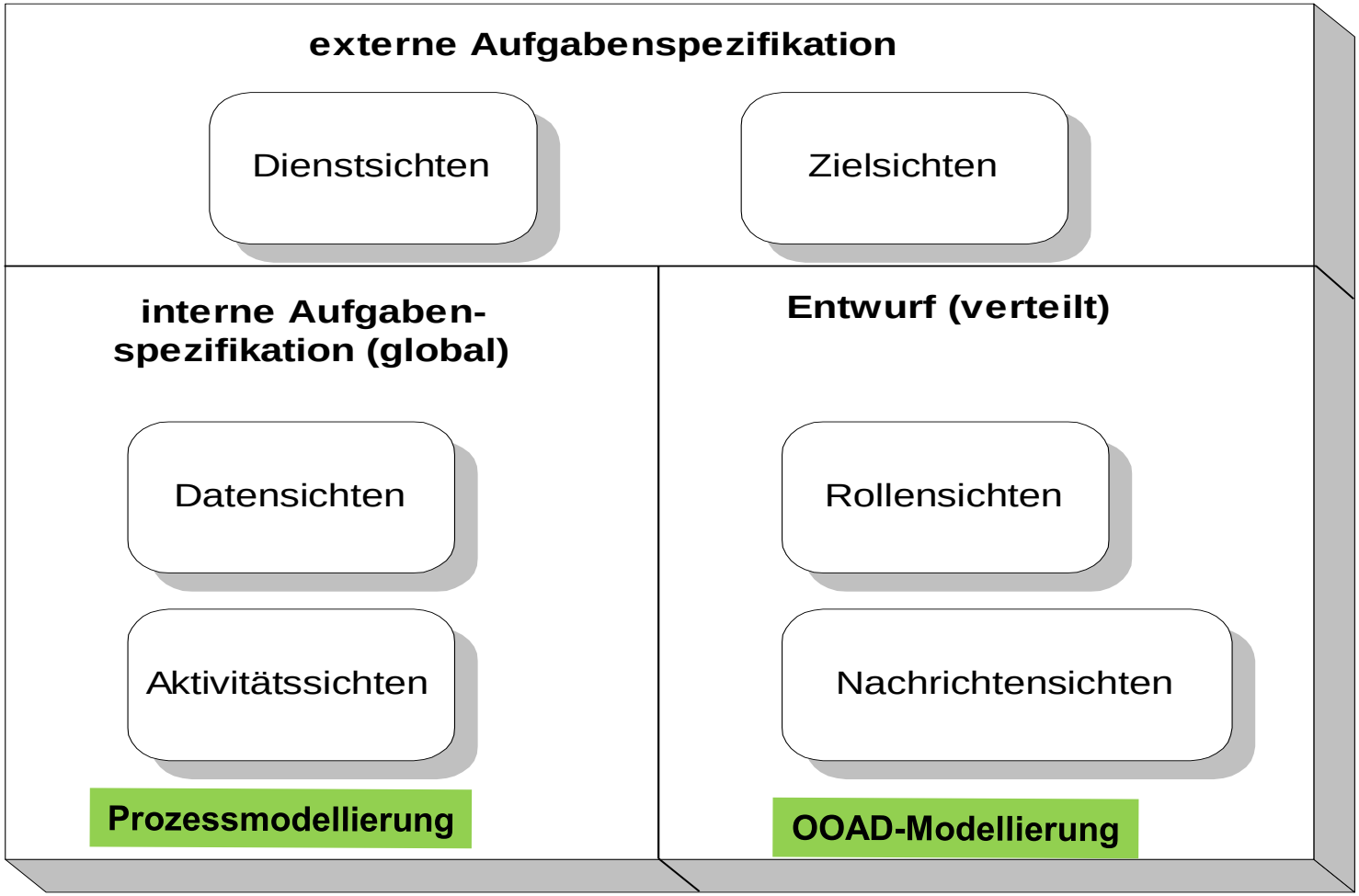
Grundlegende Systemkonzepte



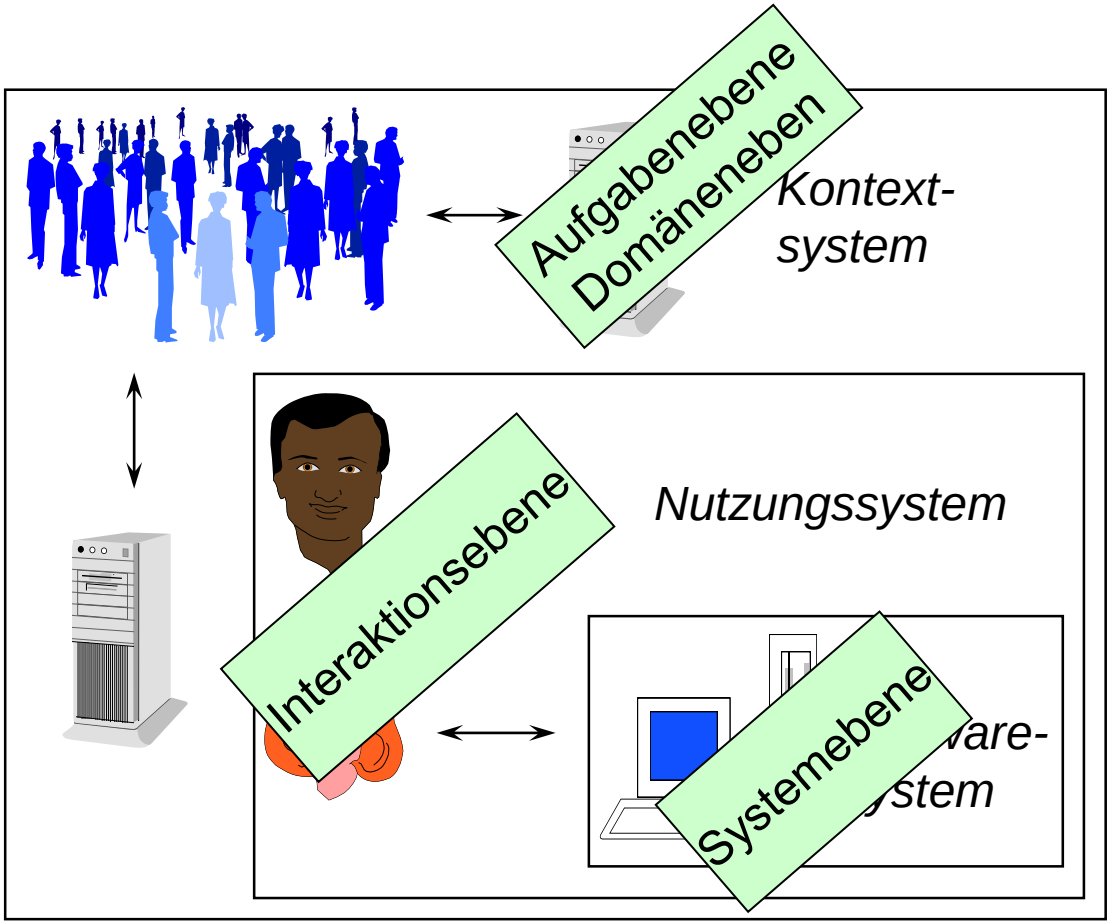
Fokus der Modellierungstechniken

Konzept	Diagramm
Daten-Struktur	Entity-Relationshipsdiagramm
Datenzustands-Folgen	(Daten-)Zustandsdiagramm
Aktivitäts-Struktur	Datenflussdiagramm
Aktivitäts-Folgen (Prozess)	Aktivitätsdiagramm, Petrinetz,…
Dienst-Struktur	Nutzungsdiagramm
Dienst(aufrufs)-Folgen	(Kontroll-)Zustandsdiagramm
Rollen-Struktur	Klassendiagramm
Rollenverhalten	(Kontroll-)Zustandsdiagramm
Nachrichten-Struktur	Objektmodell, Use Case Text
Nachrichten-Folgen	Sequenzdiagramm, Kommunikationsdiagramm
Ziel-Struktur	Zielstrukturdiagramm

Aufgabenorientierte Systemmodellierung



Wdh. Gestaltungsbereiche der SW-Entwicklung



Zuordnung der Modelle

- Externe Sicht auf Anwendungskontext: Rollen, Aufgaben im Unternehmen, Geschäftsziele
- Interne Sicht auf Anwendungskontext: IST/SOLL-Aktivitäten, Domänendaten
- Entwurf der Schnittstelle System/Kontext: Use cases (Interaktion Mensch/Maschine), keine expliziten Rollen

- Externe Sicht auf Software: Systemfunktionen, Qualitätsanforderungen
- Interne Sicht in Bezug auf Software: Interaktionsdaten, keine expliziten Aktivitäten
- Entwurf der Softwarestruktur: Klassendiagramme, Interaktionsdiagramme

Zusammenhänge der Modelle eines Projektes

- Vielfältige Beziehungen zwischen Modellen müssen während des SWE verwaltet bzw. berechnet werden können
  - Strukturelle Abhängigkeit
    - ist Teil/Erweiterung von, ist Ergänzung zu, importiert oder nutzt Elemente von,
    - z.B: Sequenzdiagramm importiert Klassendiagrammelemente
  - Kausale Abhängigkeit
    - wird benötigt für/stützt sich auf, ist Vorversion von, ist Beispiel für, ist Prüfergebnis von
    - Z.B. Analyseklassendiagramm stützt sich auf ER-Diagramm
  - Semantische Beziehung
    - ist Übersetzung von/ist Quelle von, ist Spezifikation von/ist Implementierung von, ist Abstraktion von/ist Detaillierung von, ist durch Transformation entstanden aus
- => Werkzeugunterstützung (Traceability) ist nötig

Literatur

- B. Paech, *Aufgabenorientierte Softwareentwicklung*, Springer Verlag 2000

Zusammenfassung

- Modellierungstechniken sind notwendig um komplexe Zusammenhänge übersichtlich darzustellen
- Notation muss dem Zweck angemessen sein
- Modelle sind Entwicklungsergebnisse und sind damit (genau wie Code) systematisch zu entwickeln und weiterzuverarbeiten