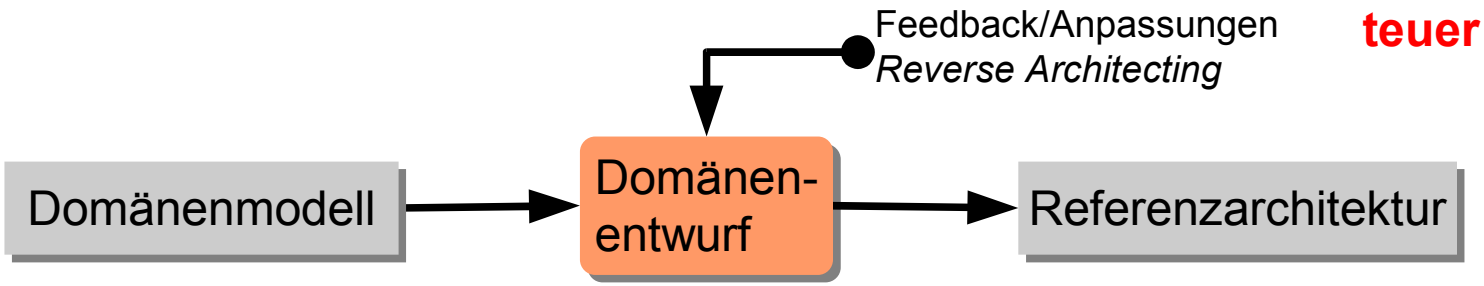
Betriebssystemtechnik

Operating System Engineering (OSE)

Architekturmodell: der Weg zur Klassen-Hierarchie

Domänenentwurf



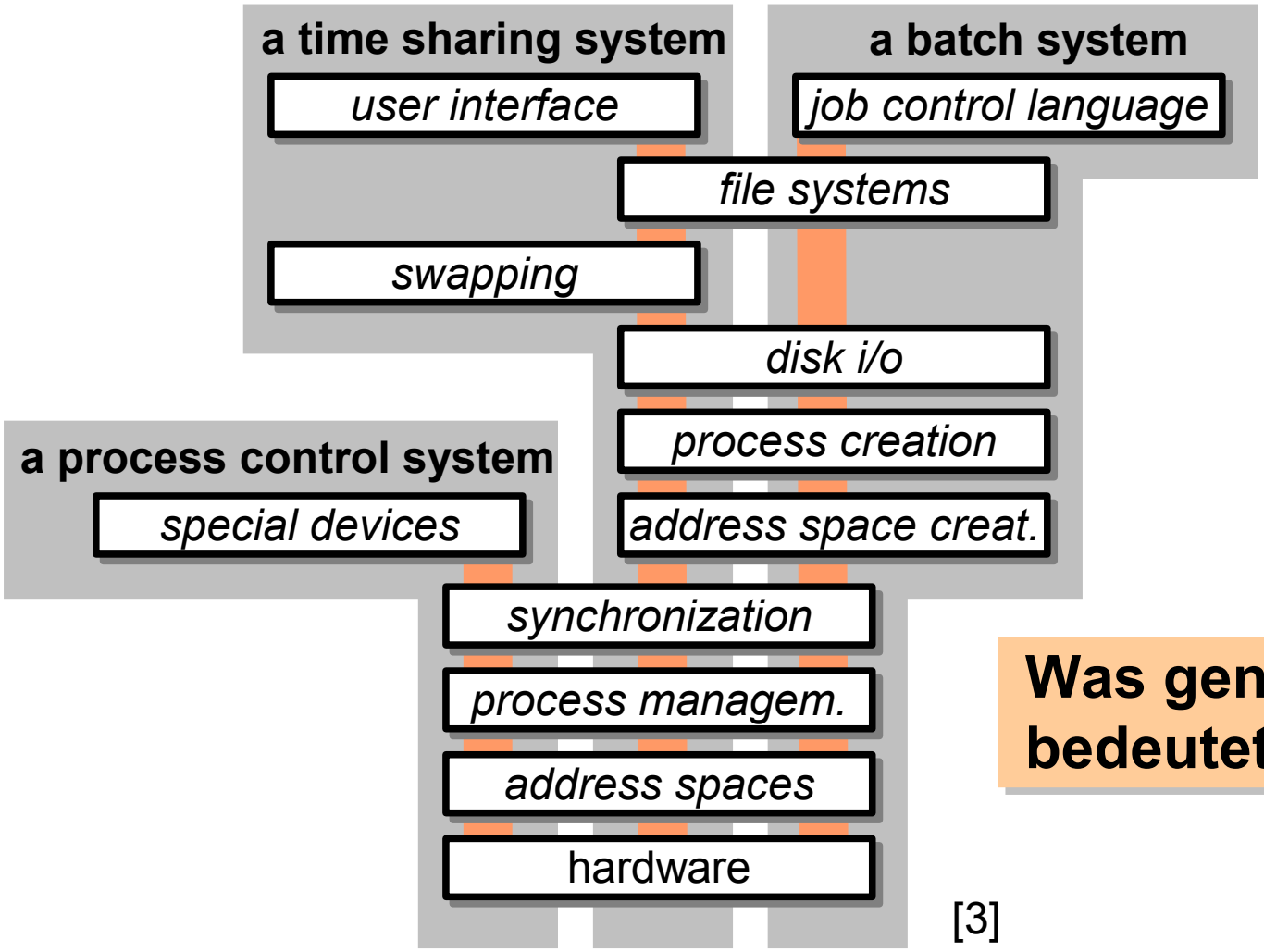
- **Analyse**
  - Identifikation der Architekturtreiber, Mechanismen und Sichten
- **Modellierung**
  - **... ? ...**
  - Architekturdokumentation mit Variationspunkten
- **Evaluierung**
  - Erkennen von Risiken in den Entwurfsentscheidungen

In Anlehnung an das Modell zur Architekturentwicklung aus [1].

Architektur-Modellierung

- bisher haben wir diverse Hierarchien
  - Modul-Hierarchie
  - Funktionale Hierarchie [3]
  - Benutzt-Hierarchie [4]
- jetzt kommt noch eine hinzu: die **Belang-Hierarchie** [1]
- gesucht ist eine Modulstruktur
  - durch Programmiersprache bestimmt: AspectC++
    - Klassen, Methoden, **Aspekte**, (Templates)

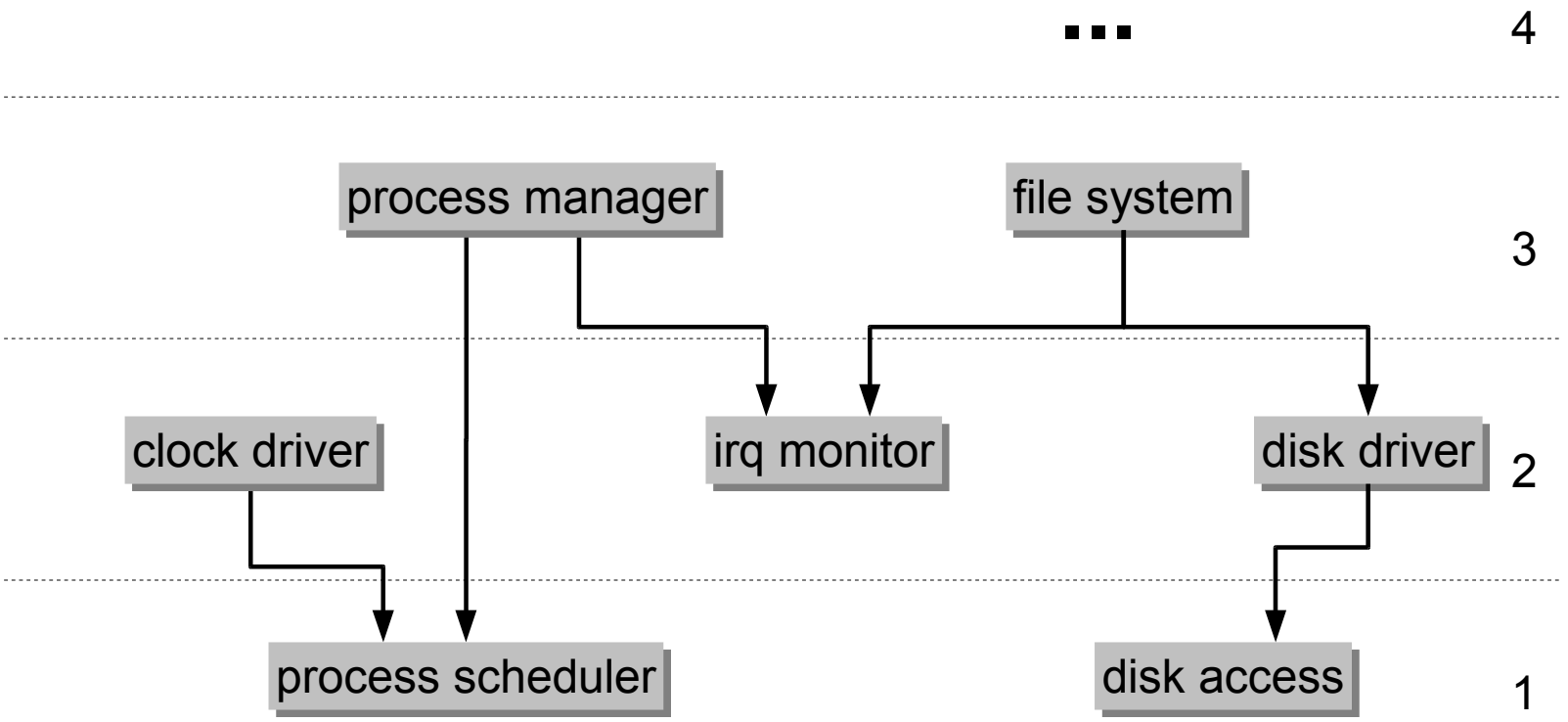
Funktionale Hierarchie von FAMOS



Was genau bedeutet das?

[3]

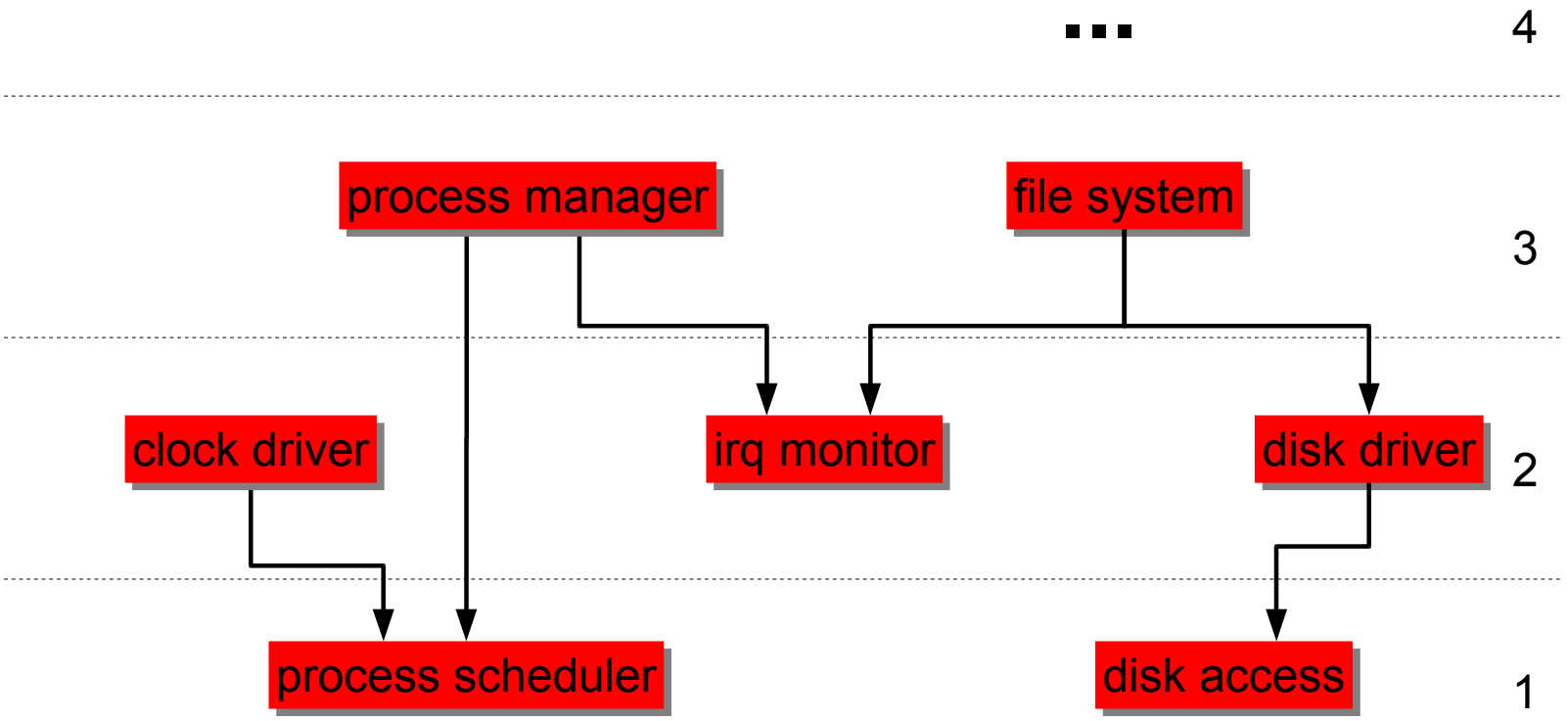
Das Modell der funktionalen Hierarchie



Funktionale Hierarchie – Elemente

Funktionen

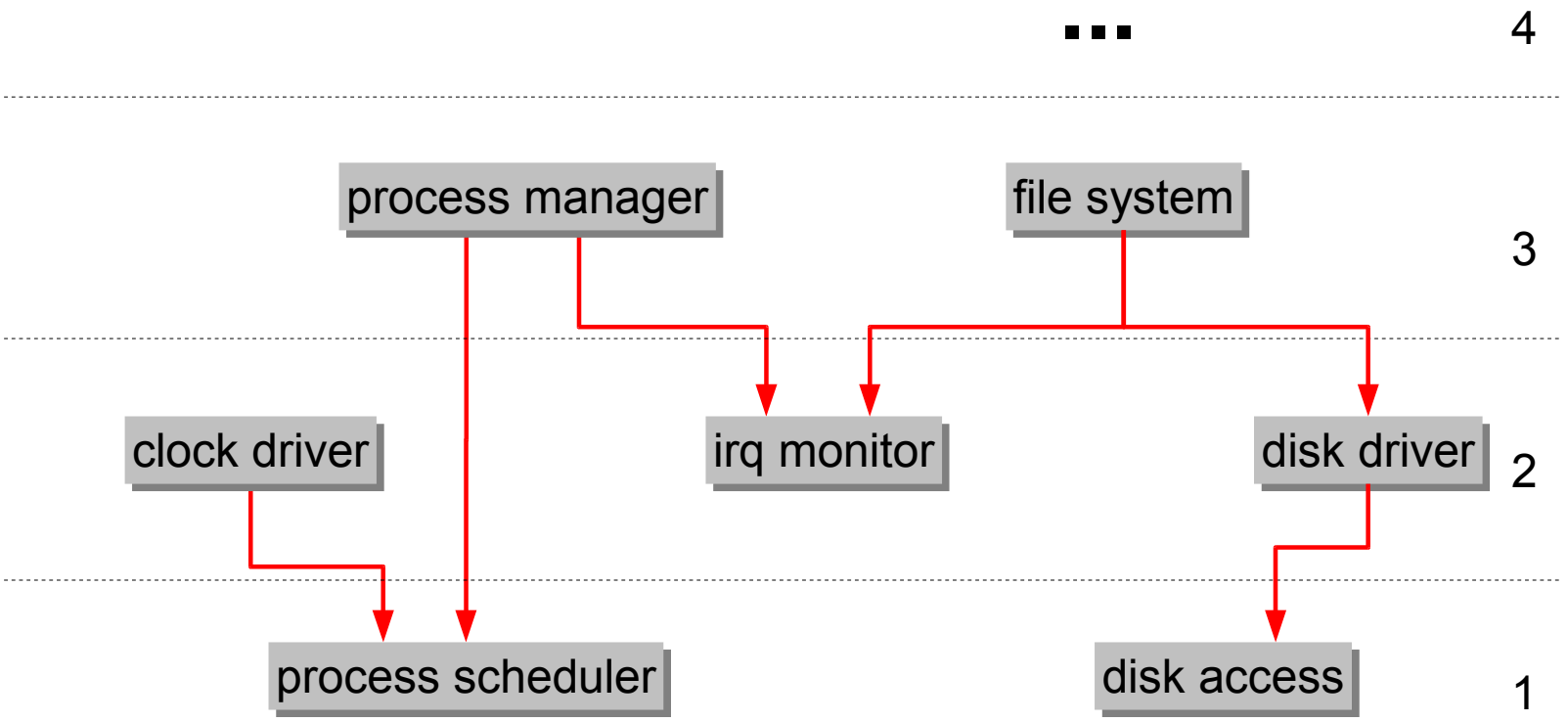
- elementare Systembausteine der Software (Entwurfsebene)
- „logische Funktionen“



Funktionale Hierarchie – Elemente

Funktionale Abhängigkeiten

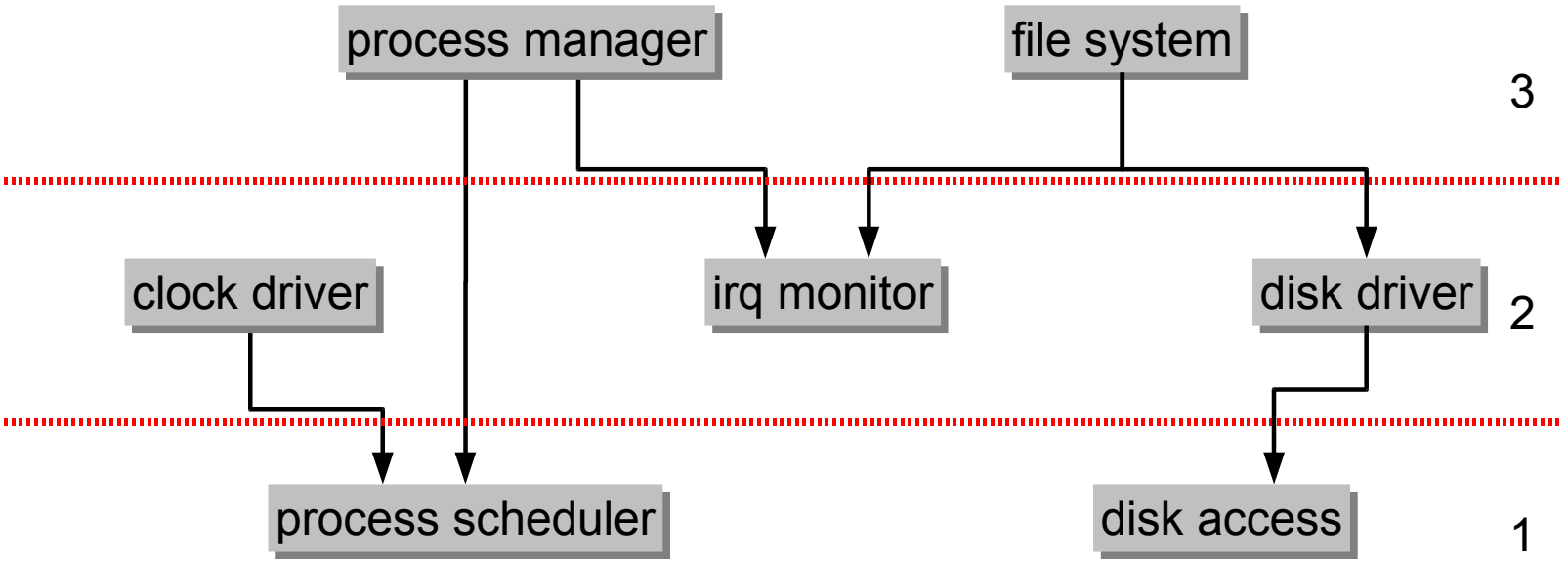
- logische Abhängigkeiten zwischen Systembausteinen („***verwendet***“)
- müssen einen **azyklischen Graphen** ergeben



Funktionale Hierarchie – Elemente

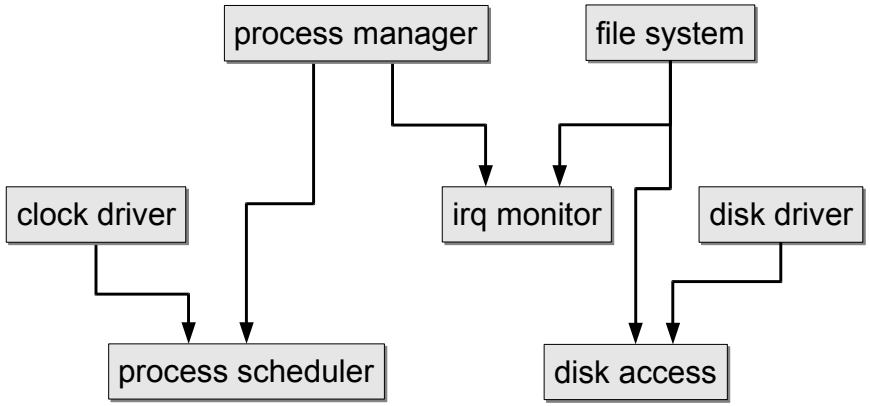
Ebenen

- Gruppierung zusammenhängender Funktionen
- Ebene **0** ist die **Hardware**
- Ebenen **1...*n*** fügen virtuelle Hardware als **minimale Erweiterungen** hinzu
- Functionen der Ebene ***n*** *kennen* alle Funktions der Ebenen **0...*n***



Funktionale Hierarchie – Vorteile

- verständlich
- gut für „bottom-up“ Entwicklung geeignet
- feingranulare Konfigurierbarkeit durch „minimale Erweiterungen“
- erfordert keine speziellen Implementierungsmechanismen



It is the system design which is hierarchical, not its implementation Habermann, 1976 [3]

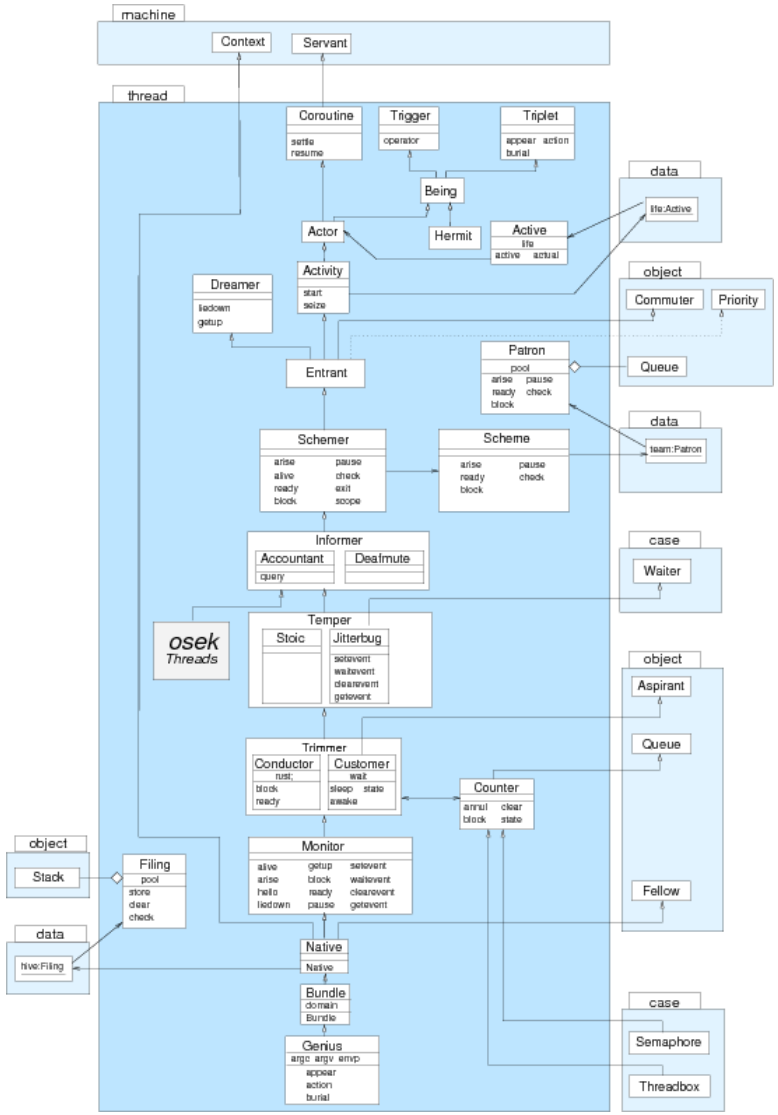
Das PURE Projekt (1995–2003)

Ziel: Eine hochgradig konfigurierbare BS Produktlinie für eingebettete Systeme

- Herausforderungen
  - Umgang mit großer Variabilität im Problemraum
  - Umsetzung der Variabilität im Lösungsraum
- Entwicklungsmethoden
  - Merkmalmodelle - Beschreibung der Variabilität
  - **Funktionale Hierarchien** **- Variabler Entwurf**
  - ...

Problem 1: keine Verschachtelung

- hochgradig konfigurierbare Systeme können nicht durch eine einzige FH beschrieben werden
- Beispiel: PURE Prozess-verwaltung
  - 13 Ebenen
  - konfigurierbar in Bezug auf ...
    - Scheduling-Strategie
    - Prozess-Kontext
    - Systemschnittstelle
    - ...



Problem 1: keine Verschachtelung

- hochgradig konfigurierbare Systeme können nicht durch eine einzige FH beschrieben werden
- Be
  ver
  -
  -
  - ...

Lösungsansatz:

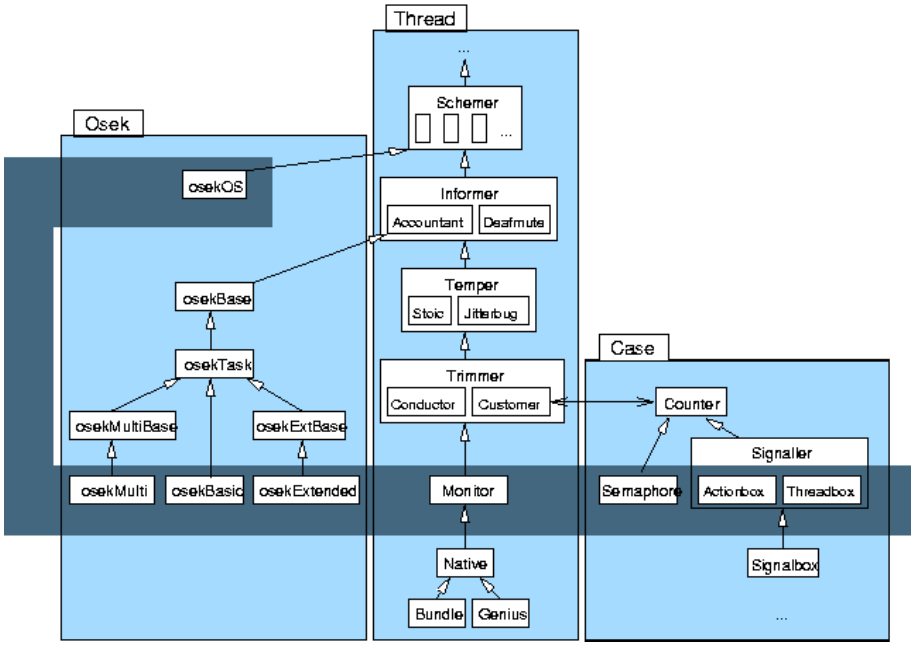
Rekursive Modellierung komplexer Funktionen als „Familie in der Familie“.

→ erlaubt schrittweise Verfeinerung

Problem 2: Querschneidende Belange

- eine Implementierung mit Aspekten kann nur schwer abgeleitet werden

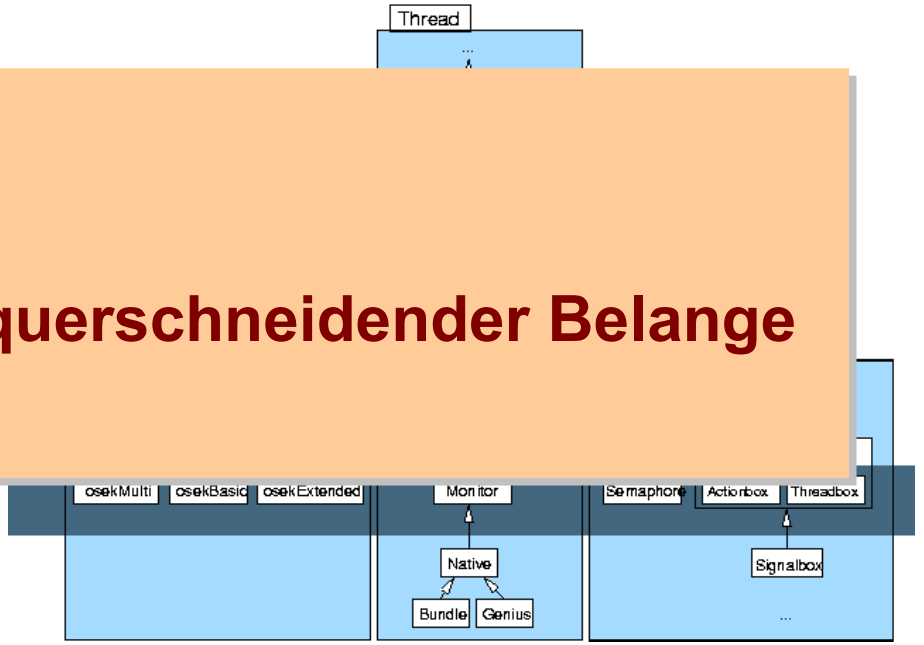
- Beispiel: PURE Unterbrechungssynchronisation
  - 166 *Join-Points*
  - 15 betroffene Klassen



Problem 2: Querschneidende Belange

- eine Implementierung mit Aspekten kann nur schwer abgeleitet werden

- Beispiel: PURE Unterbrechungssynchronisation
  - 166 *Join-Points*
  - 15 betroffene Klassen

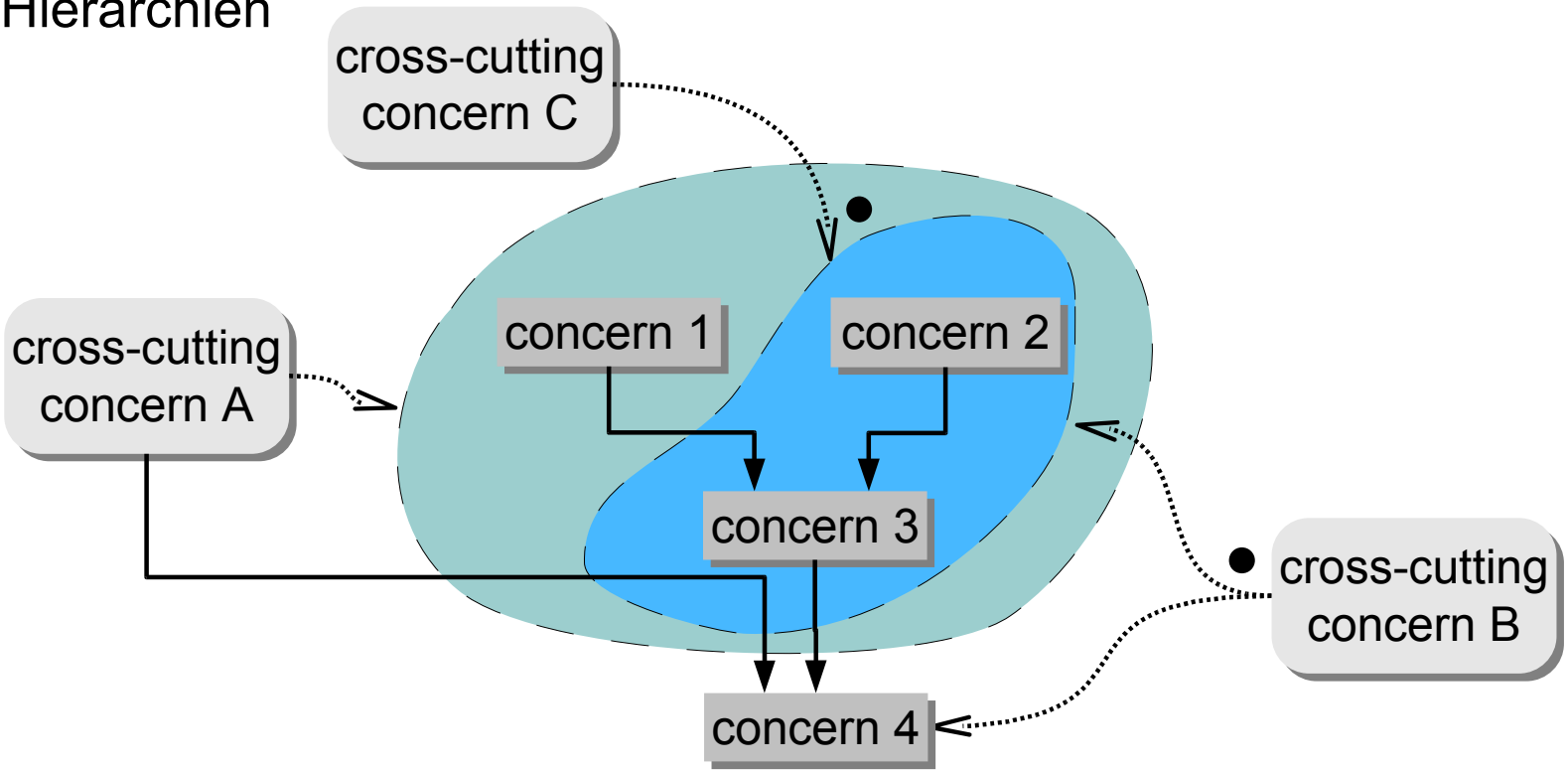


Lösungsansatz:

Explizite Modellierung querschneidender Belange

Das Modell der Belang-Hierarchien

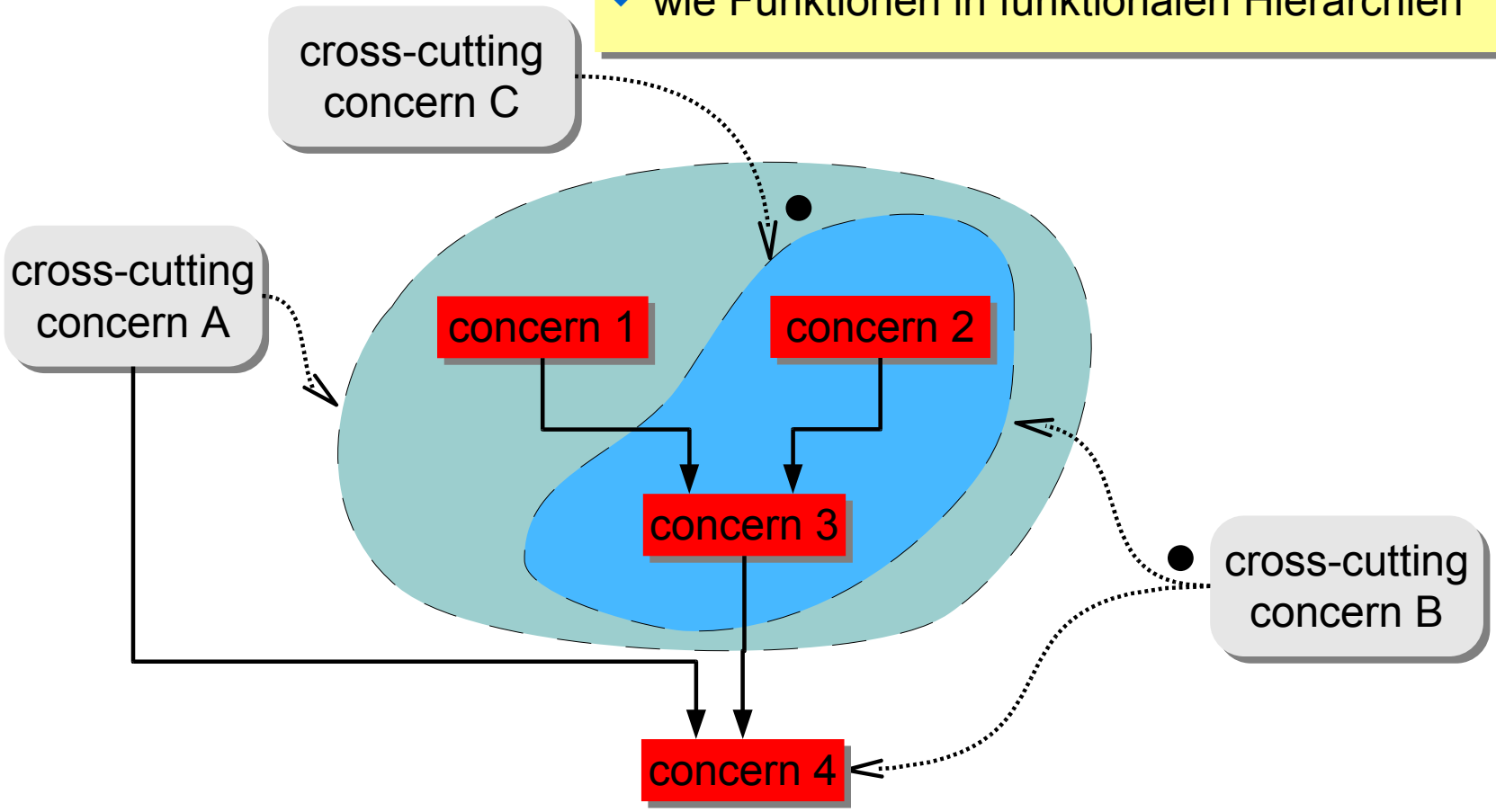
- Erweiterung der funktionalen Hierarchien
  - beschreibt funktionale Abhängigkeiten (zyklenfrei)
  - erfasst quer schneidende Belange in den Funktionen
  - erlaubt Verfeinerungen von Funktionen durch Unterbelang-Hierarchien



Belang-Hierarchien – Elemente

Herkömmliche Belange

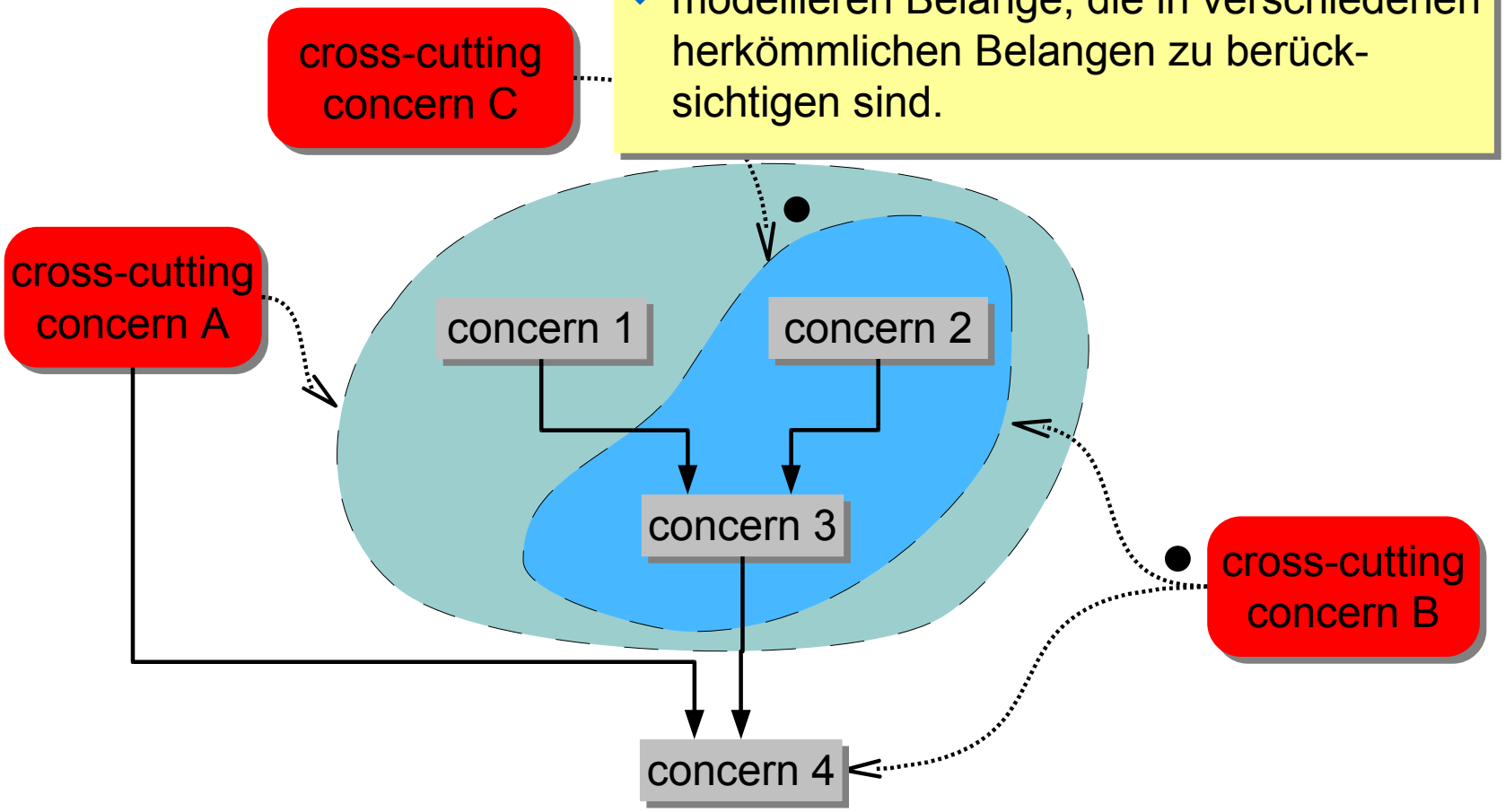
- wie Funktionen in funktionalen Hierarchien



Belang-Hierarchien – Elemente

Querschneidende Belange

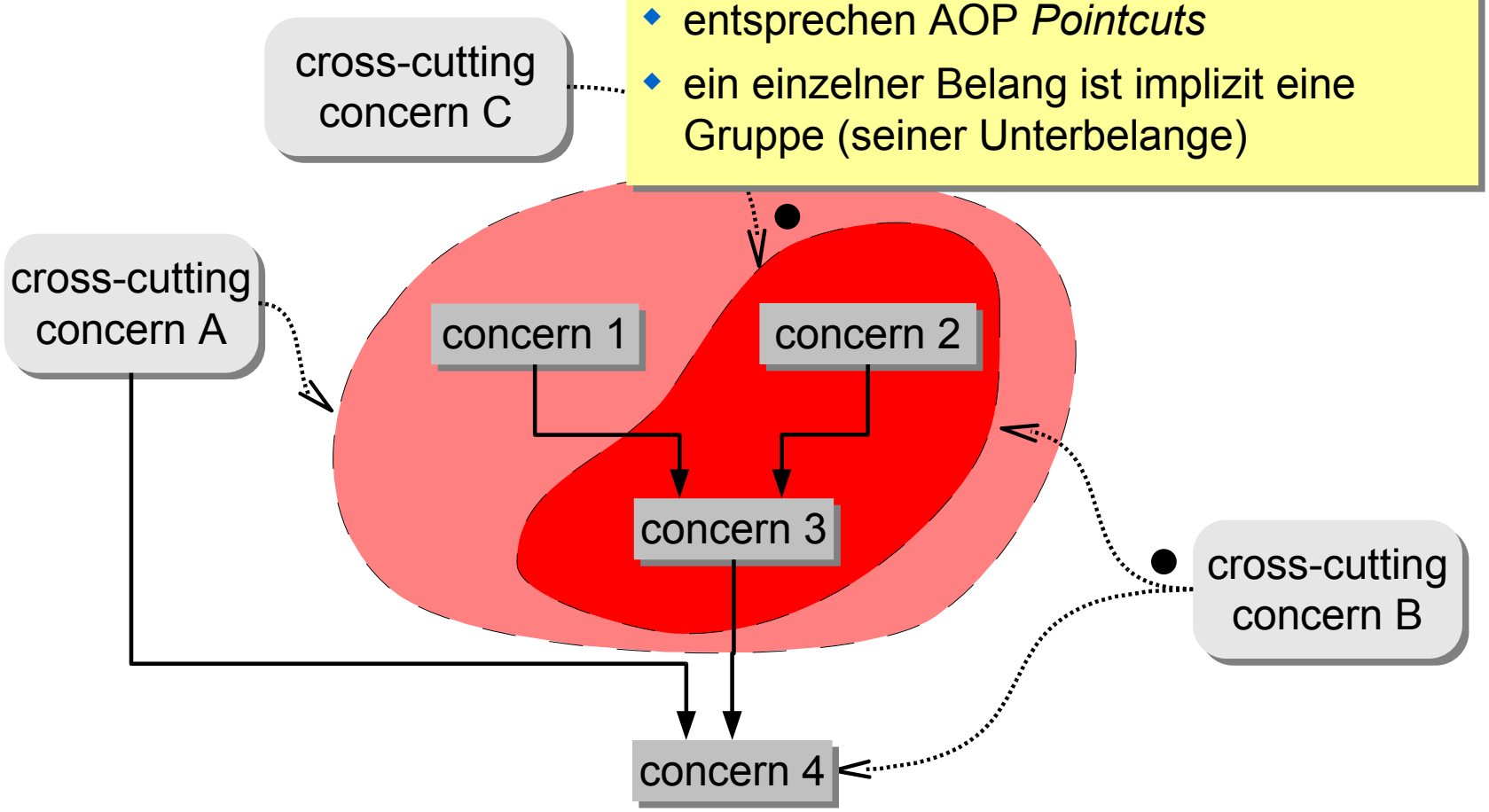
- modellieren Belange, die in verschiedenen herkömmlichen Belangen zu berücksichtigen sind.



Belang-Hierarchien – Elemente

Belanggruppen

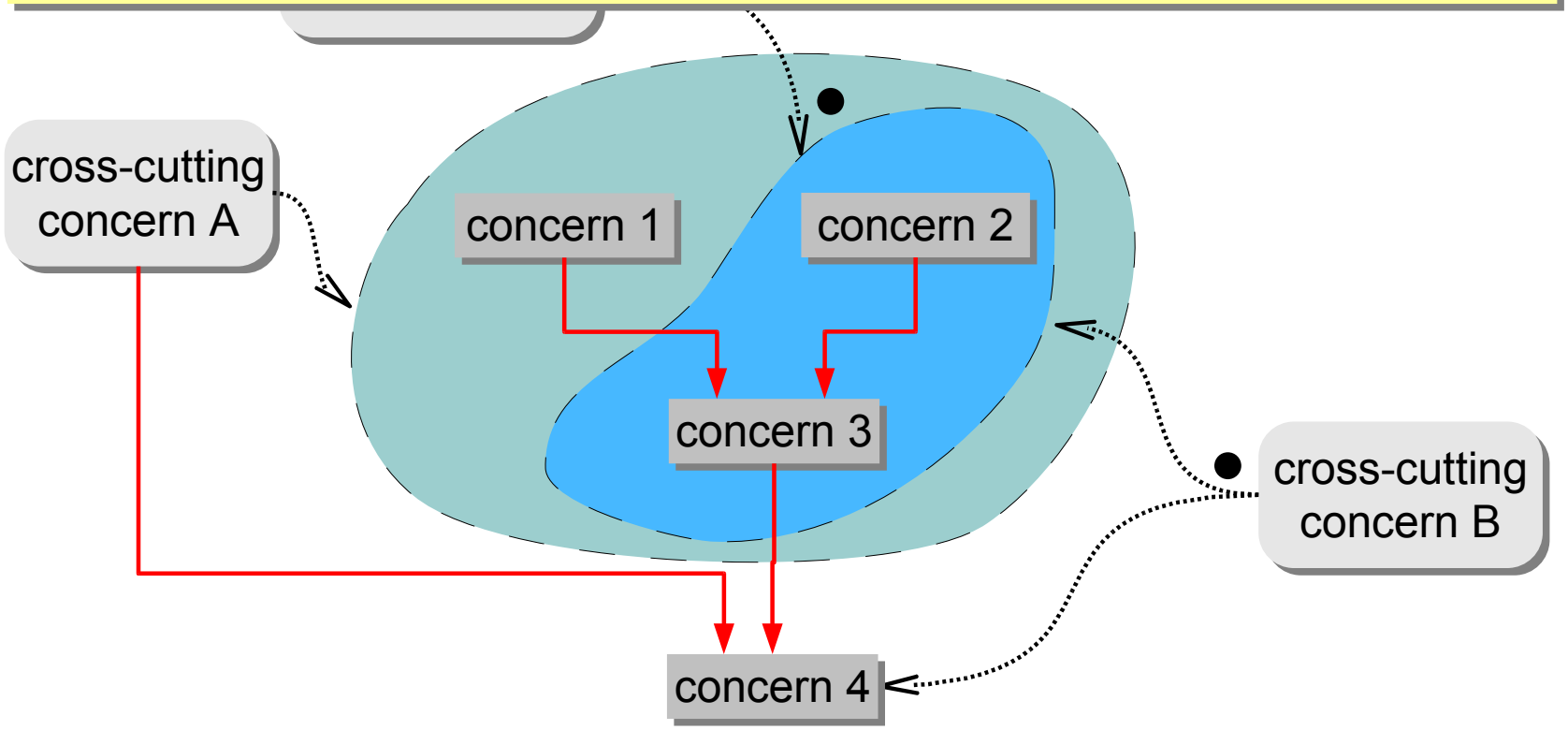
- entsprechen AOP *Pointcuts*
- ein einzelner Belang ist implizit eine Gruppe (seiner Unterbelange)



Belang-Hierarchien – Elemente

Funktionale Abhängigkeiten

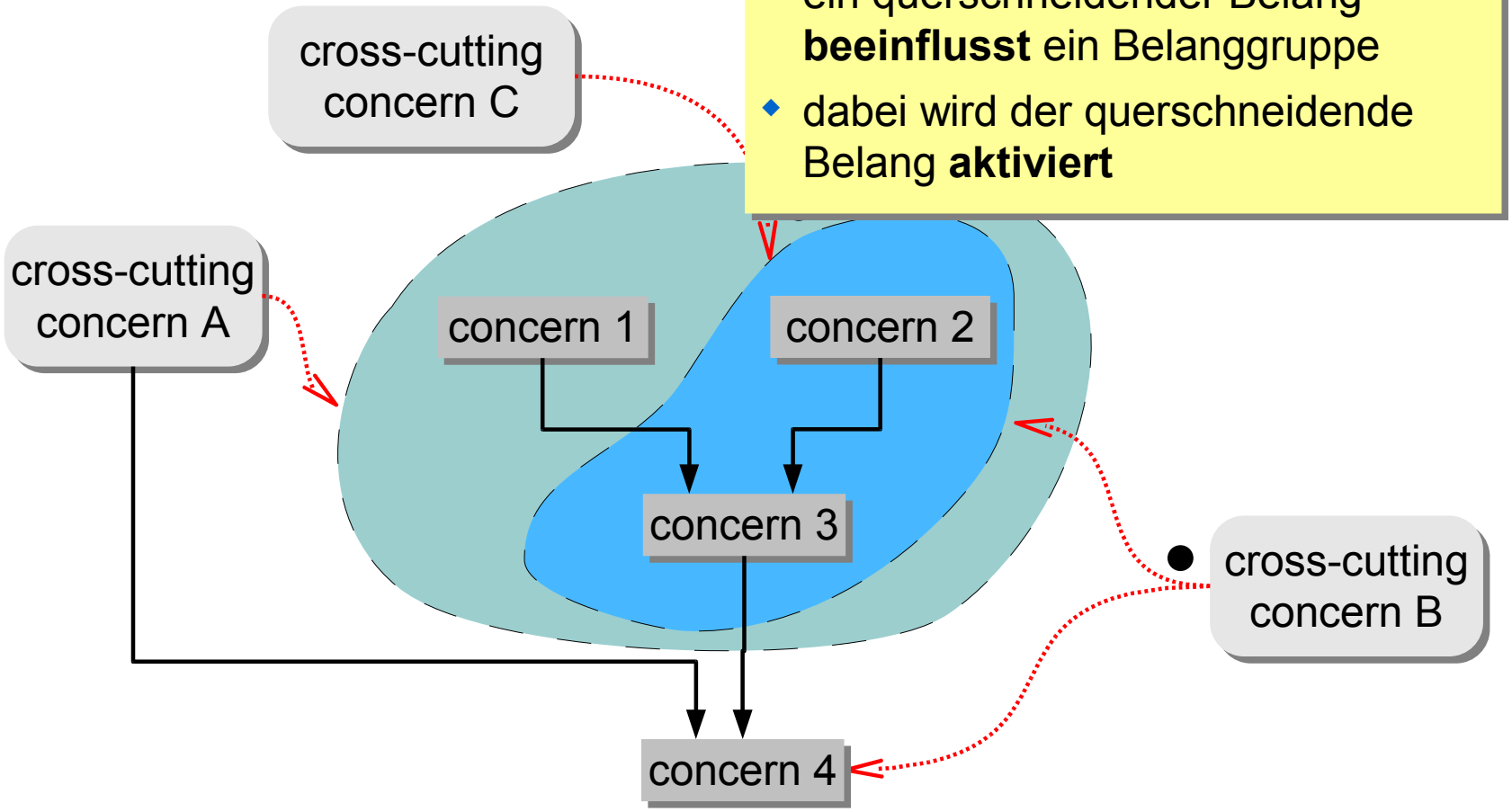
- wie in funktionalen Hierarchien
- querschneidende Belange können auch funktionale Abhängigkeiten haben



Belang-Hierarchien – Elemente

Beeinflussung/Aktivierung

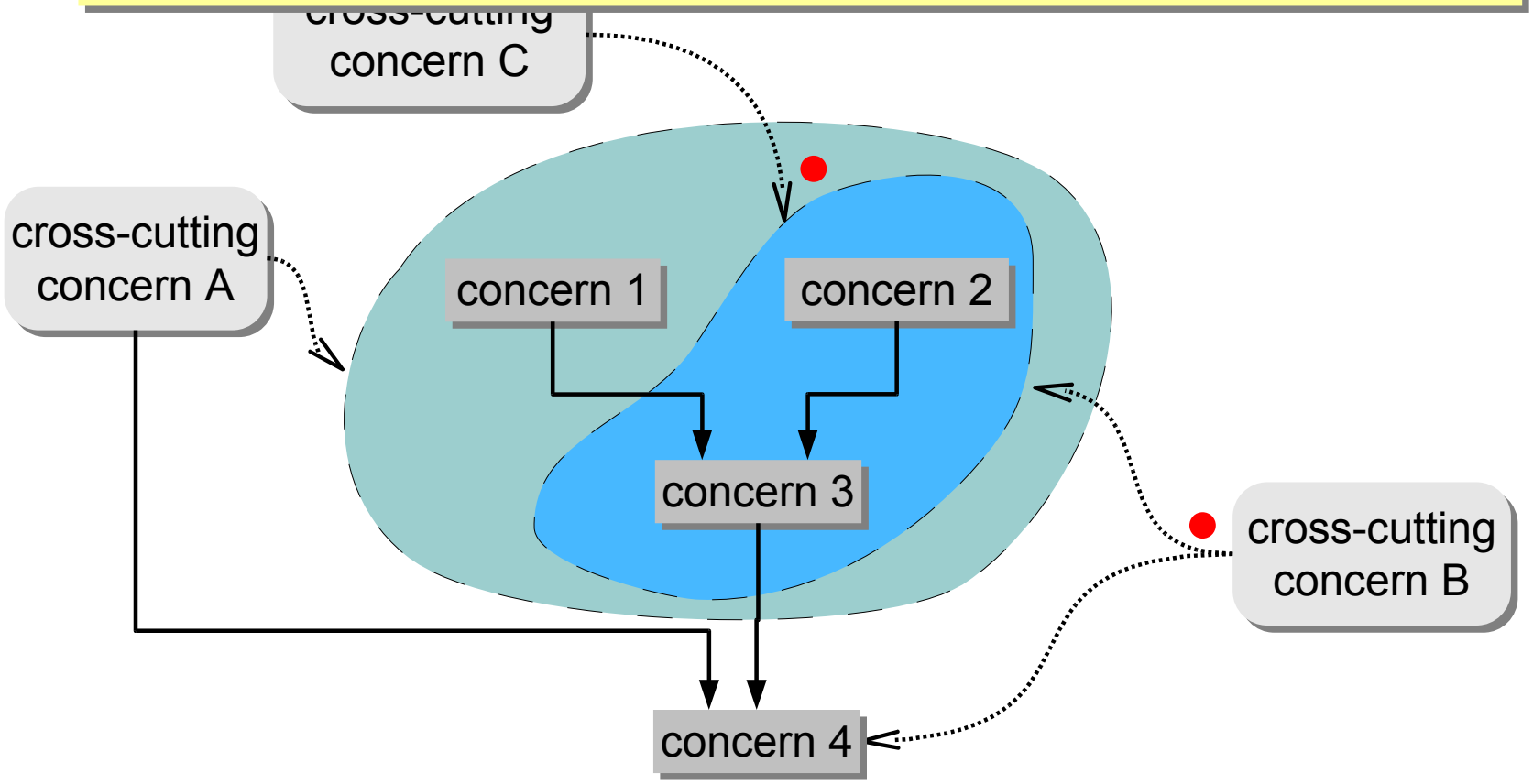
- ein querschneidender Belang **beeinflusst** ein Belanggruppe
- dabei wird der querschneidende Belang **aktiviert**



Belang-Hierarchien – Elemente

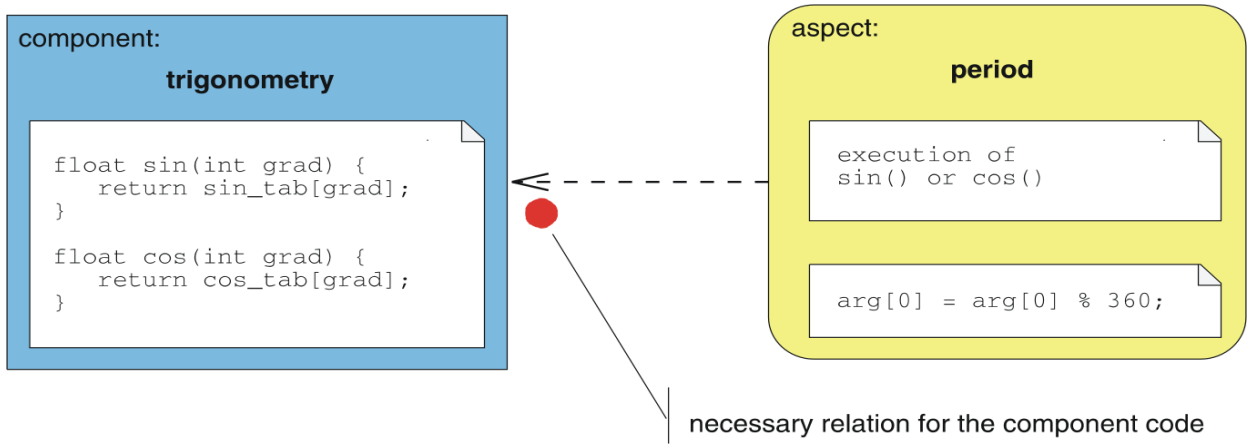
Notwendige Beeinflussung/Aktivierung

wenn die jeweilige Aufgabe ohne die Beziehung nicht erfüllt werden kann

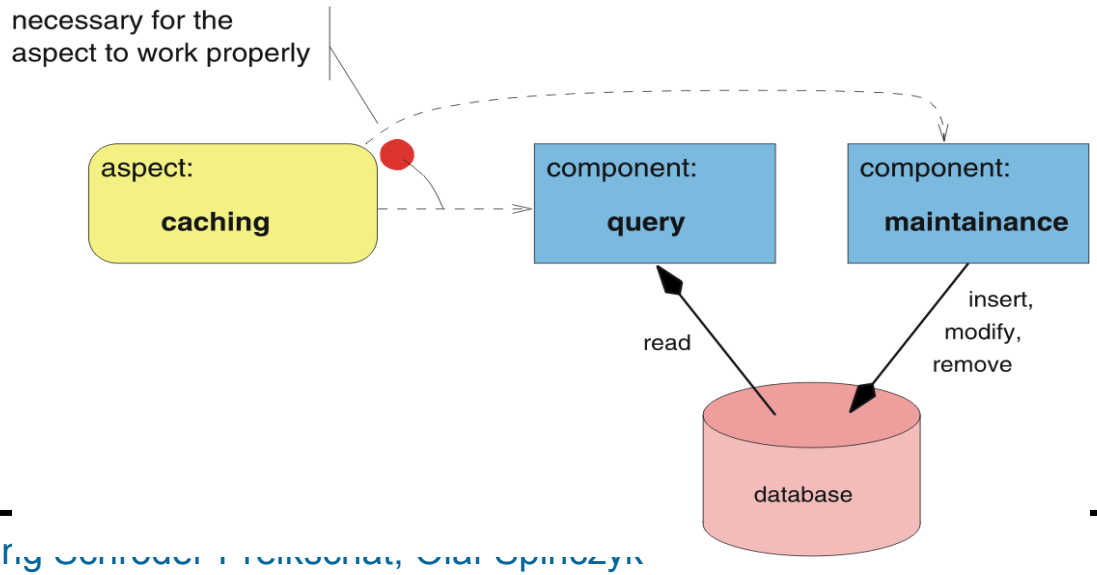


Beeinflussung und Aktivierung

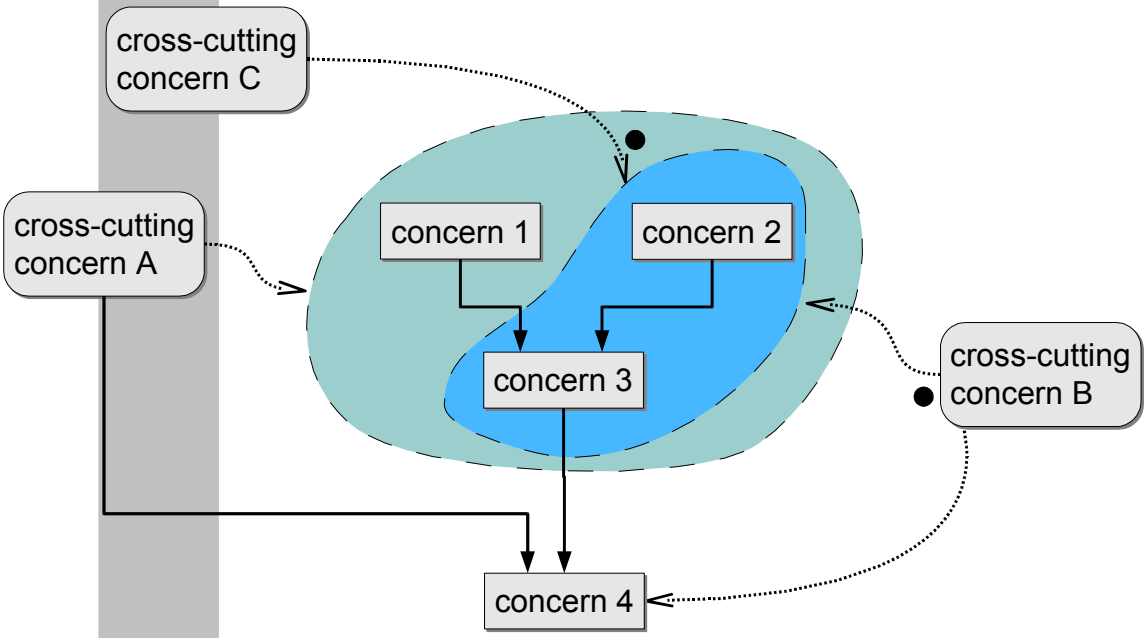
notwendige Beeinflussung



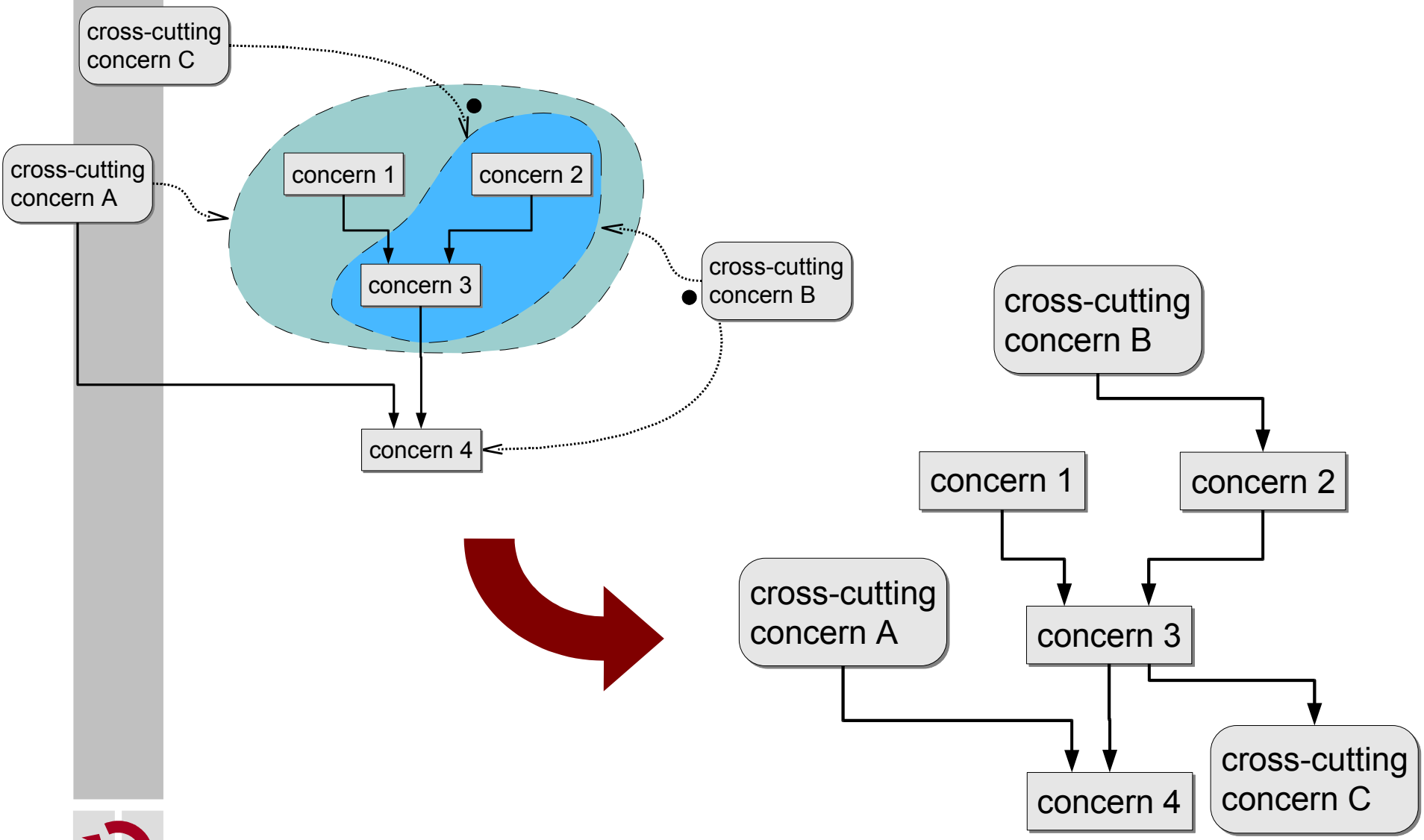
notwendige Aktivierung



Ableitung eines Abhängigkeitsmodells

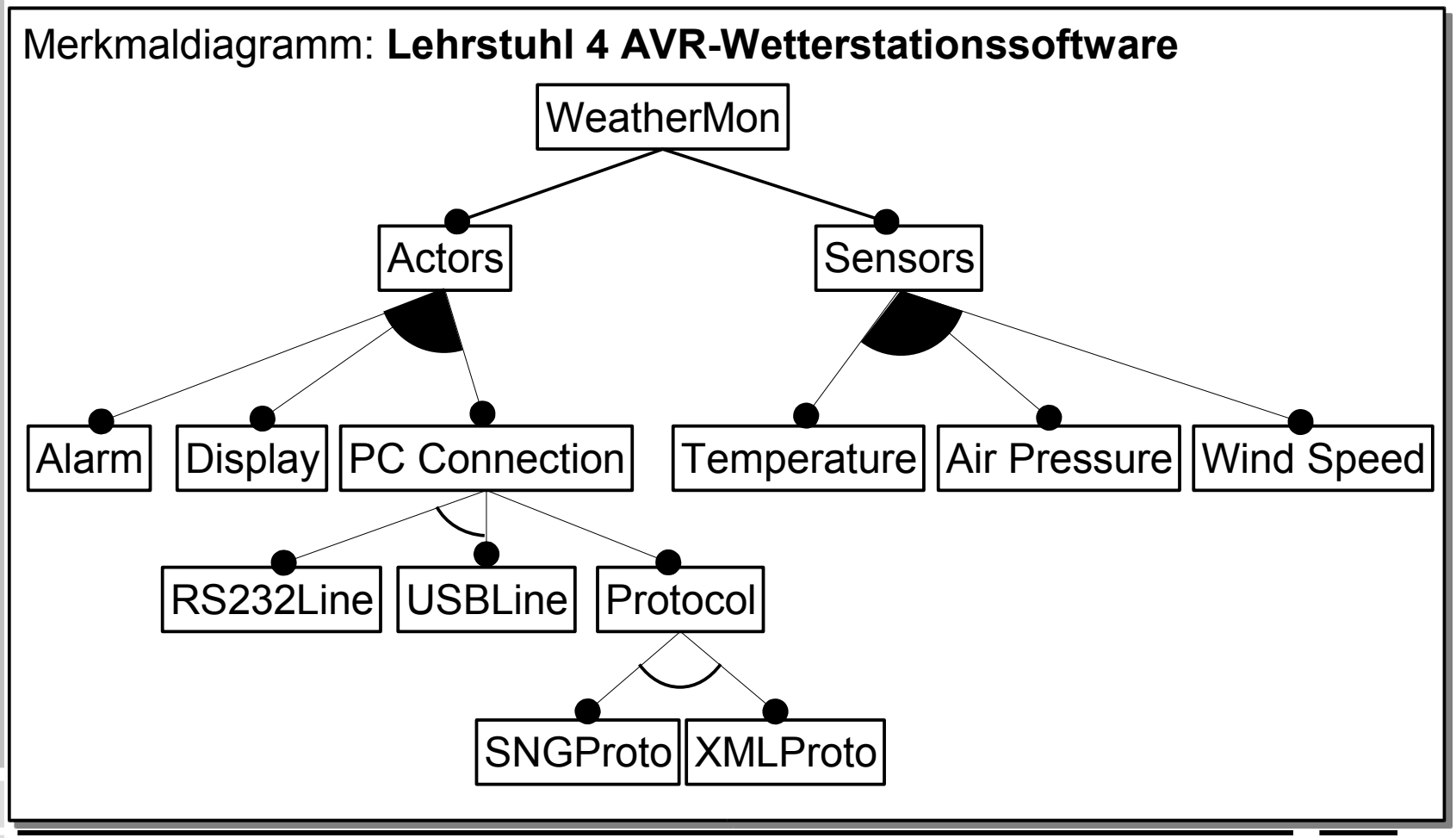


Ableitung eines Abhängigkeitsmodells



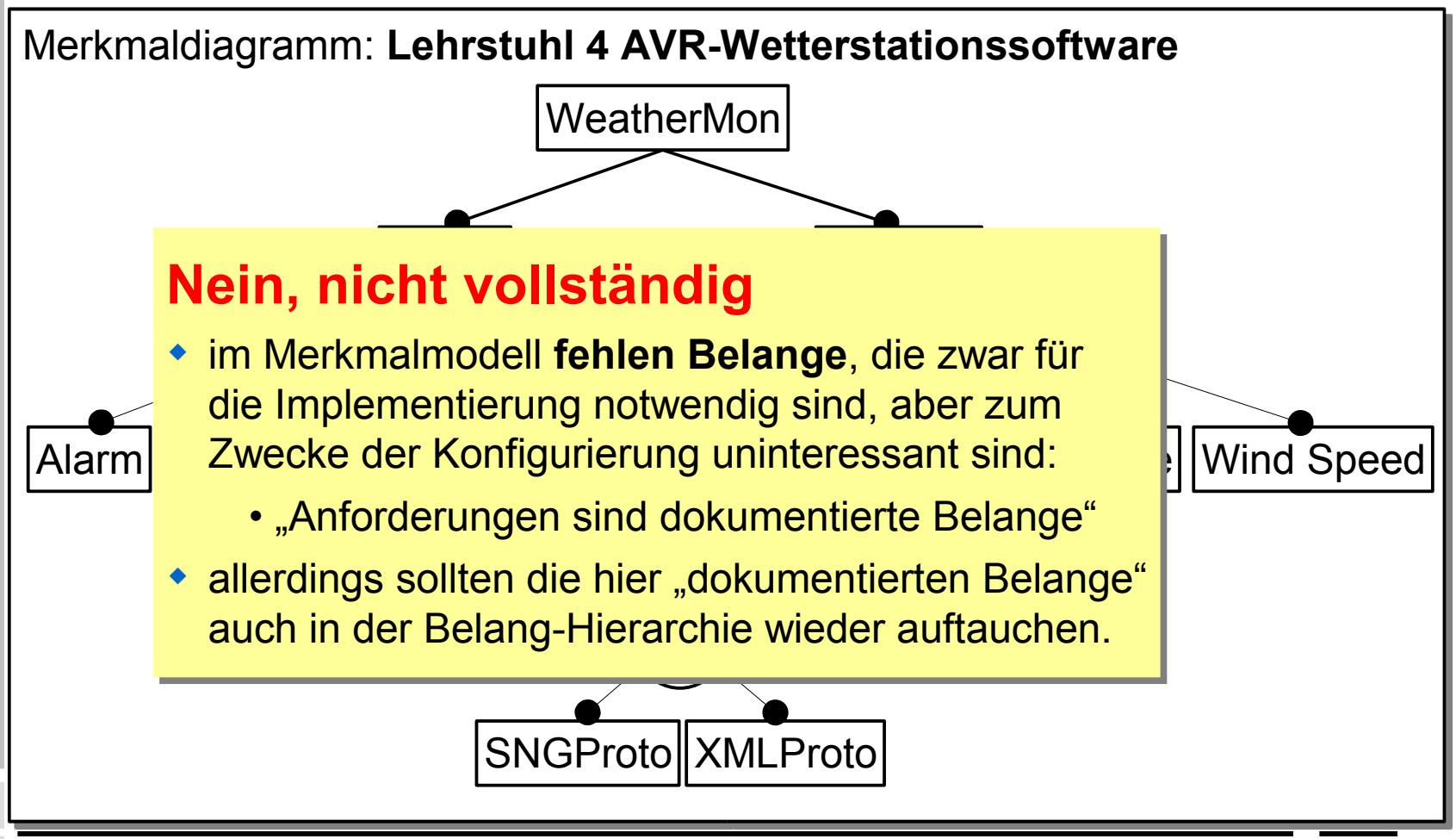
Beispiel: Lehrstuhl 4 Wetterstation

- Lässt die Belang-Hierarchie aus dem Merkmalmodell systematisch ableiten?



Beispiel: Lehrstuhl 4 Wetterstation

- Lässt die Belang-Hierarchie aus dem Merkmalmodell systematisch ableiten?



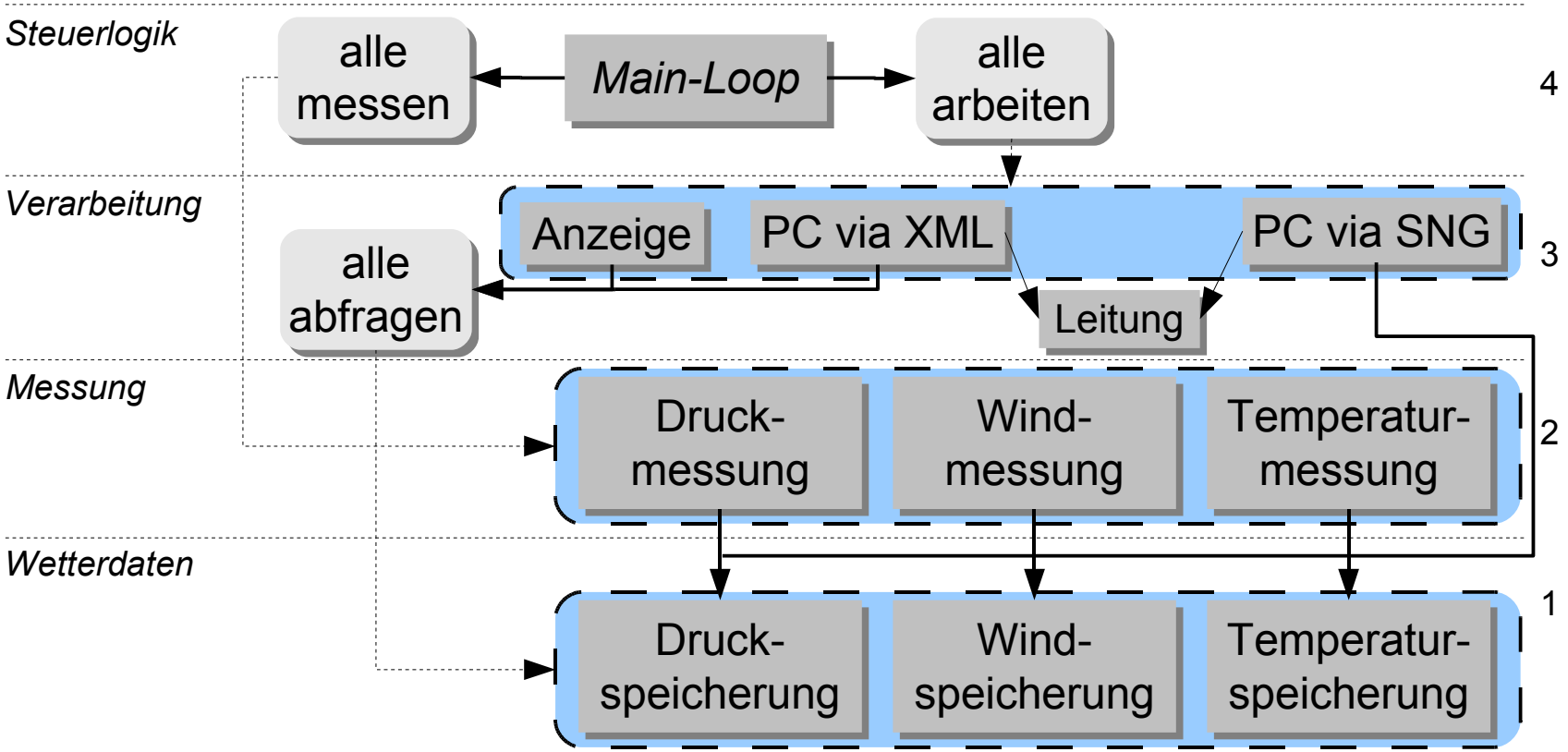
Beispiel: Lehrstuhl 4 Wetterstation

Belang-Hierarchie: **Lehrstuhl 4 AVR-Wetterstationssoftware**

Steuerlogik	4
Verarbeitung (Aktoren)	3
Messung (Sensoren)	2
Wetterdaten (-haltung und -repräsentation)	1

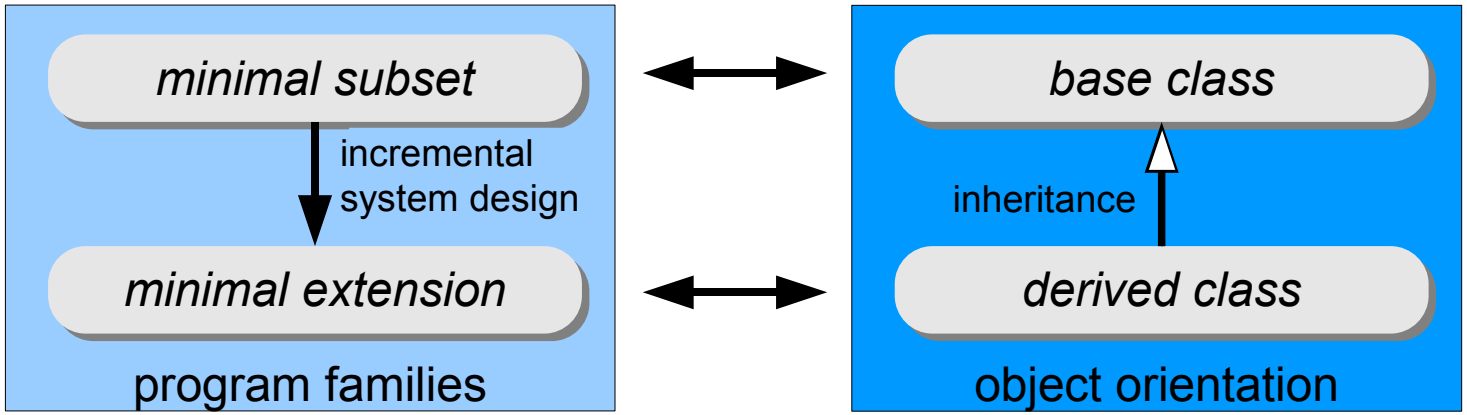
Beispiel: Lehrstuhl 4 Wetterstation

Belang-Hierarchie: **Lehrstuhl 4 AVR-Wetterstationssoftware**



OO vs. familienbasierter Entwurf

- Vererbung kann als Vehikel für die schrittweise Erweiterung genutzt werden

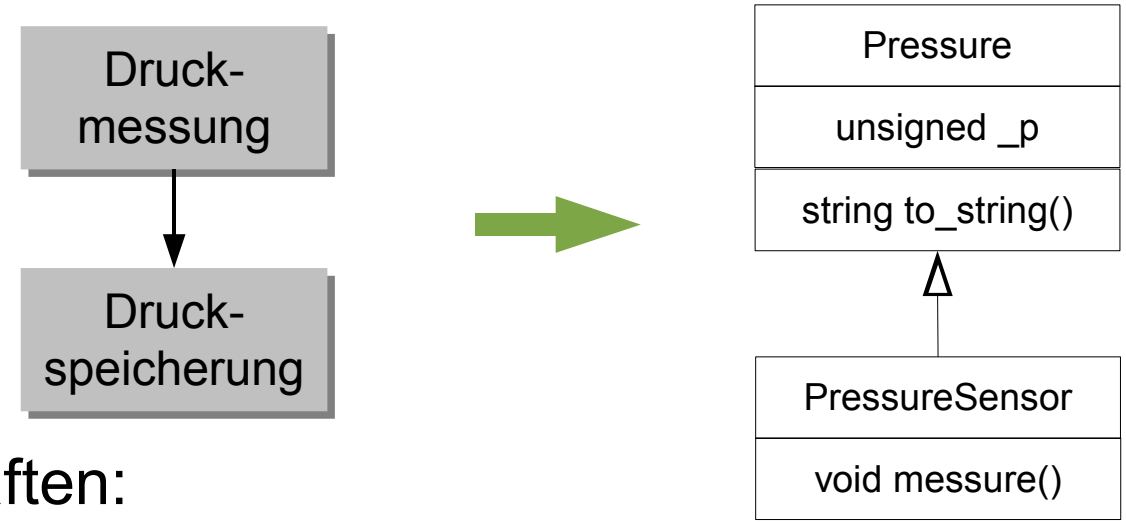


[2]

- **Pro**
  - funktionale Hierarchien lassen sich (relativ) leicht in eine Modulstruktur transformieren
- **Contra**
  - Entwurf wird leicht als schlechter OO-Entwurf missverstanden

Funktionen werden Klassen (1)

- die funktionale Hierarchie wird „einfach auf den Kopf gestellt“:

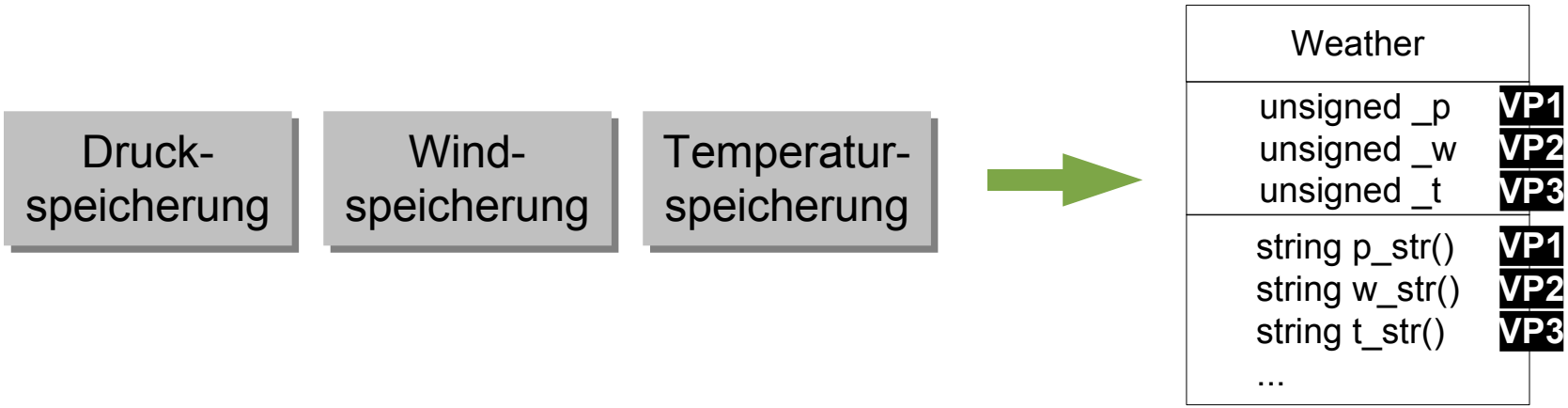


Eigenschaften:

- Konfigurierung durch Anwender + Binder
- eventuell Mehrfachvererbung
- Trennung des Zustands bei Funktionen einer Ebene
- eignet sich für grobe Funktionen
  - Klassen ohne Attribute machen z.B. selten Sinn

Funktionen werden Klassen (2)

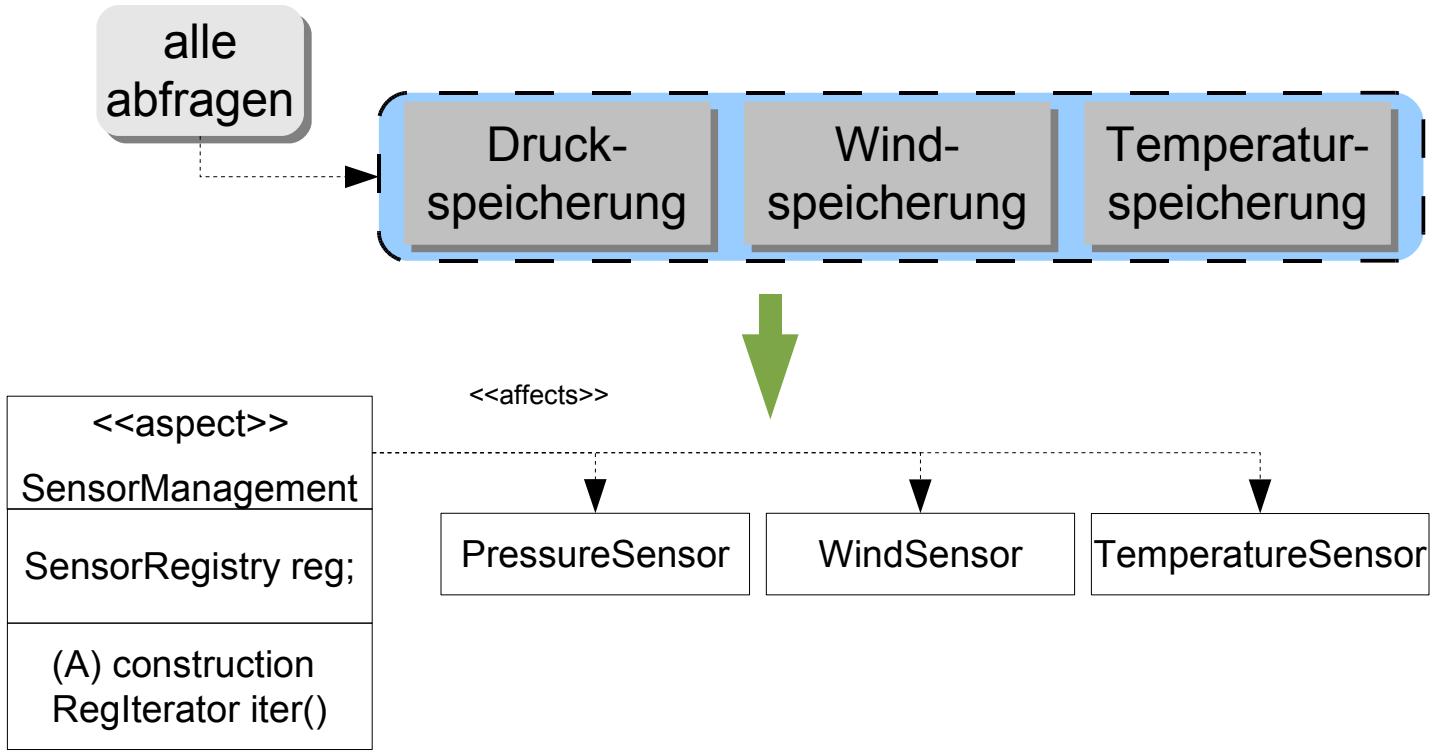
- Funktionen einer Ebene können auch zu einer Klasse zusammengefasst werden:



Eigenschaften:

- Konfigurierung **eventuell** durch Anwender + Binder
- gemeinsamer Zustand möglich
- für kleine Funktionen die angenehmere Schnittstelle

Querschn. Belange werden Aspekte

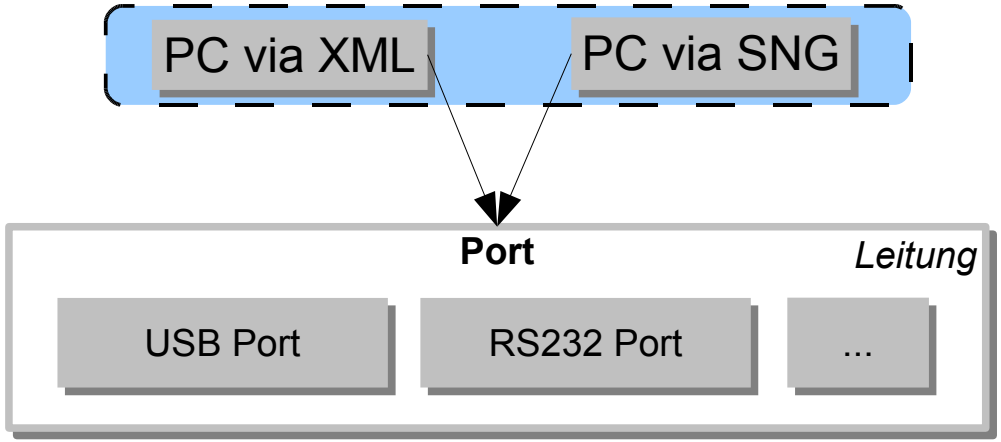


Eigenschaften:

- Entkopplung der Funktionen erleichtert Konfigururierung
- nicht immer möglich/sinnvoll

Konfigurierbare Funktionen

- ... machen eine gemeinsame Schnittstelle notwendig:



- wahlweise statisch gebunden, z.B. „Klassenalias“
- oder dynamisch gebunden, z.B. mit abstrakter Klasse

Zusammenfassung

- Grundregeln bei der Ableitung einer AO-Klassenhierarchie aus einer Belang-Hierarchie:
  - Erweiterungen (Ebenen) werden durch Vererbung ausgedrückt
  - Querschneidende Belange können i.d.R. als Aspekte realisiert werden
  - Konfigurierbare Funktionen erfordern eine Schnittstellendefinition
    - Bindung kann statisch oder dynamisch erfolgen
    - das Hinzufügen abstrakter Schnittstellenklassen erfolgt immer anwendungsgetrieben, d.h. die Entwicklung beginnt **nicht** mit einer abstrakten Klasse wie es in OO-Entwürfen oft zu finden ist
- der Prozess vermeidet zwecks Konfigurierbarkeit zyklische Abhängigkeiten in der Modulstruktur

Ausblick

- Untersuchung verschiedener Techniken zur Umsetzung von Variabilität in der Implementierung der Komponenten
  - Stand der Kunst: Beispiel ECOS
  - Werkzeugbasierte Lösungen
  - Programmiersprachenbasierte Lösungen

Literatur

[1] O. Spinczyk. *Aspektorientierung und Programmfamilien im Betriebssystembau*. Dissertation, Otto-von-Guericke-Universität Magdeburg, 2002.

[2] W. Schröder-Preikschat. *The Logical Design of Parallel Operating Systems.* Prentice Hall, 1994. ISBN 0-13-183369-3.

[3] A. N. Habermann, L. Flon, and L. Cooprider. *Modularization and Hierarchy in a Family of Operating Systems*. Communications of the ACM, 19(5):266-272, 1976.

[4] D. L. Parnas. *Some Hypotheses about the 'uses' Hierarchy for Operating Systems*, Technical Report, Technische Hochschule Darmstadt, Darmstadt, West Germany, 1976.