Methodische und Praktische Grundlagen der Informatik (MPGI 3)

WS 2008/09

Softwaretechnik

Steffen Helke

Andreas Mertgen (Organisation)

Rojahn Ahmadi, Georgy Dobrev, Daniel Gómez Esperón, Simon Rauterberg, Jennifer Ulrich (Tutoren)

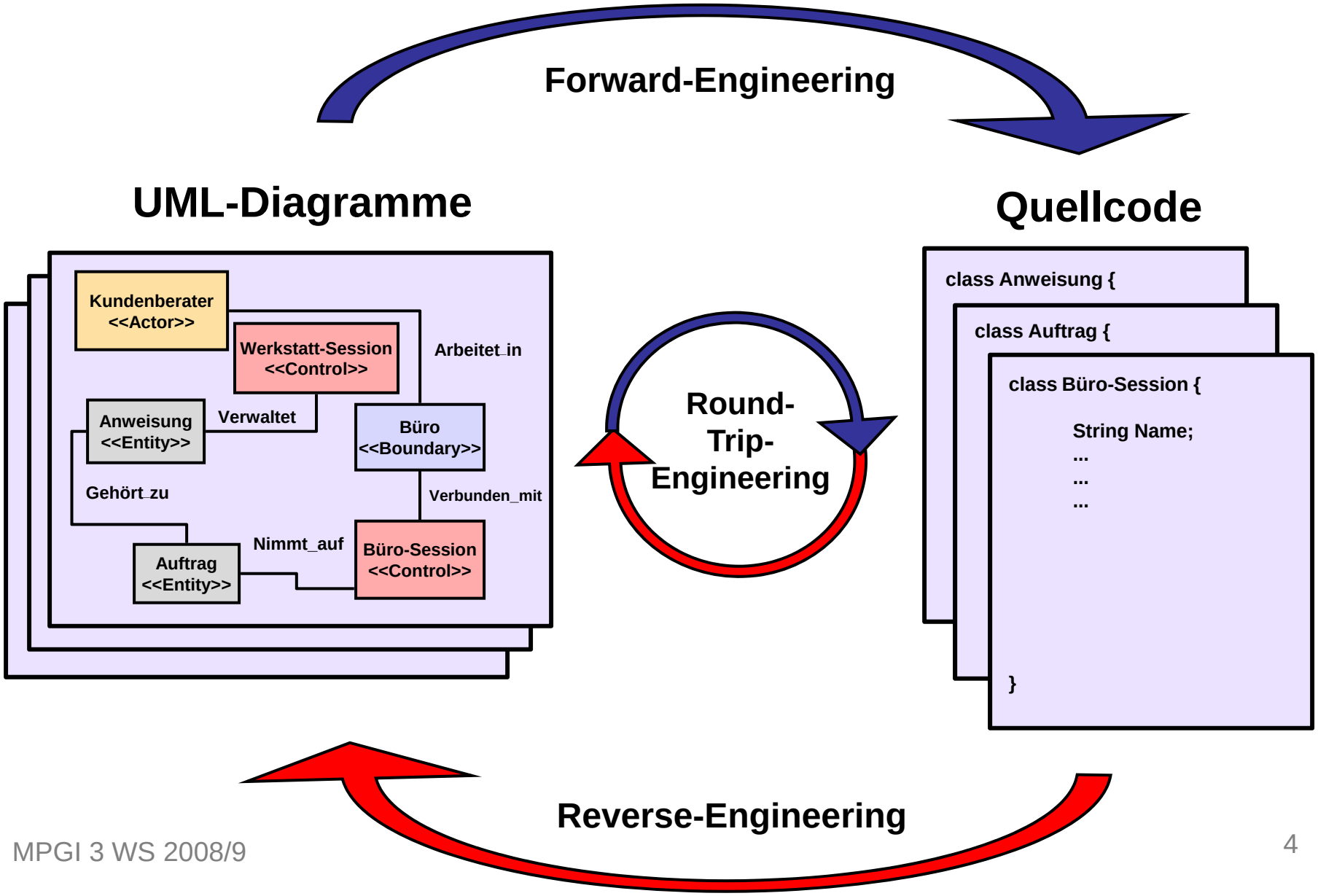
Was machen wir heute?

- **Wiederholung**
  - Implementierungskonzepte
- **Entwurf**
  - Architekturstile
- **Ausblick**
  - Z-Syntax

Wie kommt man zu einer guten Implementierung?

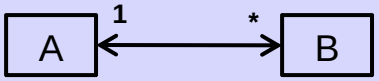
- **Auswahl einer passenden Architektur**
- **Einsetzen von Entwurfsmustern**
- **Verwenden von Case-Tools**
- **Umsetzen von etablierten Implementierungsstrategien**

Entwicklung durch den Einsatz von Case-Tools



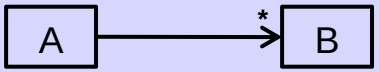
Implementierung von Assoziationen

1. Gegenseitige Referenzierung



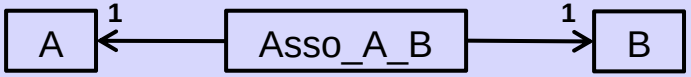
+ schnelle Navigation in beiden Richtungen, Standardimpl.
- redundante Information, Overhead bei Updates

2. Einfache Referenzierung



+ schnelle Navigation in einer Richtungen,
  keine Update-Overheads, keine redundante Information
- aufwendige Navigation in Gegenrichtung

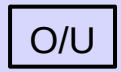
3. Assoziationsklassen



+ automatisch konsistent, kein unnötiger Speicherbedarf
- aufwendige Navigation in beiden Richtungen

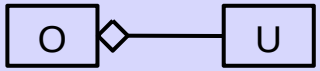
Implementierung von Vererbung

1. Ausnutzung von Flags



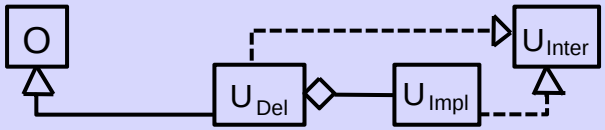
+ ohne Vererbung, intuitives Konzept
- aufwenige Impl., schlecht wartbar, Typfehler zur Laufzeit

2. Delegation (aufwärts und abwärts)



+ Kernimplementierung verbleibt in den Klassen
- Oberklasse kodiert Methoden der Unterklasse über Delegation bzw. umgekehrt, Typfehler zur Laufzeit möglich

3. Interfaces und Delegation



+ Mehrfachvererbung auch in Java möglich, kein redundanter Code, Oberklasse wird nicht modifiziert
- Unterklasse kodiert Methoden des Interfaces mit Delegation

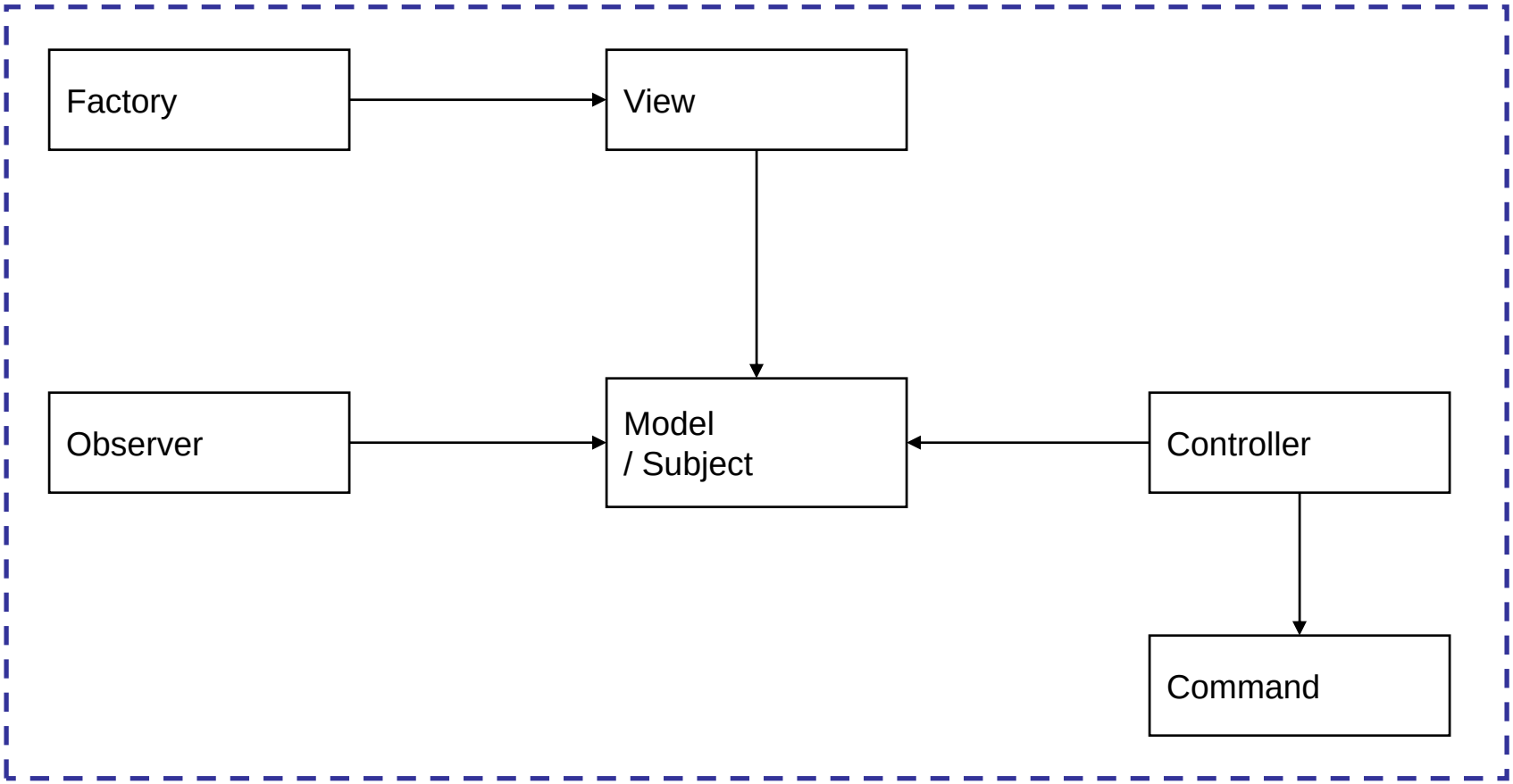
Software-Architekturen

Was genau bedeutet *Architektur* für die *Objektorientierung*?

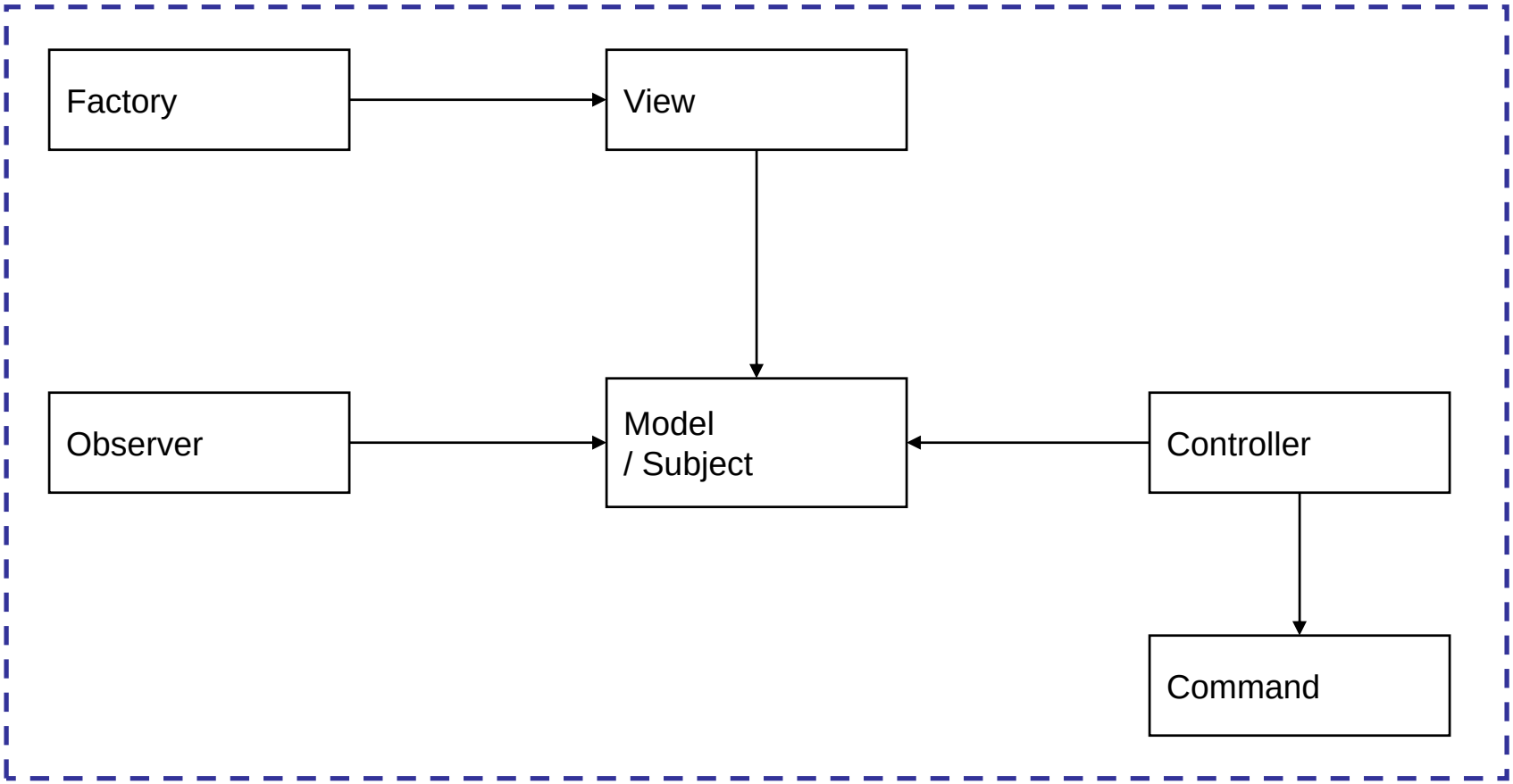
Hinweis:
Folgende Folien sind urheberrechtlich geschützt:
Dahl-Nygaard Lecture © 2007 Jonathan Aldrich

Objekte [Dahl and Nygaard ’67]

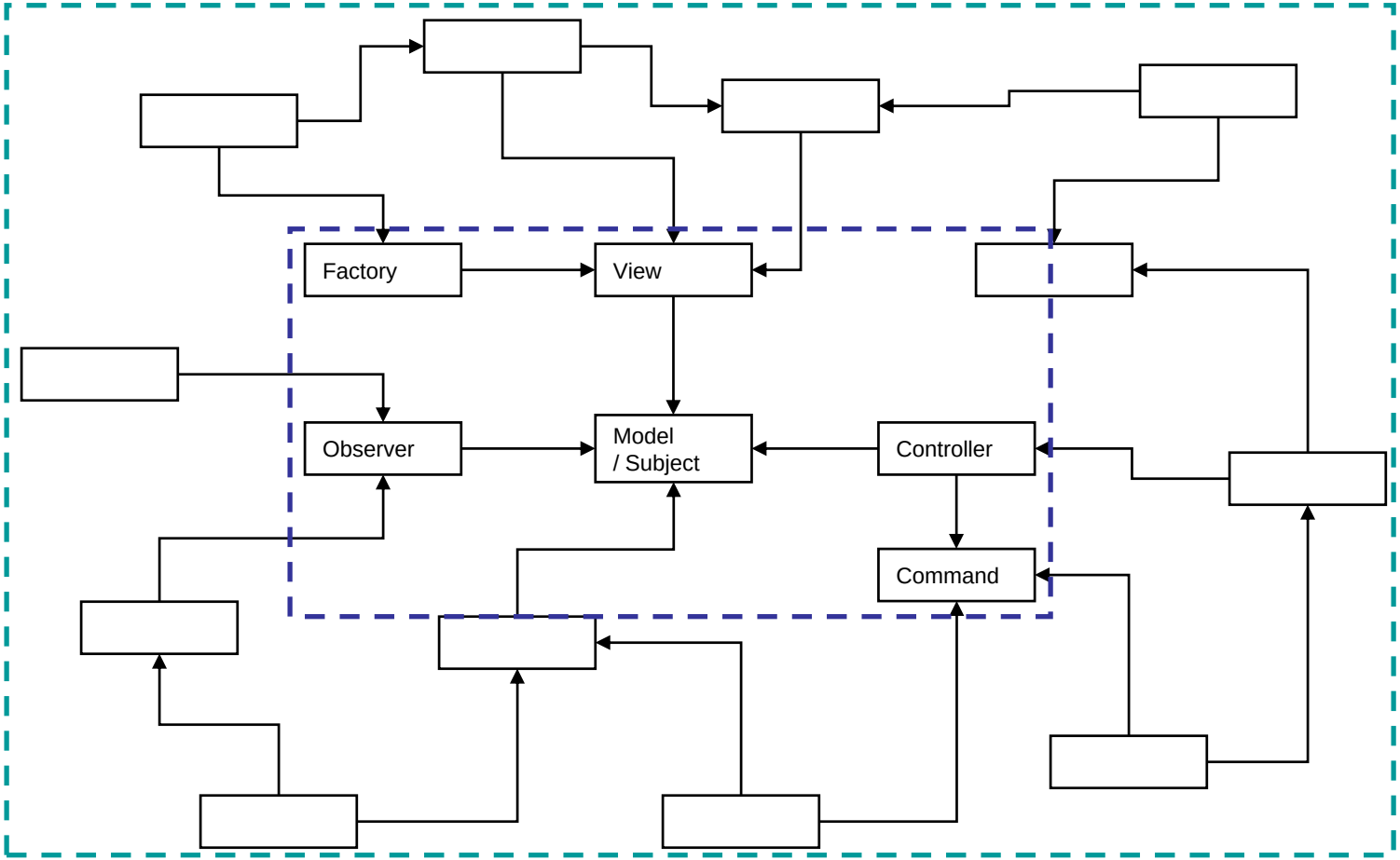
Entwurfsmuster [Gamma, Helm, Johnson, and Vlissides '95]



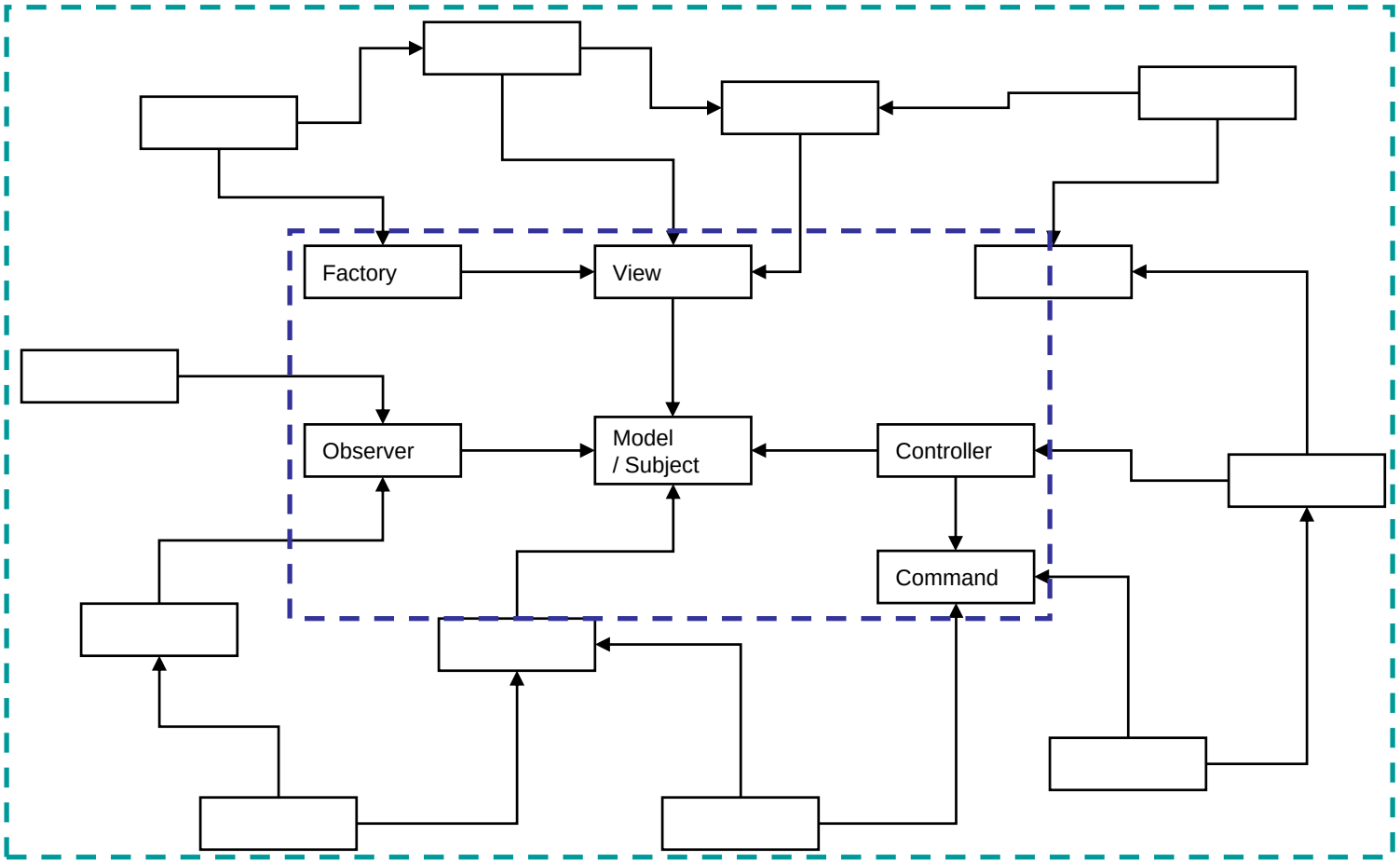
Entwurfsmuster [Gamma, Helm, Johnson, and Vlissides '95]



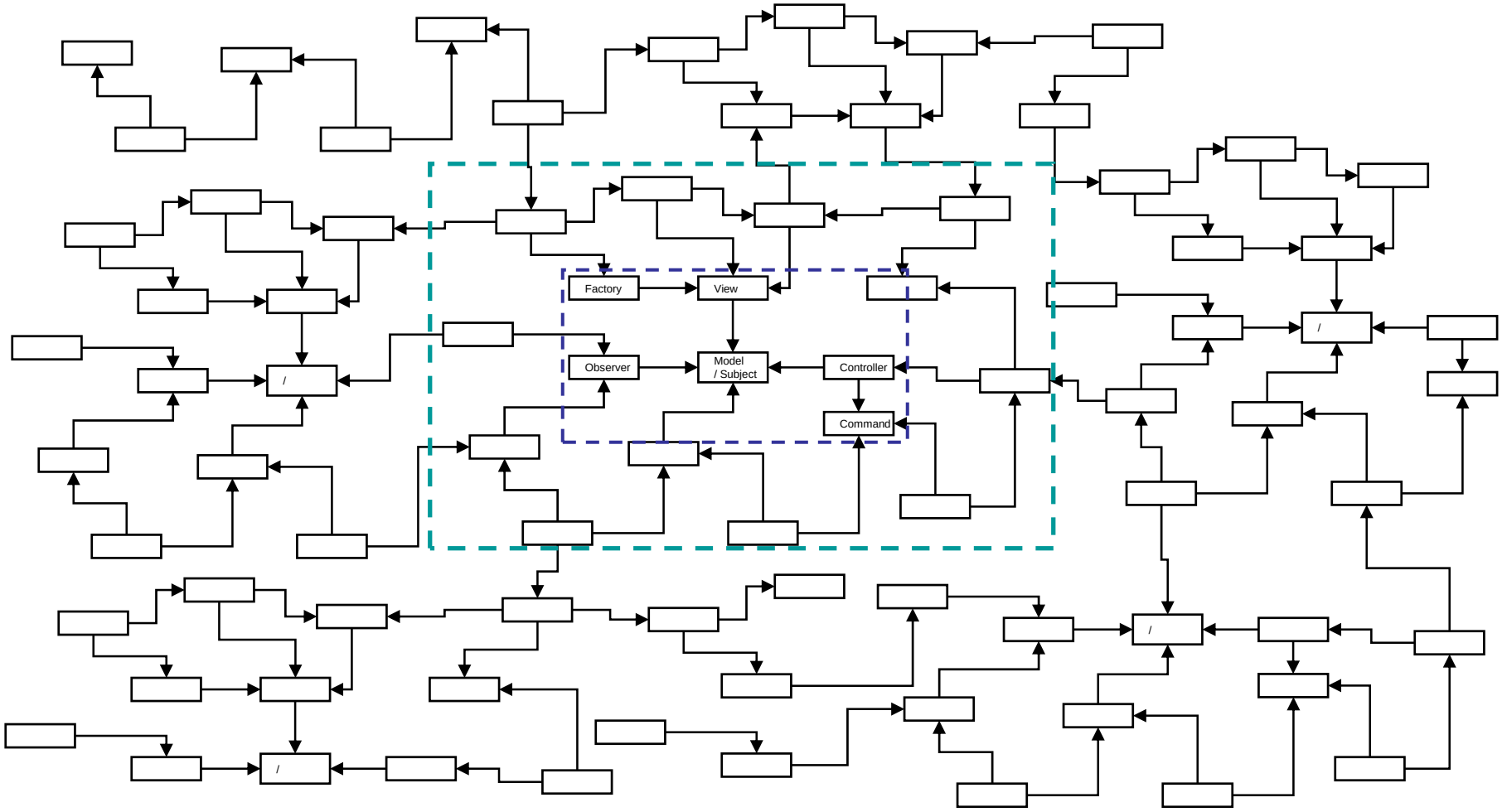
Architekturen [Perry and Wolf 1992] [Garlan and Shaw 1993]



Objektorientierte Architekturen - über Entwurfsmuster hinausgehende Konzepte

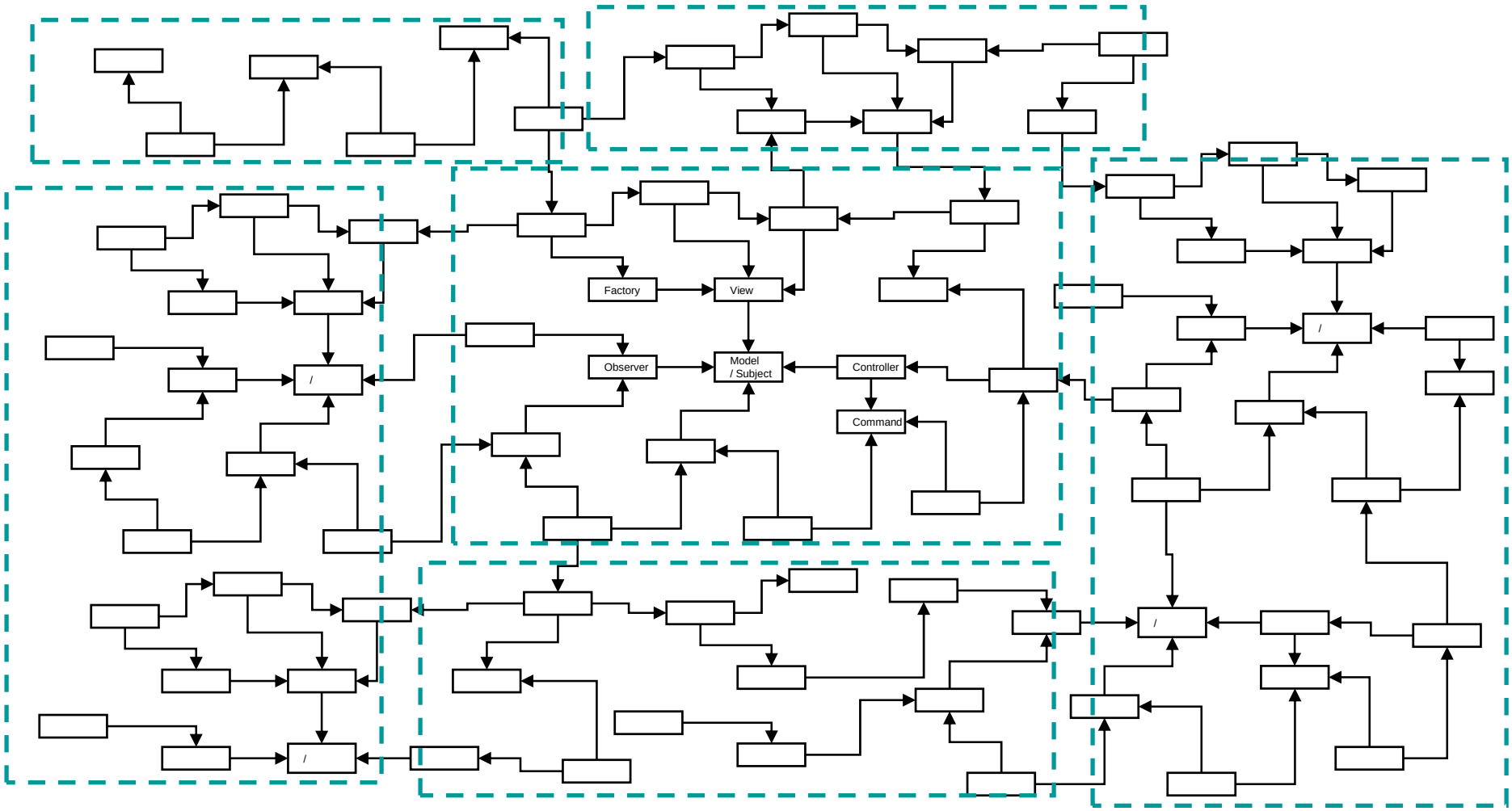


Objektorientierte Architekturen - über Entwurfsmuster hinausgehende Konzepte

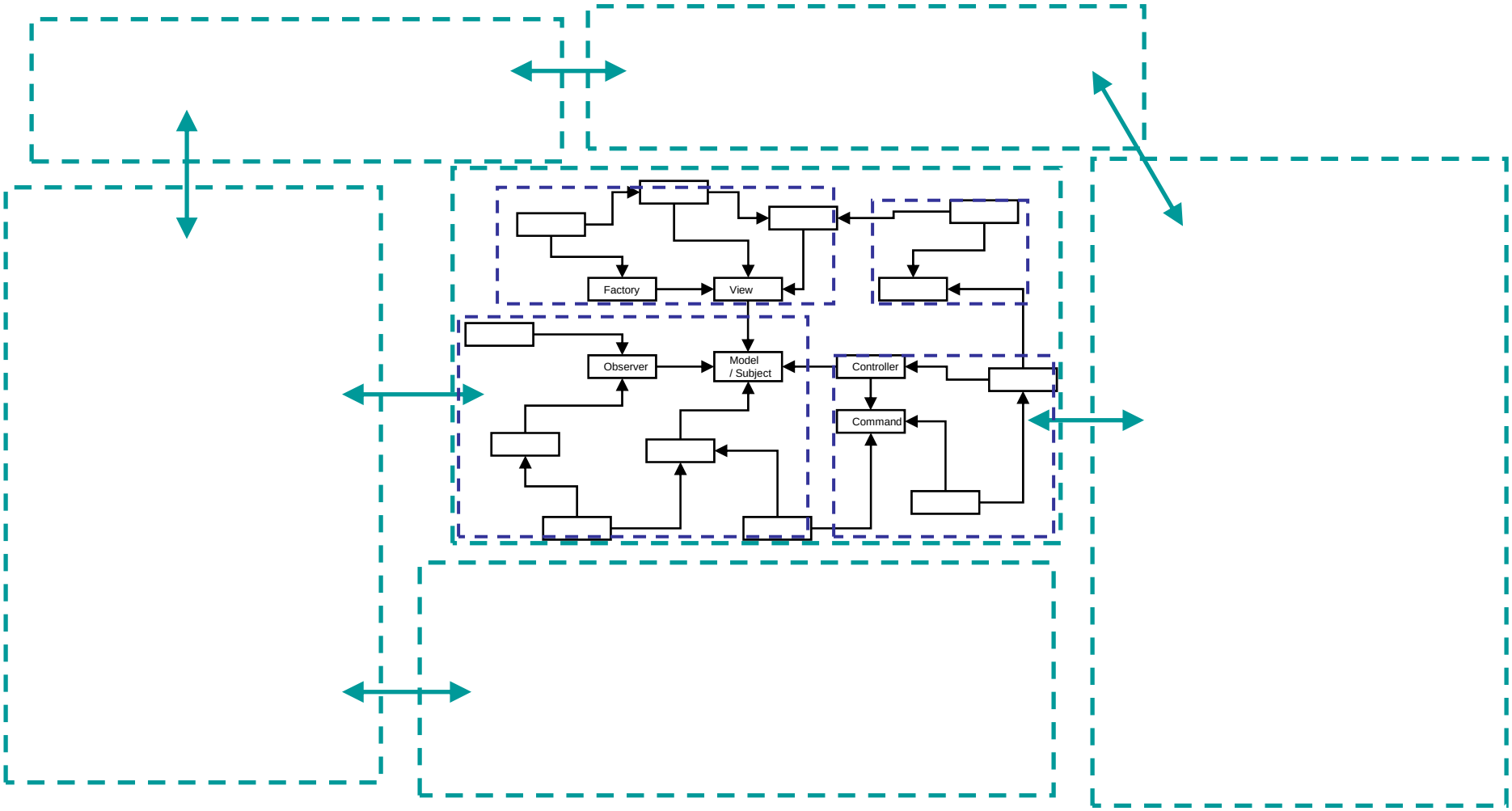


Dahl-Nygaard Lecture

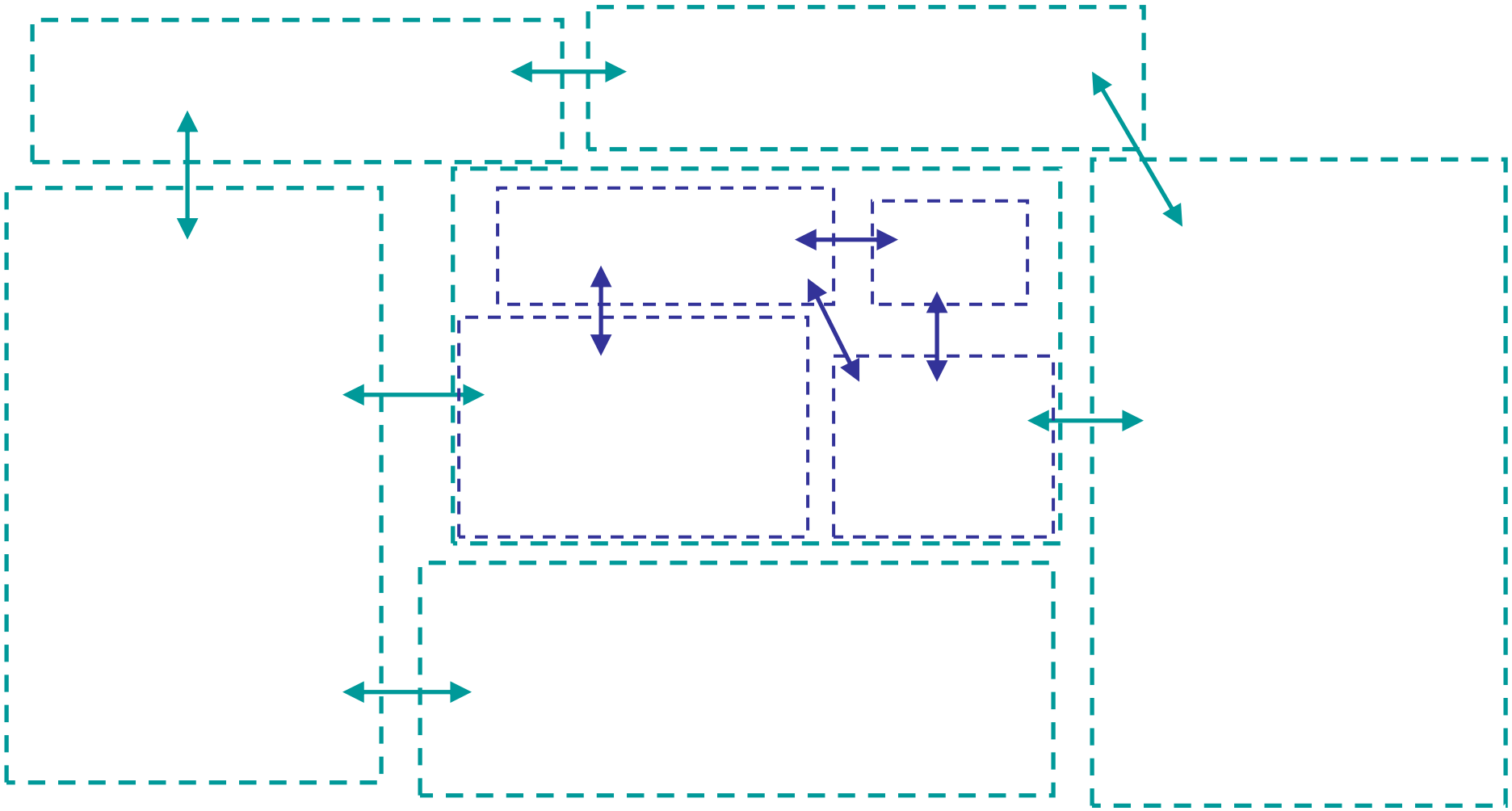
Objektorientierte Architekturen - über Entwurfsmuster hinausgehende Konzepte



Objektorientierte Architekturen - über Entwurfsmuster hinausgehende Konzepte



Objektorientierte Architekturen - über Entwurfsmuster hinausgehende Konzepte



Aspekte in der Software-Architektur

Statische Bestandteile

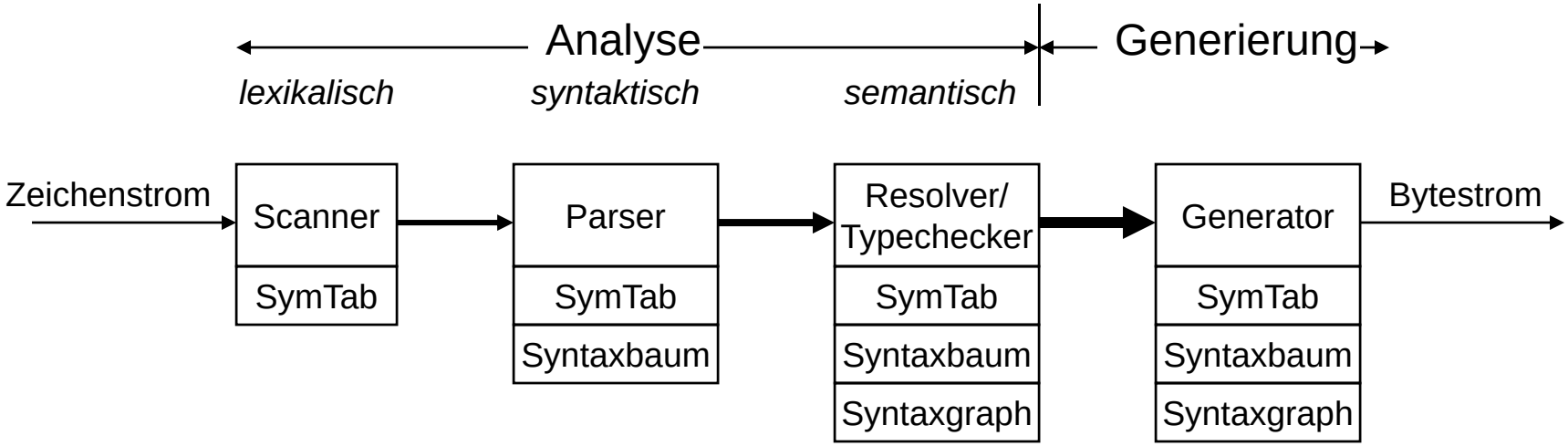
- Subsysteme (**Komponenten**)
- Beziehungen zwischen Subsystemen (**Konnektoren**)
- Klassifizierung von Subsystemen und deren Beziehungen (**Architektur-Stile**)

Dynamisches Verhalten

- **Schnittstellen** von Subsystemen
- **Szenarien** zwischen Komponenten

Funktionaler Entwurf von Compilern

Funktionen und Datenflüsse

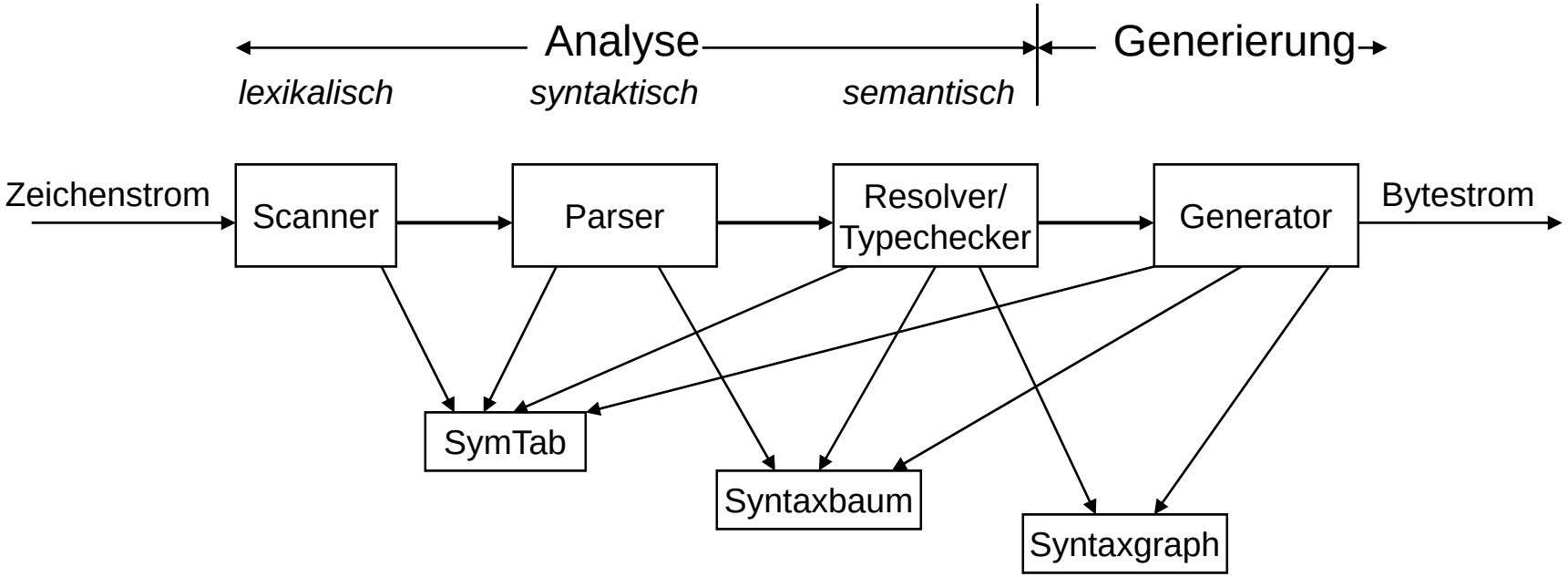


Kritik

- Wachsende Datenflüsse
- Ev. mehrfache Implementierung derselben Strukturen

Architektur mit Abstrakten Datentypen

Definition eigenständiger Datenstrukturen



Vorteile von ADTs

- Gekapselte Daten
- Zugriffsoperationen

Architekturzentrierte Entwicklung

1. Architektur-Stile
- Klassifikation von Systemen

Analyse

2. Anwendungsarchitektur
- Teil des Fach-Entwurfes

Entwurf

3. Technische Architektur
- Existierende Komponenten (Eigenentwicklung vs. Fremdanbieter)
- Technische Plattform / Infrastruktur

Realisierung

Wichtige Architektur-Stile

1. Datenflusssysteme

- Batch-sequentiell, Pipes and Filters

2. Call-and-return Systeme

- Hauptprogramm und Subroutinen, OO-Systeme, Hierarchische Schichten

3. Unabhängige Komponenten

- Kommunizierende Prozesse, Ereignisgesteuerte Systeme

4. Virtuelle Maschinen

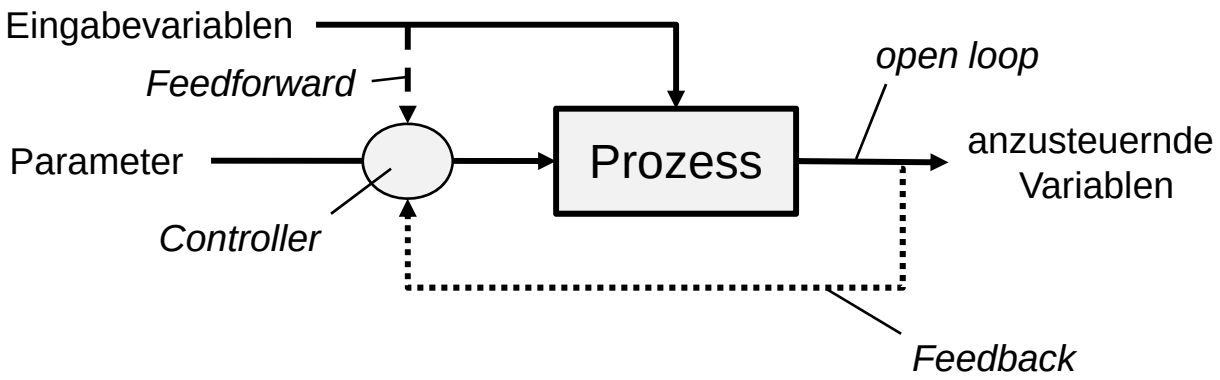
- Interpreter, Regelbasierte Systeme

5. Datenzentrierte Systeme (Repositories)

- Datenbanksysteme, Hypertext-Systeme, Blackboards

Stile (1): Prozesssteuerung

Struktur:



Konzepte:

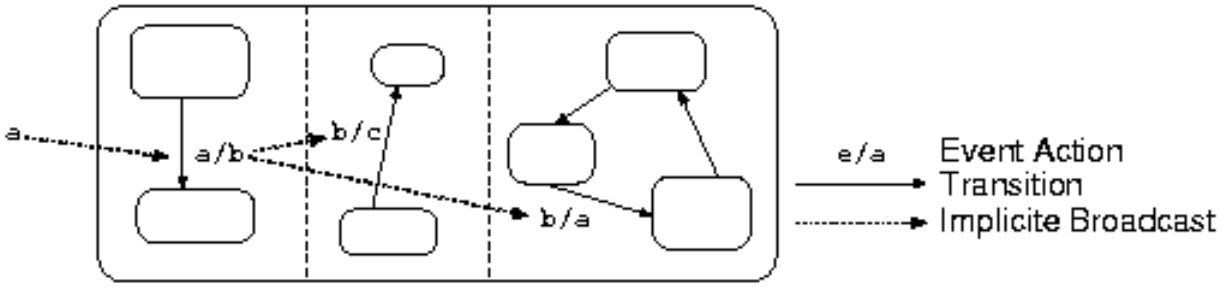
- Steuerung kontinuierlicher Prozesse
- Sensoren und Stellglieder
- Regelkreis: Verknüpfung eines Kontrollalgorithmus mit dem zu steuernden Prozess.
- Abbildung: technische Signale ↔ Messwerte

Variationen:

- Aktive Schleife
- Getaktete Berechnung
- Ereignissteuerung

Stile (2): Ereignisbasiert

Struktur:



Typische Eigenschaften:

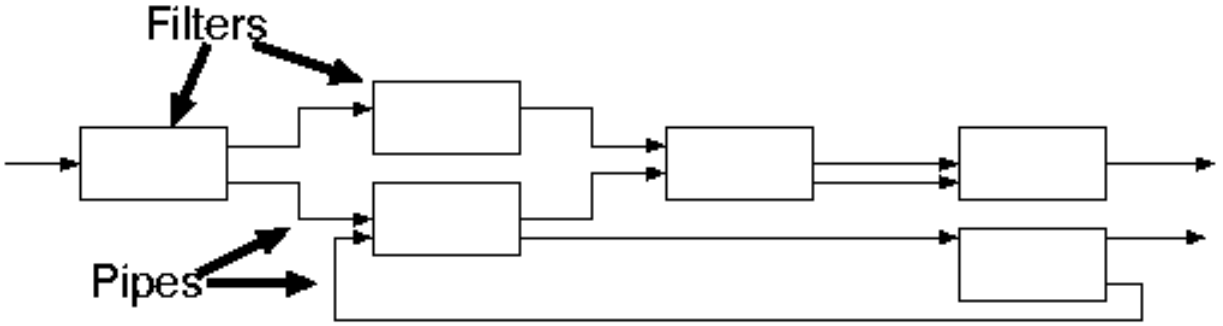
- Ein Teil der Kontrolle wird vom Aufrufer zum Aufgerufenen verlagert („broadcast“ – „subscribe“)
- Große Offenheit für das Hinzufügen von Komponenten
- Fast keine Gewissheit über die Reaktionen auf ein Event

Nachteil:

- Nicht **kompositional**: Effekt eines Events ergibt sich nur bei **Betrachtung des gesamten Systems**.

Stile (3): Pipes & Filters

Struktur:



Typische Eigenschaften:

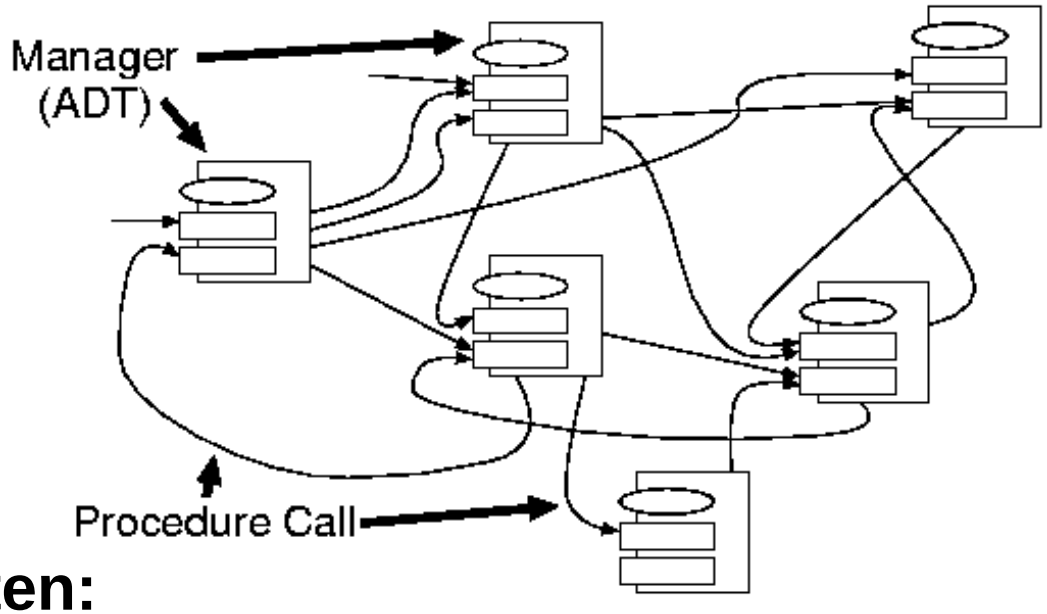
- Daten werden als kontinuierliche Ströme übertragen
- Berechnungen laufen inkrementell ab
- Keine weitere Kenntnis/Kommunikation zwischen Komponenten

Nachteile:

- Nicht interaktiv
- Standardisierung der Datenformate?

Stile (4): Objektorientierte Organisation

Struktur:



Typische Eigenschaften:

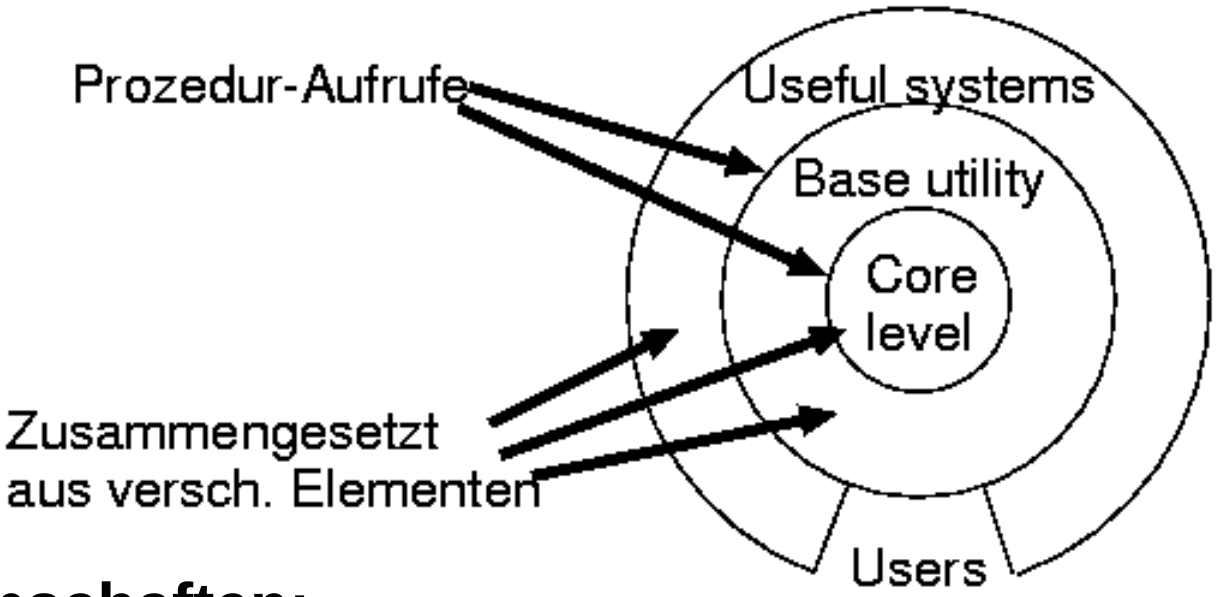
- Objektzustand ist **gekapselt**
- Objekt ist **verantwortlich** für seinen Zustand
- Systemzustand ist auf viele Objekte **verteilt**

Nachteil:

- Starke Vernetzung: Klassen sind u.U. von Details der Schnittstellen vieler anderer Klassen abhängig

Stile (5): Schichten

Struktur:



Typische Eigenschaften:

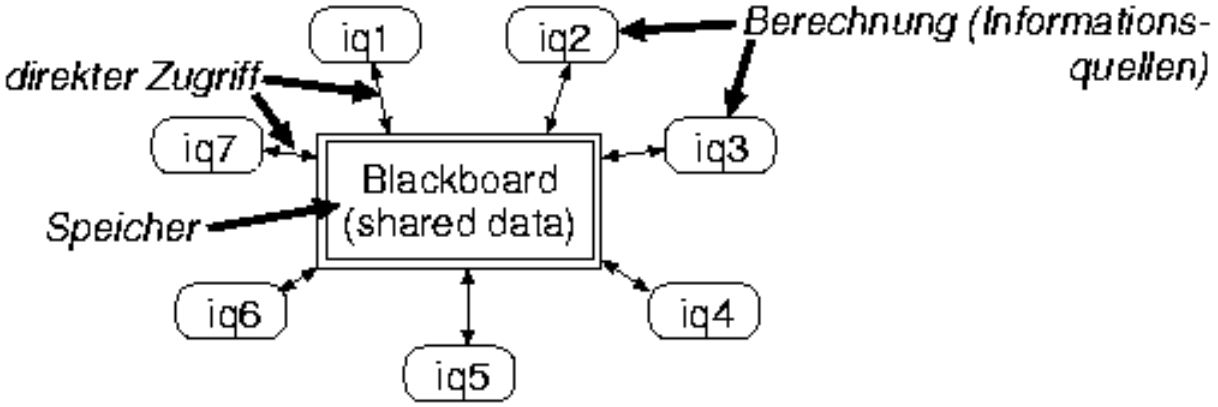
- Innere Schicht ist nur für die direkt umschließende sichtbar
- Explizite Schnittstellen
- „Abstrakte Maschinen“

Nachteil:

- Schichtenlösungen sind häufig schwierig zu finden bzw. ineffizient.

Stile (6): Repositories

Struktur:



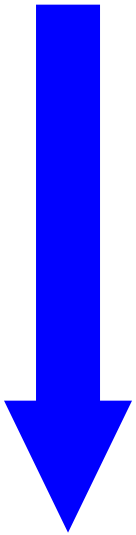
Typische Eigenschaften:

- **Informationsquellen** (Klienten): Erzeugen verändern Daten, können leicht hinzugefügt/ausgetauscht werden
- **Zentrale Datenstruktur**: Mehr oder minder aktiver Speicher.
- Steuerung mittels bidirektionaler Kommunikation
- Keine direkte Kommunikation zwischen Klienten.

Varianten:

- Insbesondere verschiedene Realisierungen der Kommunikation

Architekturstile: Konsequenzen



- Problemklasse
- Architekturstil
- Benötigte Techniken,
  (Sprachen, Kommunikation, Bibliotheken)
- Entwicklungsmethode

Aspekte in der Technischen Architektur

- Auswahl vorgefertigter Komponenten
- Infrastruktur
  - Betriebssystem
  - Middleware
  - Datenbanksystem
  - Frameworks
  - Programmiersprache(n)
  - Bibliotheken
- Kompatibilität
  - Durch Beachtung von Standards
  - Unter Berücksichtigung exakter Versionen
  - Durch Adaptierung

Architektur-Prozesse

- **Ideal: sequentieller Prozess**

  Architekturstil → Anwendungsarchitektur → technische Architektur
  - Jeweils nur eine Abstraktionsebene!

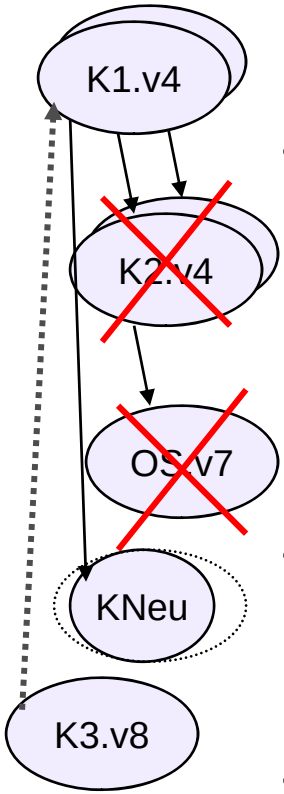
- **Existierende IT-Landschaft**
  - Festlegung auf Techniken
  - Existierende Daten, Datenbanken, Datenformate, Standards ...

- **Wiederverwendung**
  - Frameworks, Bibliotheken *(interne Schnittstellen)*
  - „Commercial Off The Shelf“ (COTS) Komponenten *(externe S.)*

    „Kommerzielle Produkte aus dem Regal“

Prozesssingularität

Alle Phasen gleichzeitig:



- **Wartung**:
  Integration einer neuen Version von Komponente K1.v3
- **Entwurf**:
  Änderungen an Schnittstellen nachziehen (K2)
  Abhängigkeiten nachziehen:
  - K1.v4 erfordert K2.v4 erfordert OS.v7
  - Kunde hat Lizenz nur für OS.v6
  - K2 kann nicht mehr verwendet werden
- **Analyse**:
  - Verteilung der Funktionalität von K2 auf andere Komponenten?
  - Eigenentwicklung eines Ersatzes für K2 (teilweise)?
- **Implementierung/Testen**:
  - Ist K1.v4 kompatibel mit K3.v8 (keine Angaben in der Doku)?