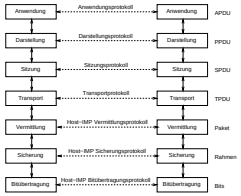
Muster in der Software–Technik

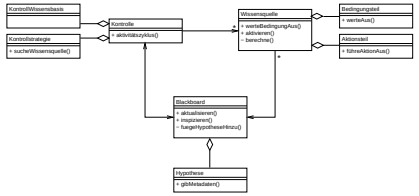
Architekturmuster – Strukturierung von Systemen

Dipl.-Informatiker Adriano Gesué

Themen der heutigen Vorlesung:

- Organisatorisches
- Inhaltliches
  - Software–Architektur
  - Architekturmuster – Allgemeines
  - Architekturmuster zur Strukturierung von Systemen
    - Schichten–Architektur
    - Blackboard–Architektur





- Literatur

Software–Architektur:

Eine Software–Architektur

- ist eine Beschreibung von
  - Subsystemen
  - Komponenten
  - Beziehungen zwischen diesen
- ist ein Artefakt
- entsteht durch Aktivitäten des Systementwurfs

Referenzmodell für Muster:

- Arten von Mustern
  - Software–Muster
    - Architekturmuster
    - Entwurfsmuster
    - Idiome
  - Prozessmuster
- Einsatzgebiet von Mustern
- Inhärentes informatisches Konzept bei Mustern
- Komplexität

Themen der heutigen Vorlesung:

- Organisatorisches
- Inhaltliches
  - Architekturmuster – Allgemeines
  - Architekturmuster zur Strukturierung von Systemen
    - Schichten–Architektur
    - Blackboard–Architektur
- Literatur

Architekturmuster:

Ein Architekturmuster

- beschreibt die grundlegende Organisationsstruktur von Software–Systemen
- beschreibt systemweite Eigenschaften
- ist eine Schablone für konkrete Software–Architekturen
- ist eine Abstraktion konkreter Software–Architekturen

Deshalb ist die Wahl eines Architekturmusters eine weitreichende Entscheidung bei der Entwicklung von Software–Systemen.

Anforderungen in der Software–Entwicklung:

- funktionale Anforderungen: was soll mein System machen?
- nichtfunktionale Anforderungen: wie ist mein System beschaffen?
  - Änderbarkeit
  - Erweiterbarkeit
  - Austauschbarkeit
  - Portabilität
  - Wiederverwendbarkeit
- Eine Unterteilung des Systems bietet sich sowohl aus funktionalen als auch nichtfunktionalen Aspekten an.

Themen der heutigen Vorlesung:

- Organisatorisches
- Inhaltliches
  - Architekturmuster – Allgemeines
  - Architekturmuster zur Strukturierung von Systemen
    - Schichten–Architektur
    - Blackboard–Architektur
- Literatur

Muster–Schema:

- Strukturierung
  - Name
  - Kontext
  - Problem
  - Lösung
  - Beispiel
  - Varianten
  - Implementierung
  - Gegenanzeigen
- Klassifizierung
- Kombination

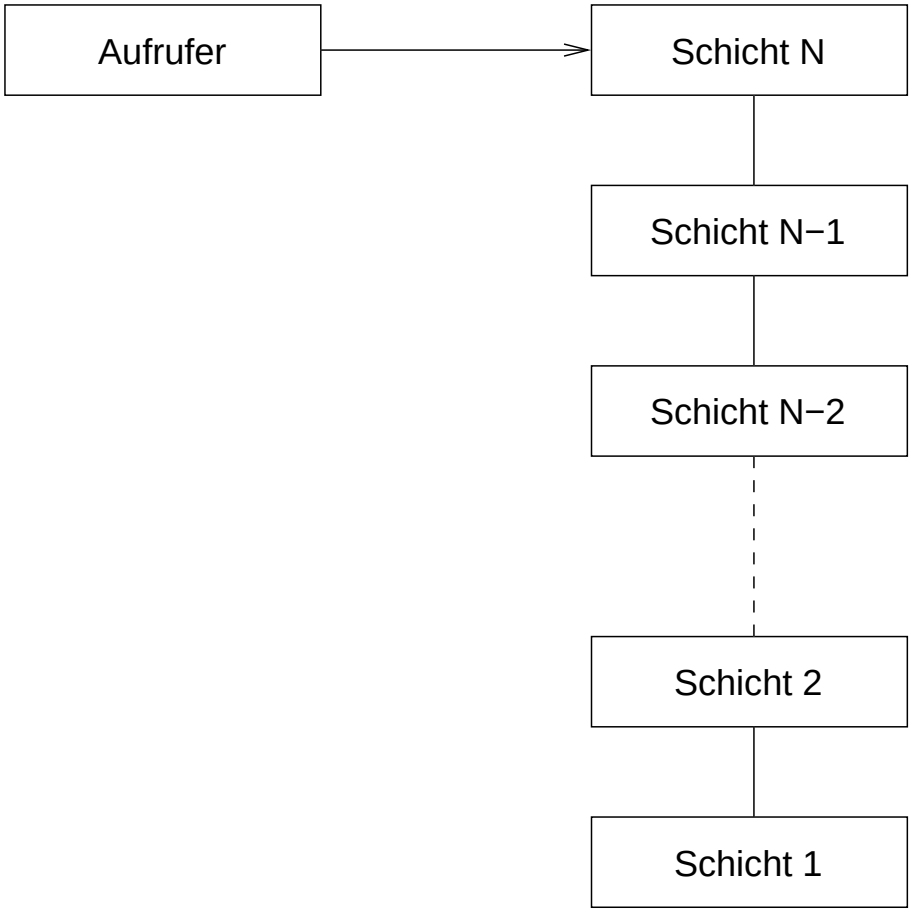
Schichten–Architekturmuster:

- **Kontext:** Große Systeme, die einer Zerlegung bedürfen.
- **Problem:** Große Software–Systeme besitzen häufig eine Mischung aus Operationen mit unterschiedlichen Abstraktionsgraden.
  - Operationen höheren Abstraktionsgrads beziehen sich auf niedrigere
  - Code–Änderungen sollen sich auf die jeweilige Komponente beschränken (Lokalität)
  - Robustheit der Schnittstellen
  - Austauschbarkeit von Komponenten, Alternativimplementierungen
  - Wiederverwendbarkeit von Komponenten

Lösung:

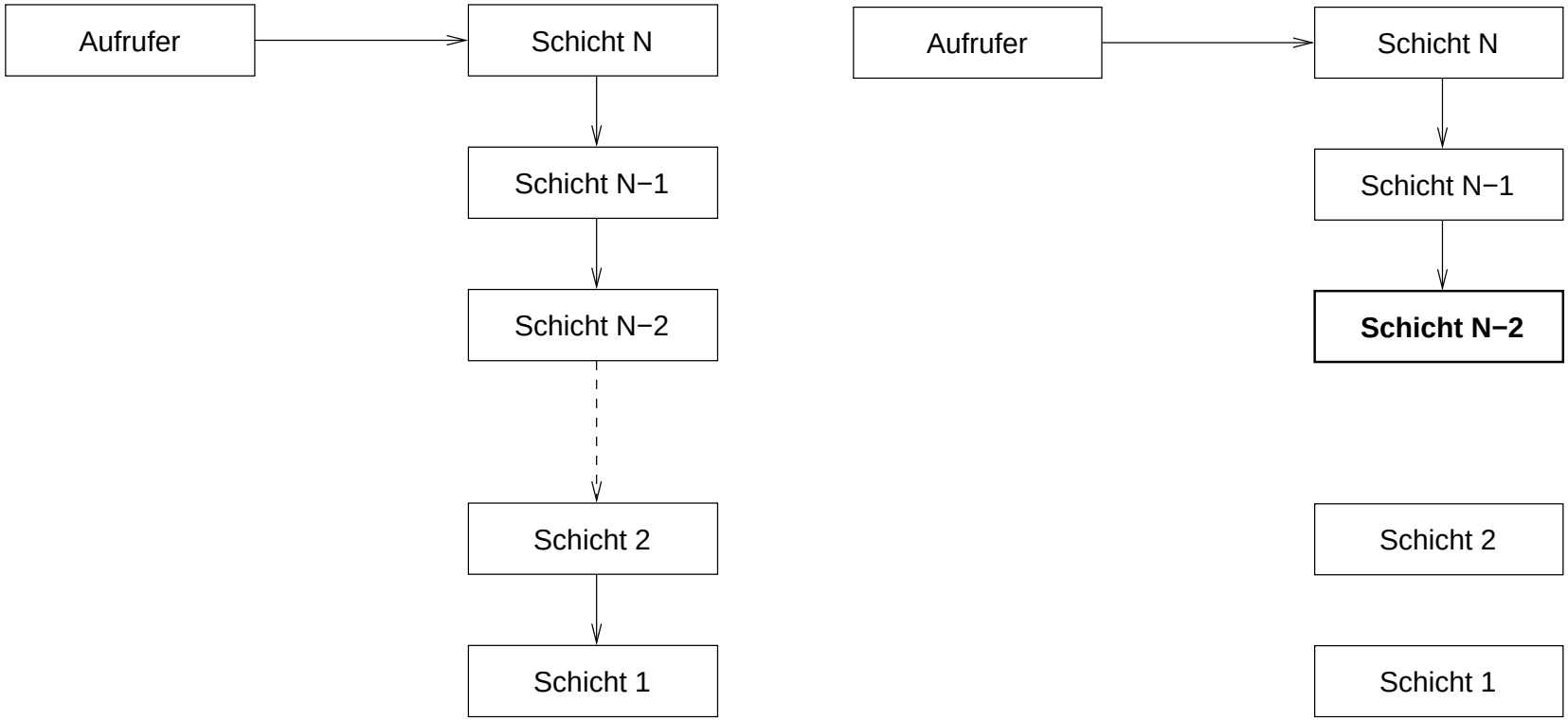
- Gruppiere Operationen gleichen Abstraktionsgrades zusammen
- Dabei entsteht für jeden Grad der Abstraktion eine **Schicht**.
- Dienste höherer Schichten stützen sich auf Dienste niedrigerer Schichten ab
- Schichten möglichst weit voneinander entkoppeln (Lokalität): Schichten interagieren jeweils nur mit oberer und unterer Schicht
- Einheitliche Schnittstelle für jede Schicht erhöht Austauschbarkeit, Wiederverwendbarkeit – Fassade– und Brücken–Muster

Schematische Darstellung der Schichten:



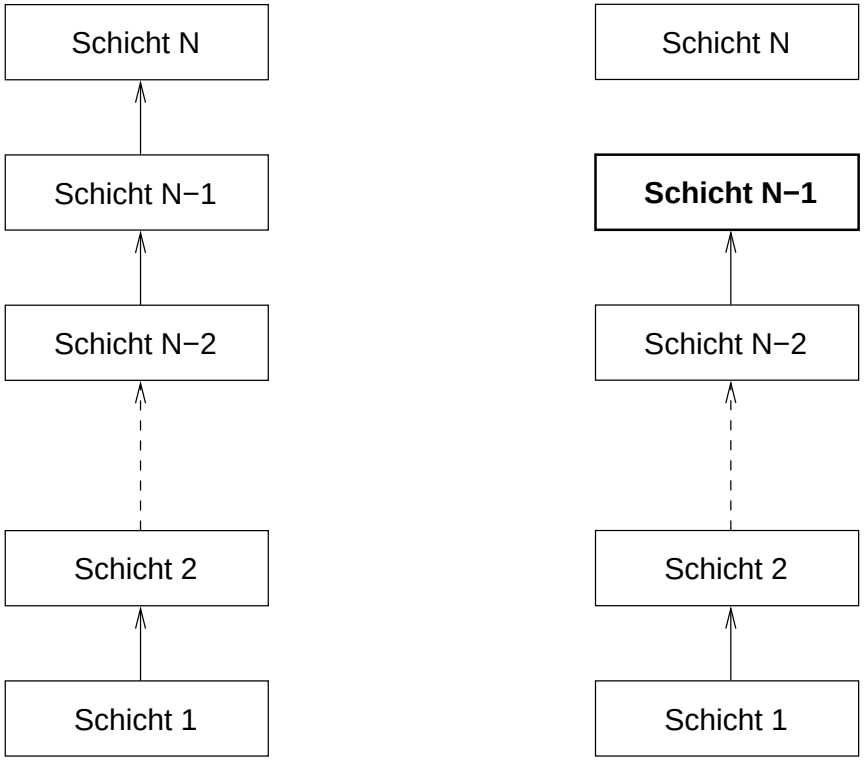
Szenarien für Aufträge höherer Schichten:

Schicht j benötigt zur Bearbeitung des Auftrages Dienste aus der Schicht $j-1$.

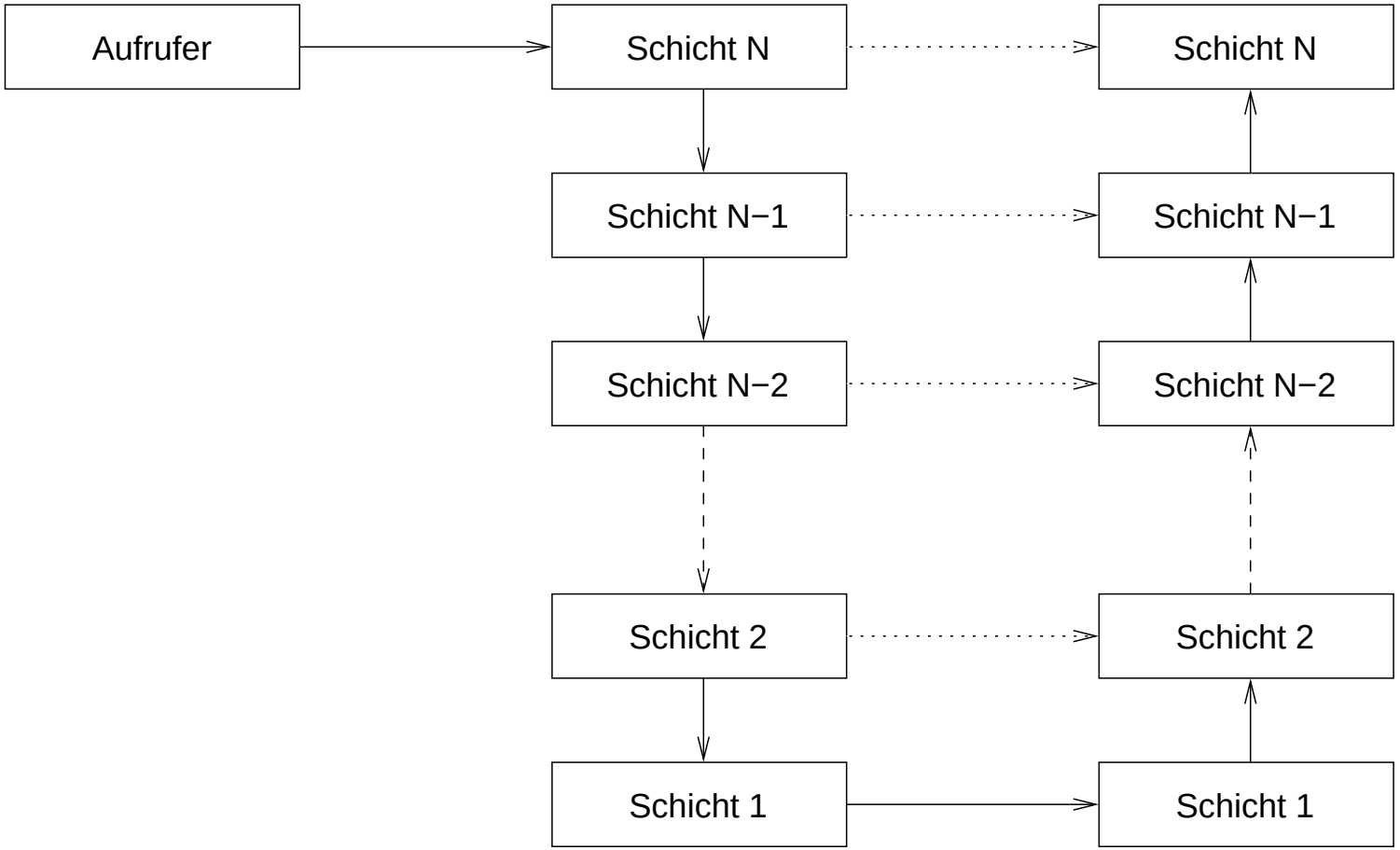


Szenarien für Benachrichtigung durch niedrigere Schichten:

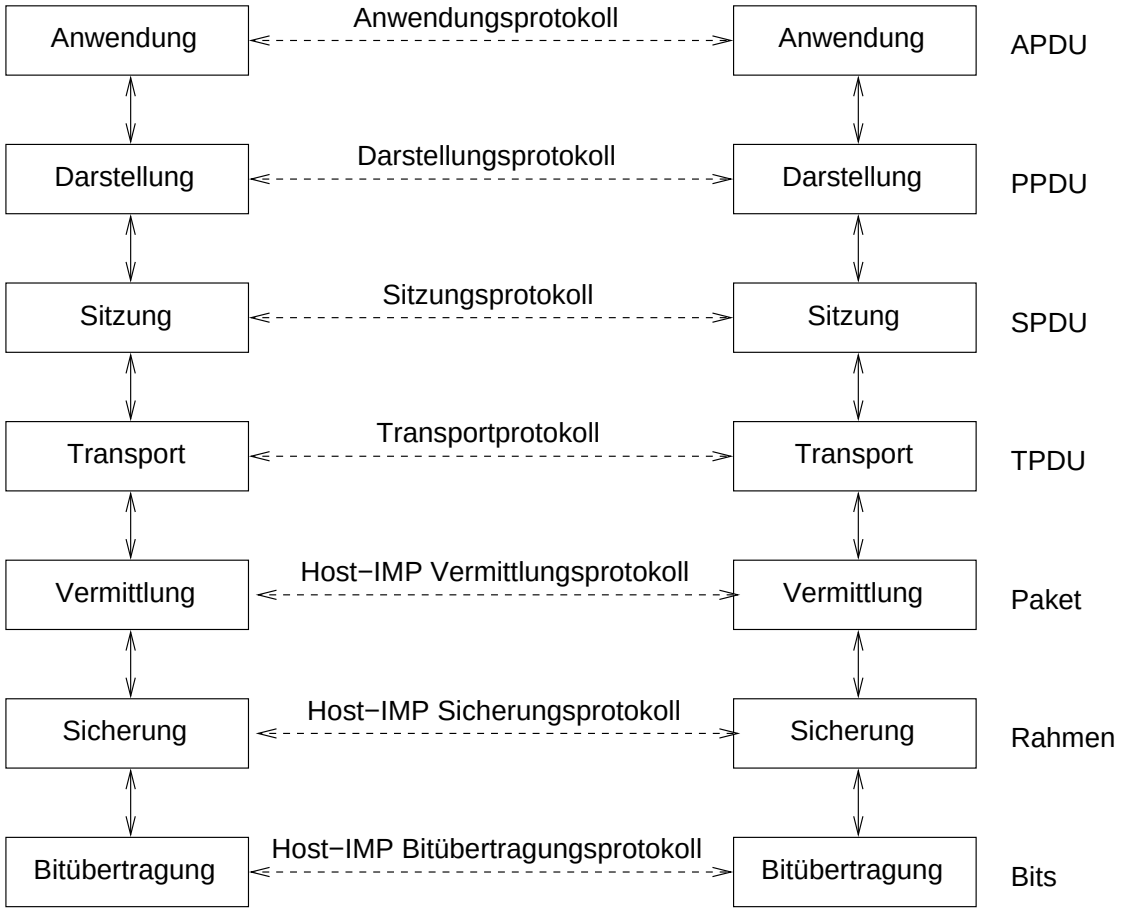
untere Schicht benachrichtigt obere Schicht über Ereignis



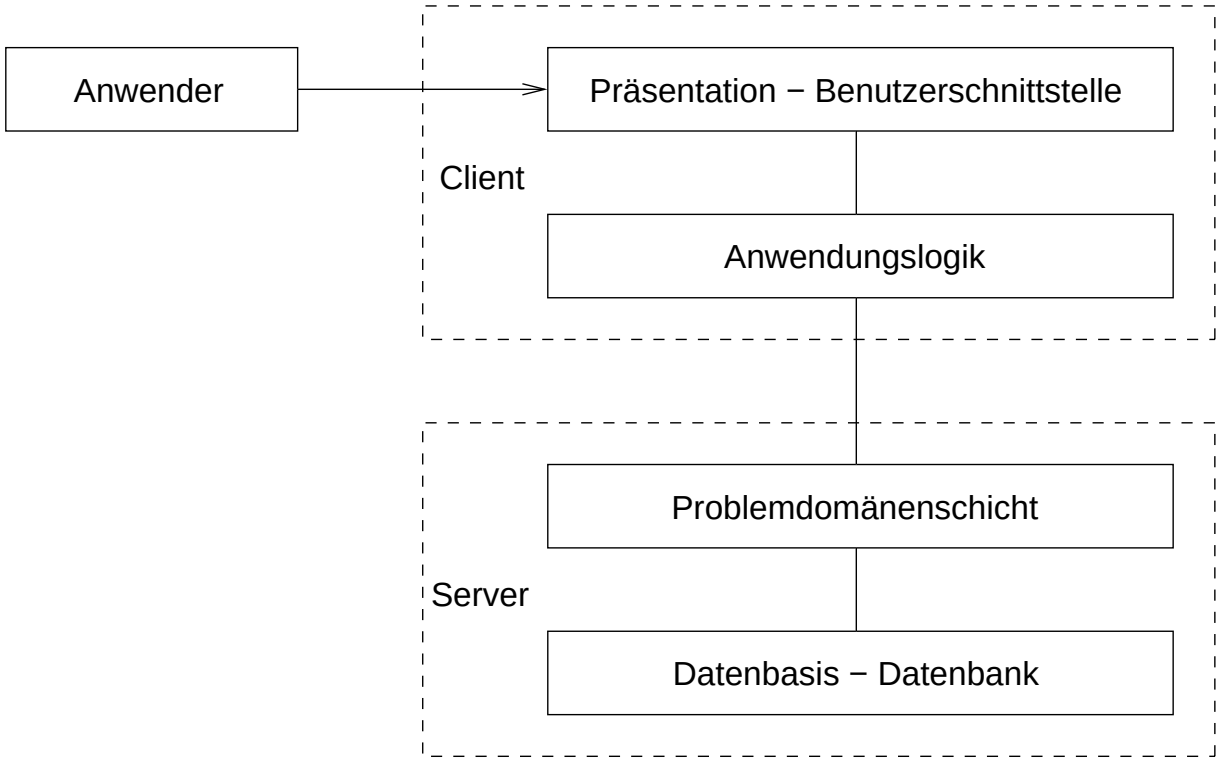
Szenarium für Interaktion zweier geschichteter Systeme:



Beispiel: OSI–Siebenschichtenmodell



Beispiel: Betriebliche Informationssysteme



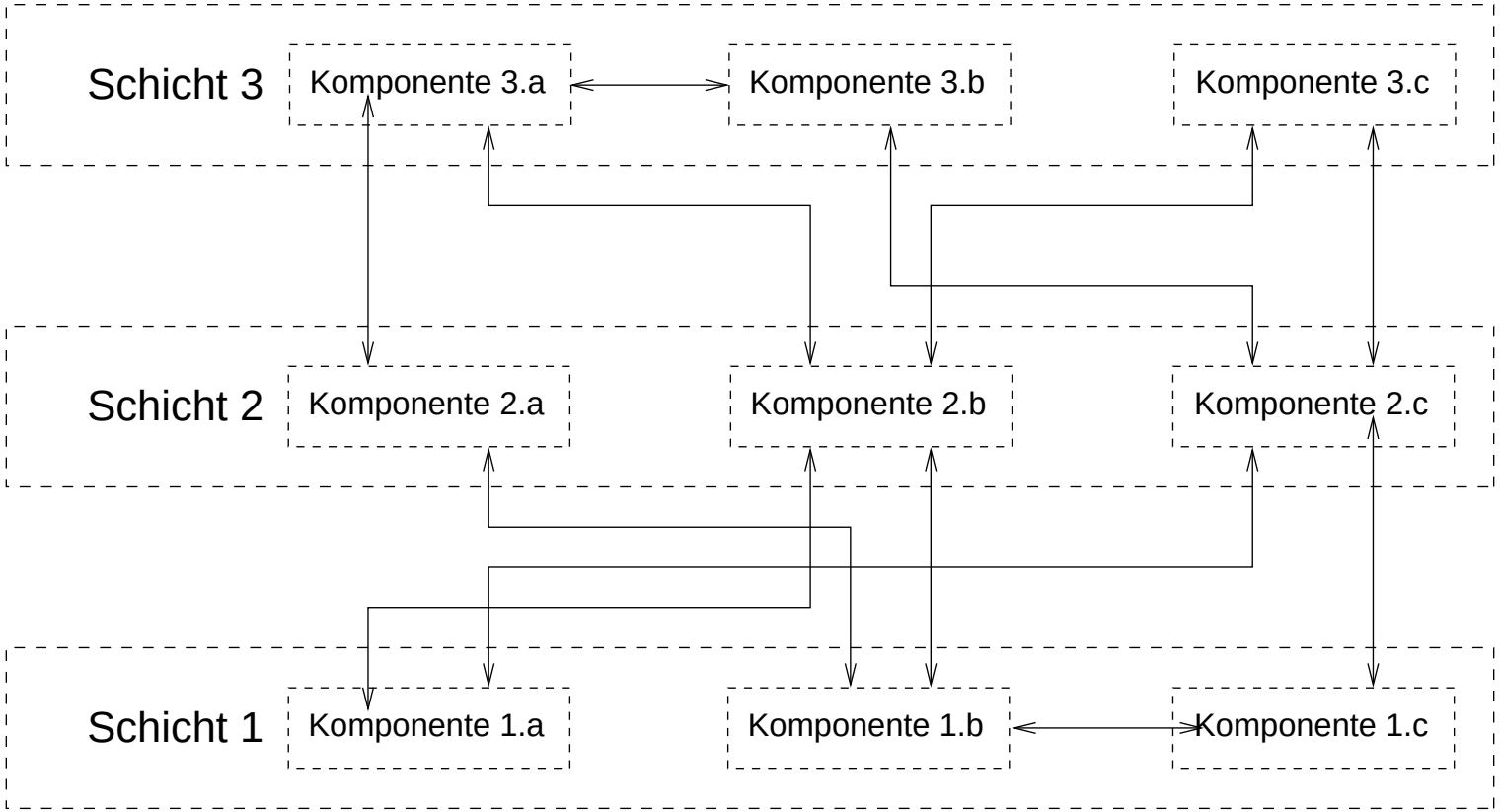
Die Zuordnung der einzelnen Schichten zu Client bzw. Server kann variieren

Implementierung:

- Schichten–Architekturen sind konzeptionell unabhängig von objektorientierten Ansätzen
- Umsetzung ist auch in anderen Programmierparadigmen möglich
- Unterscheidung nach Grad der Kopplung
  - Kopplung zwischen Komponenten
  - Kopplung zwischen Schichten

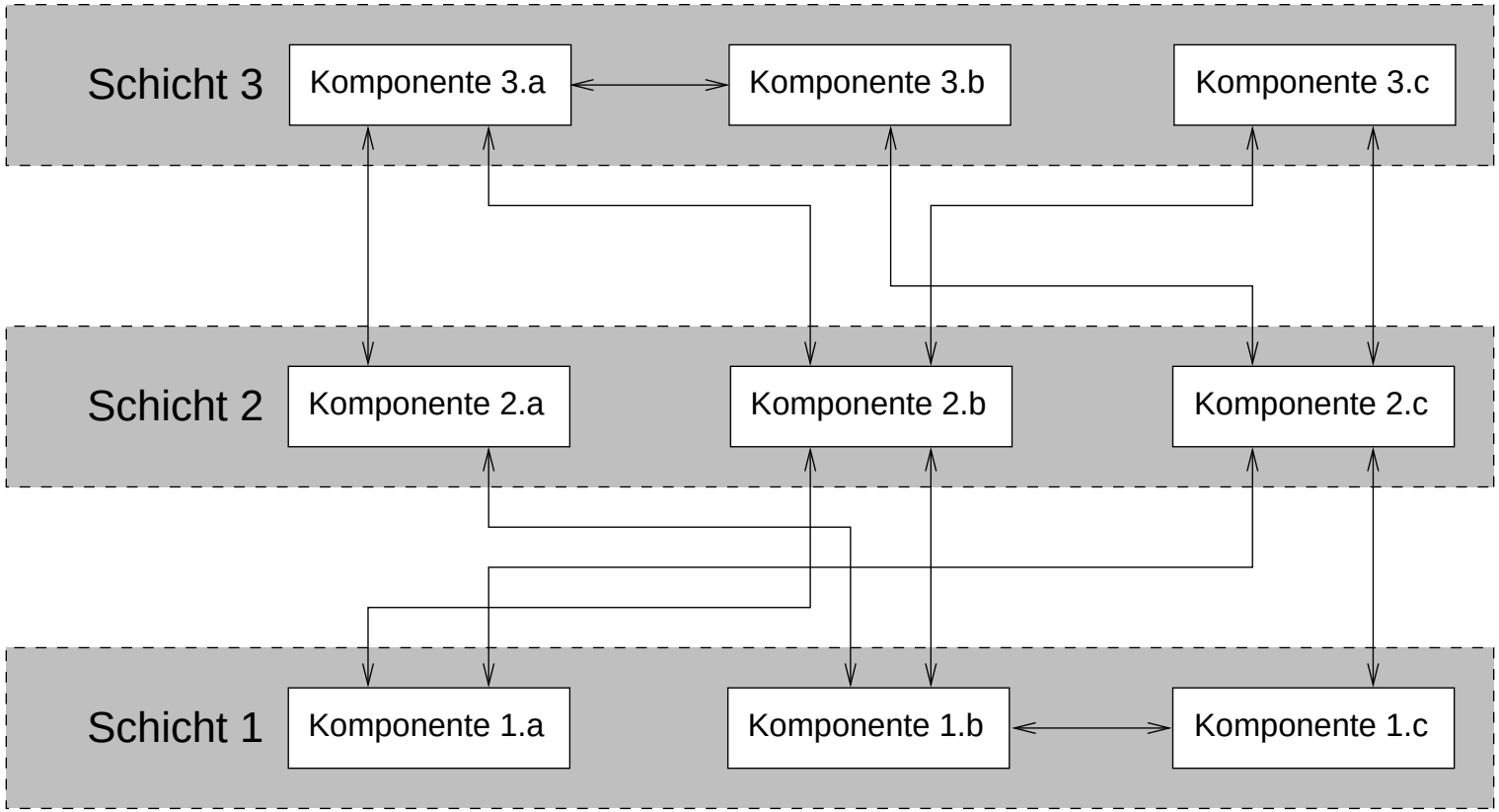
White–Box–Interaktion:

Komponenten sehen Interna der Komponenten anderer Schichten



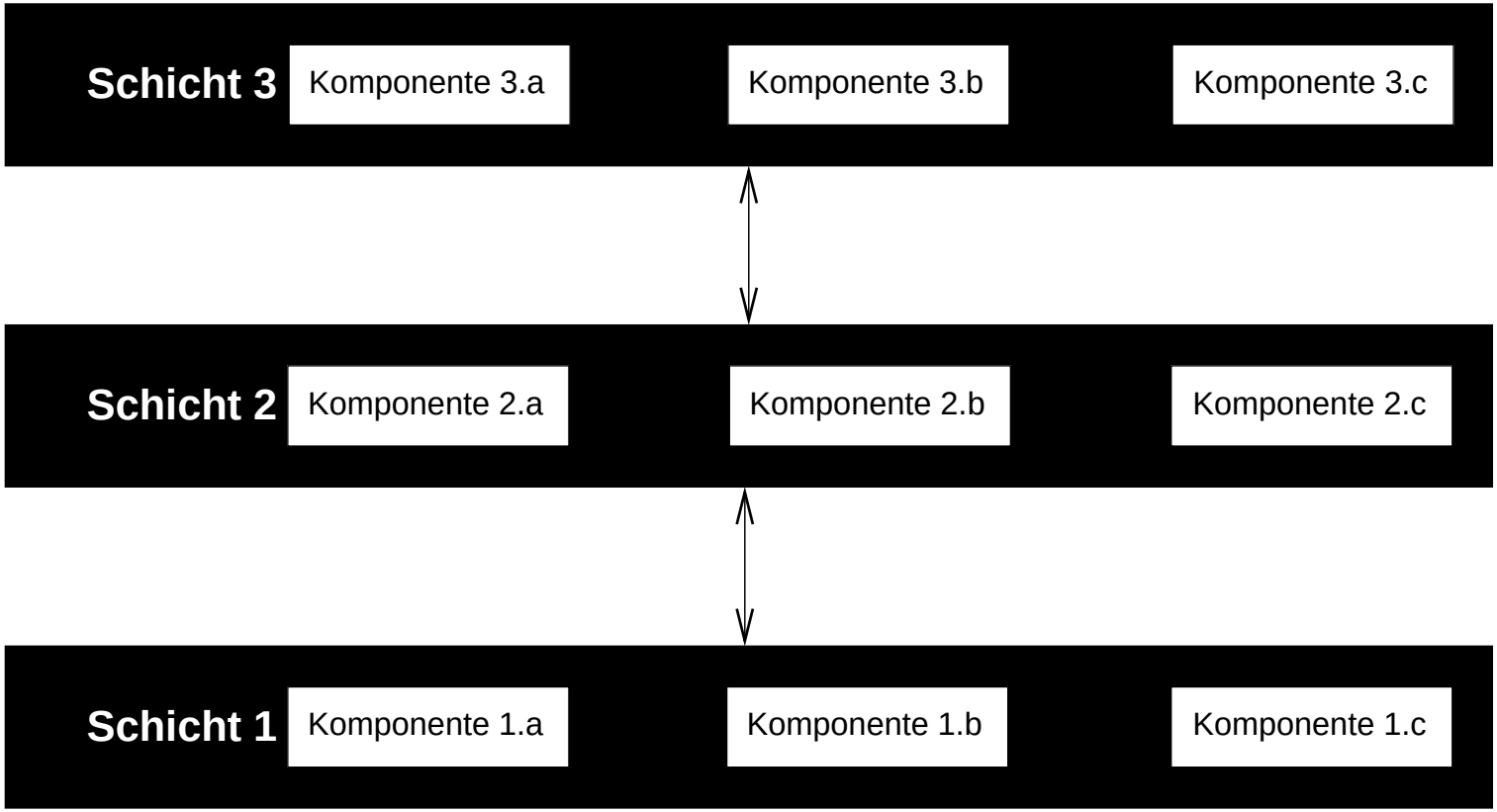
Grey–Box–Interaktion:

Komponenten rufen Dienste direkt bei Komponenten anderer Schichten auf



Black–Box–Interaktion:

Komponenten interagieren nicht direkt miteinander

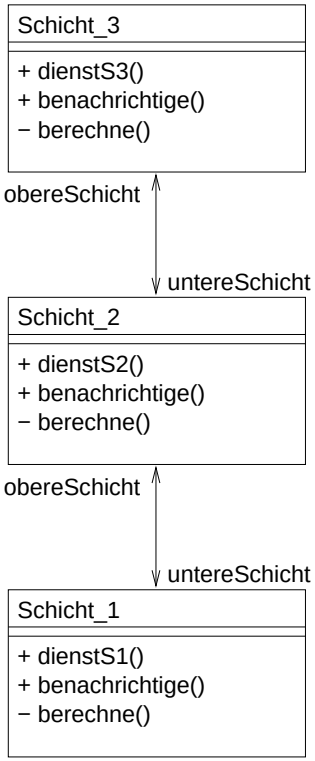


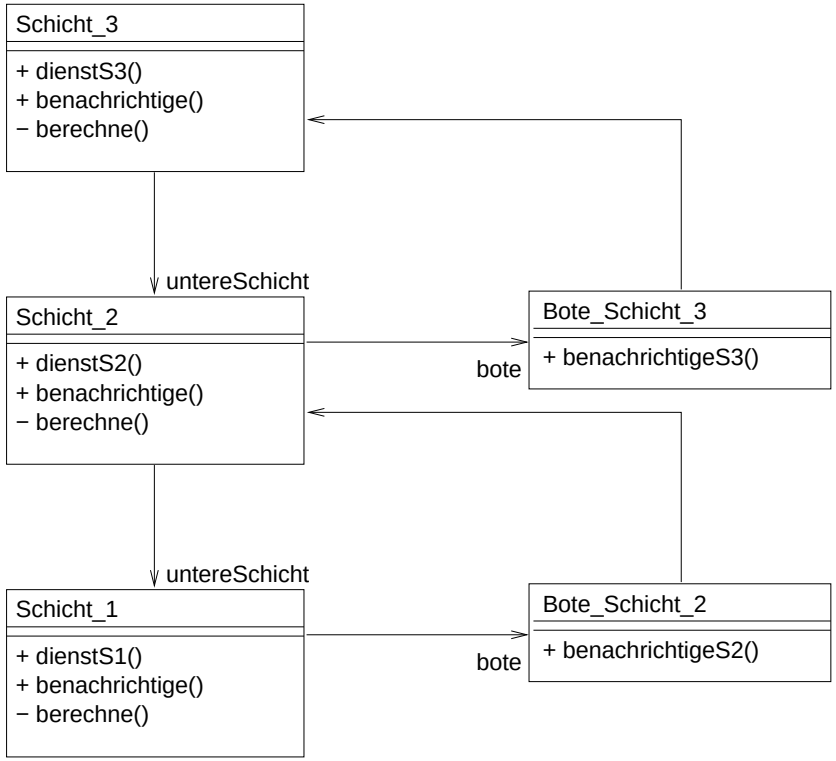
Entkopplung der Schichten:

- nicht nur Komponenten der Schichten von Komponenten anderer Schichten entkoppeln (black–box)
- auch einzelne Schichten weitgehend voneinander entkoppeln:
  - lokale Behandlung von Änderungen
  - erhöhte Austauschbarkeit

Entkopplung der Schichten:

Vorher: Schicht kennt obere und untere Schicht und ruft direkt Dienste auf

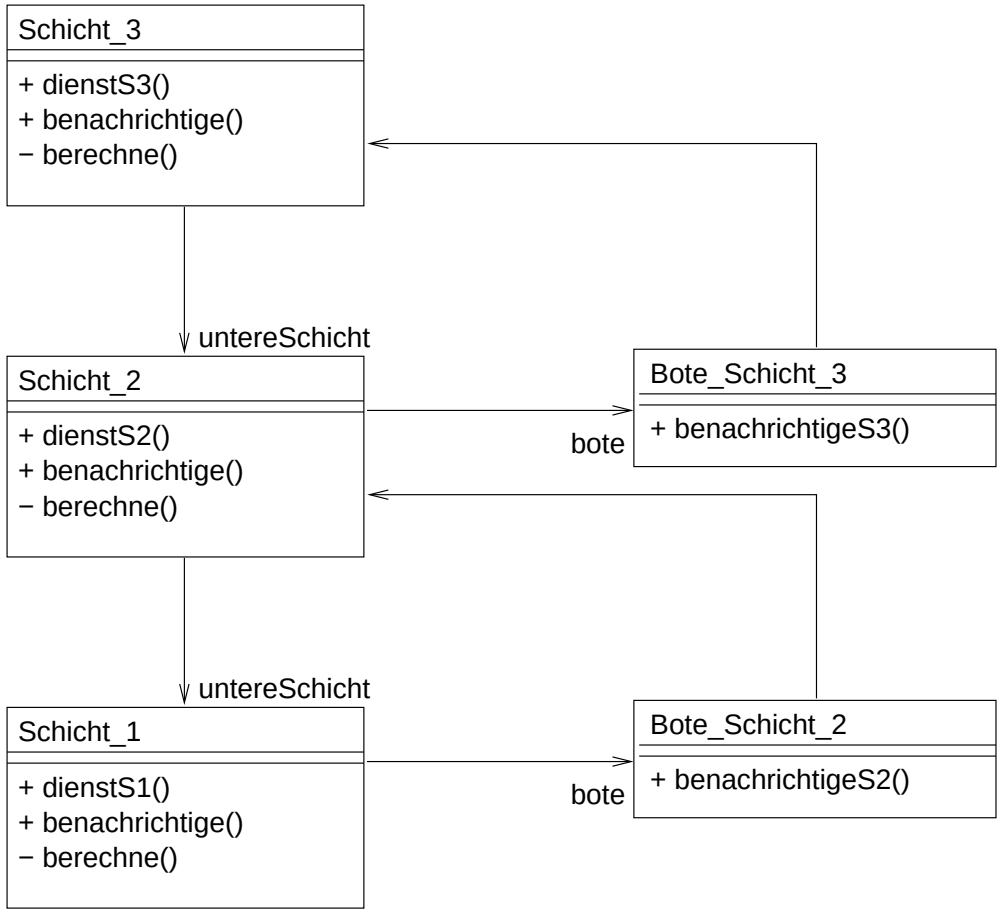




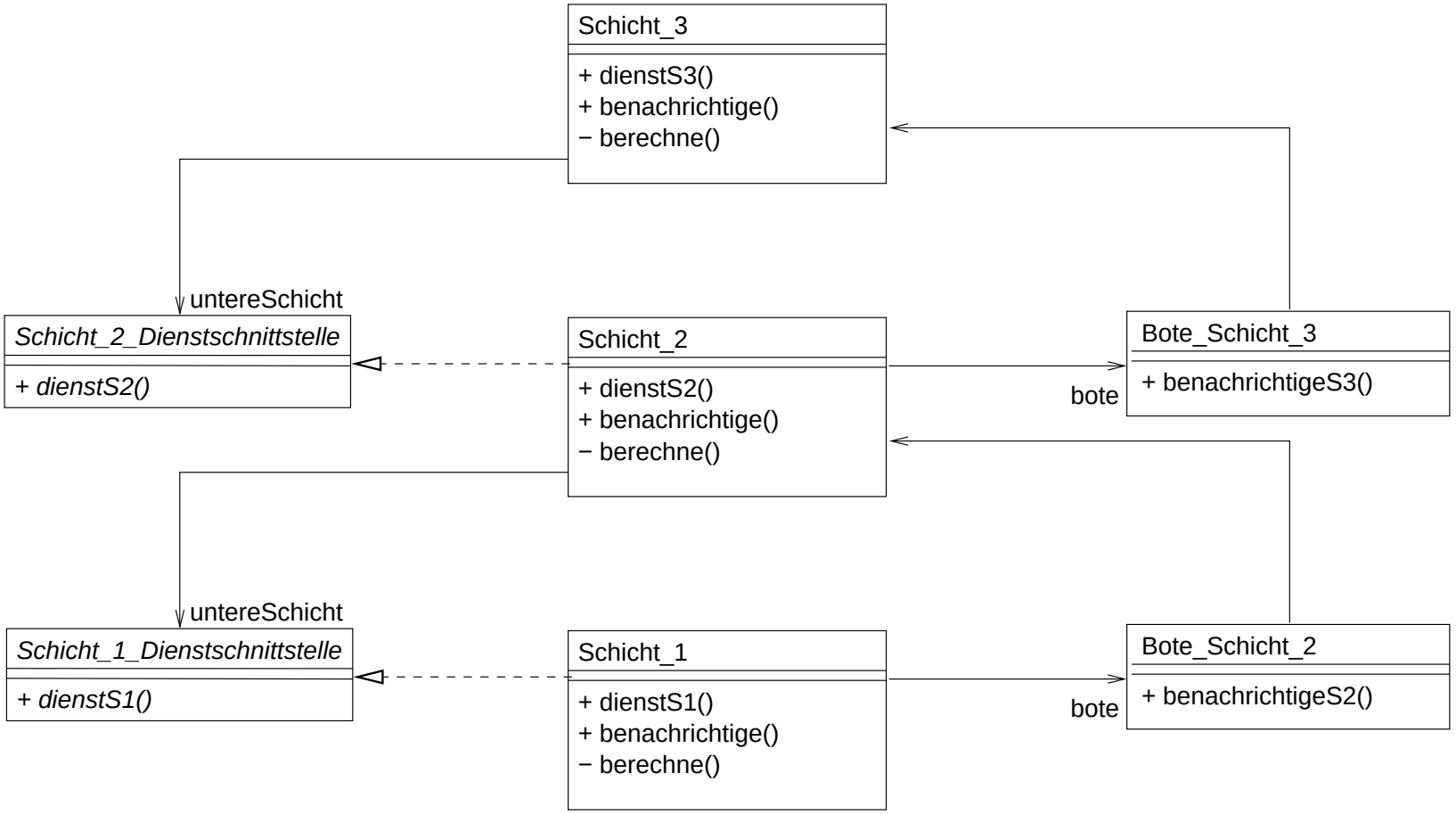
Nachher: untere Schicht hat Referenz für Verschickung von Benachrichtigungen – callbacks oder Befehls–Muster

Entkopplung der Schichten:

Vorher: obere Schicht kennt untere Schicht und ruft direkt Dienste auf.



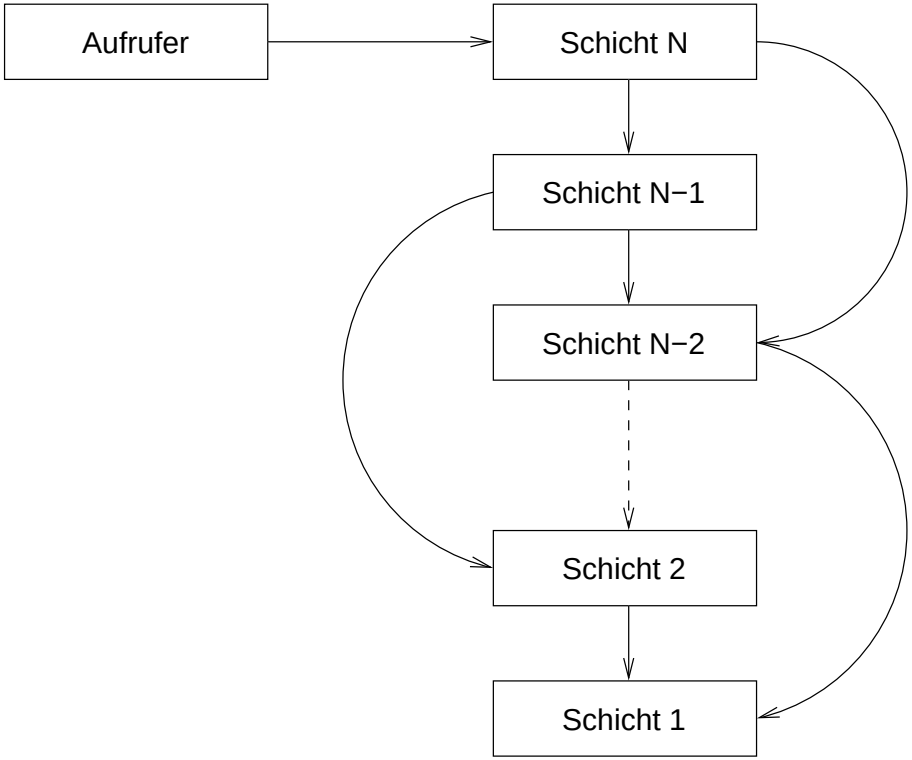
Entkopplung der Schichten:



Nachher: gegen Schnittstelle programmiert, Implementierung austauschbar

Varianten: Gelockerte Schichtung

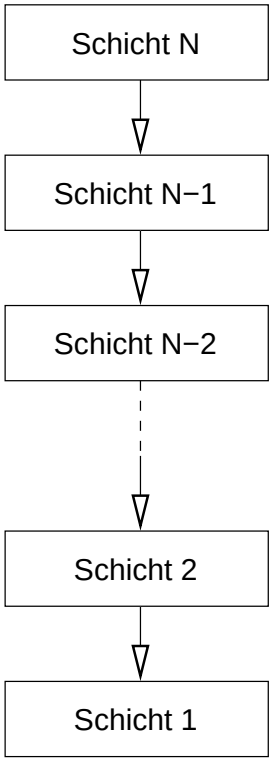
Schichten interagieren auch mit weiter entfernten Schichten

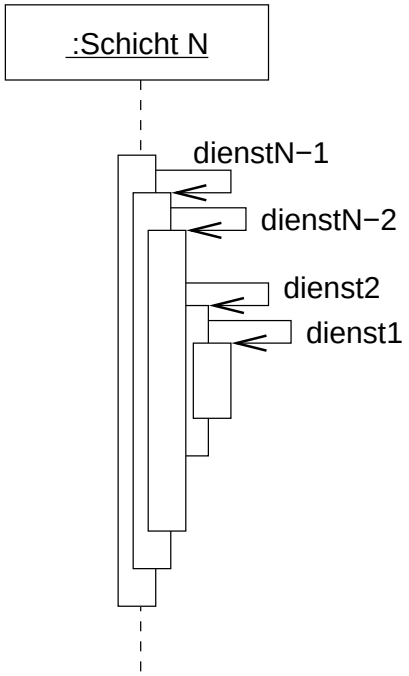


Vorteil: Performanzsteigerung

Nachteil: Wartbarkeit, Änderbarkeit geringer

Varianten: Schichtung durch Vererbung





Vorteil: Modifikation von Diensten möglich auf höherer Ebene

Nachteil: Wartbarkeit, Änderbarkeit geringer

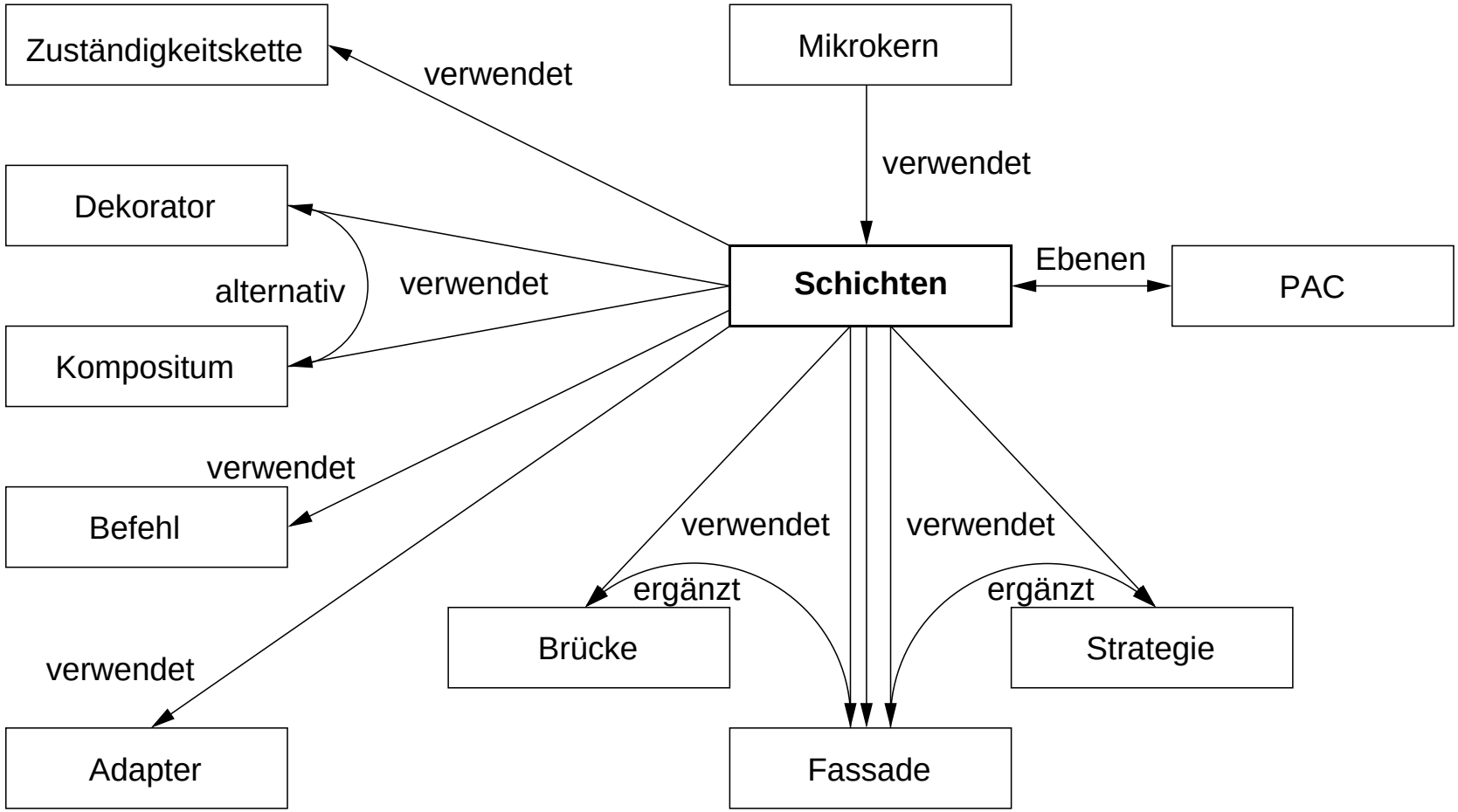
Gegenanzeigen:

- Monolithisches System ist im Allgemeinen effizienter
  - Übertragungsaufwand
  - Transformationsaufwand
- Vervielfachte Arbeit durch mehrere high–level Aufträge auf selben Daten
- Granularität der Schichten schwer zu bestimmen:
  Wiederverwendbarkeit vs. Overhead

Klassifizierung des Schichten–Musters:

- **Art:** Software–Muster: Architekturmuster
- **Einsatzgebiet:** allgemein
- **Konzept:** Interaktionsregelung, (De)Komposition
- **Komplexität:** einfach, viele Diskussionspunkte, variantenreich

Beziehungen zu anderen Mustern:



Themen der heutigen Vorlesung:

- Organisatorisches
- Inhaltliches
  - Architekturmuster – Allgemeines
  - Architekturmuster zur Strukturierung von Systemen
    - Schichten–Architektur
    - Blackboard–Architektur
- Literatur

Blackboard–Architekturmuster:

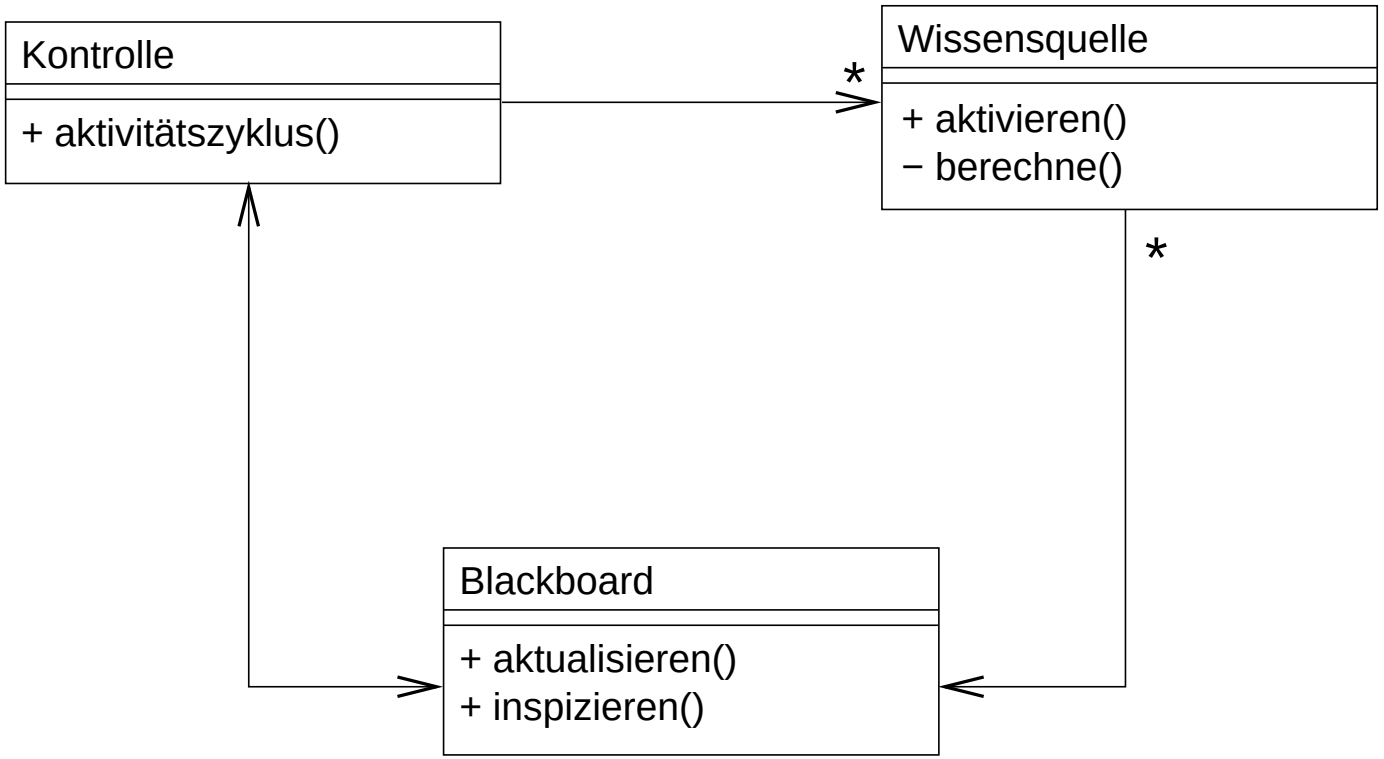
- **Kontext:** Schlecht strukturiertes oder (noch) wenig erschlossenes Gebiet, in dem geschlossene Lösungen nicht bekannt/möglich sind.
- **Problem:** Für bestimmte Probleme in diesem Gebiet ist keine Lösung mit festen Teilschritten bekannt.
  - Wandlung von Rohdaten in Informationen hohen Abstraktionsgrads
  - Teilprobleme erfordern unterschiedliche Expertise
  - für Lösungen der Teilprobleme unterschiedliche Darstellungen und Mittel
  - Oft keine vorhersagbare Strategie zur Kombination von Teillösungen
  - Unsicheres Wissen und unzuverlässige Teillösungen
  - verschiedene Algorithmen für gleiches Teilproblem – Strategie

Lösung:

System wird aufgefasst als

- Ansammlung unabhängiger Programme, jedes spezialisiert auf ein Teilproblem = Wissensquelle
- gemeinsame Nutzung und Arbeit auf einer Datenstruktur = Blackboard
- keine festen Arbeitsschritte, sondern geleitet an momentanem Bearbeitungszustand (opportunistisches Problemlösen)
- zentrale Komponente bewertet Zustand und koordiniert Spezialprogramme = Kontrolle

Grobübersicht der Architektur:



Verfeinerung der Architektur:

Blackboard

- zentraler Datenspeicher für Teillösungen und Kontrolldaten (Vokabular)
- stellt Schnittstelle bereit zum Lesen und Schreiben ans schwarze Brett
- Beiträge auf dem Brett heißen Hypothesen
- Hypothesen haben Metadaten, wie Abstraktionsgrad, Zuverlässigkeit, Abdeckungsbereich, Beziehungen zu anderen Hypothesen

Verfeinerung der Architektur:

Wissensquelle

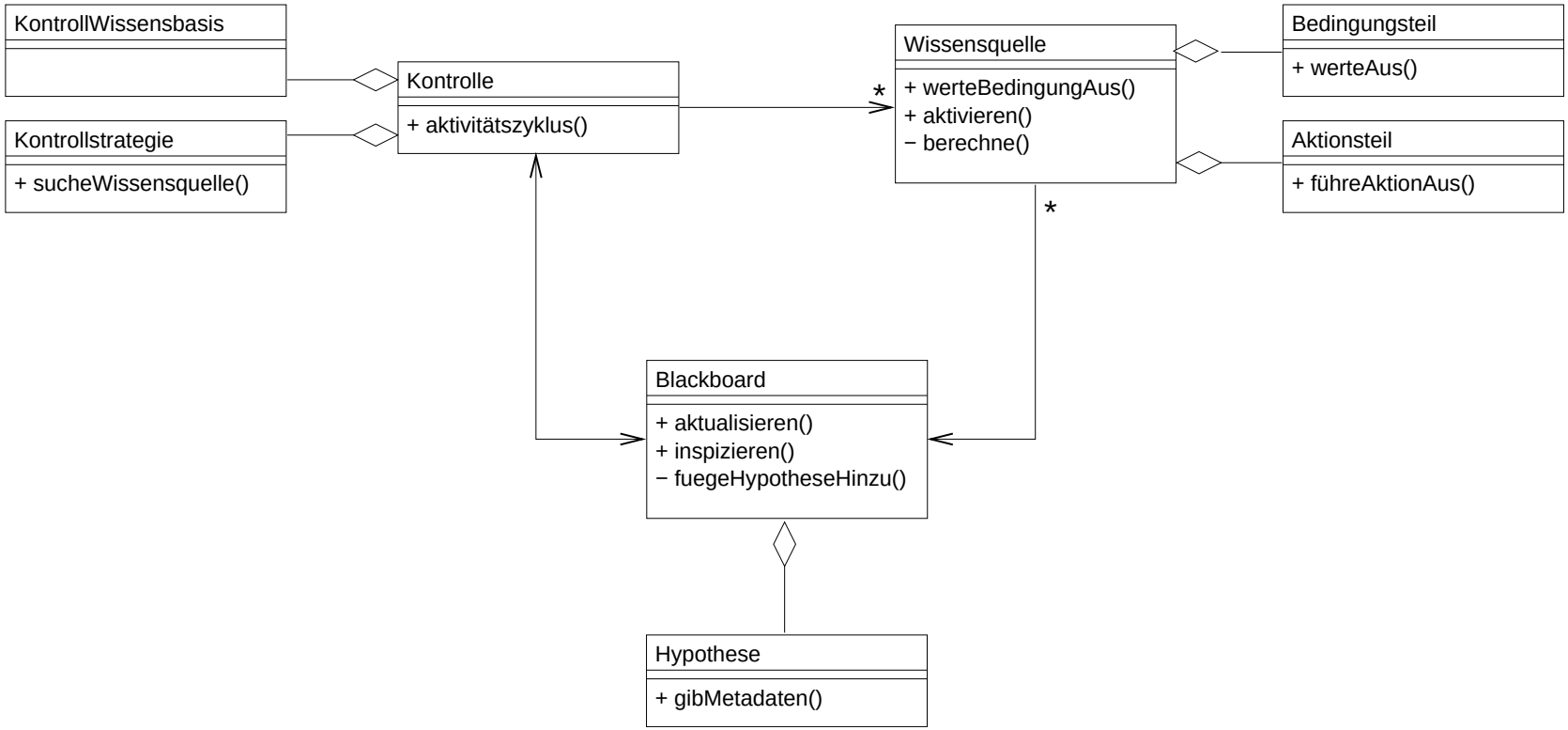
- unabhängiges Subsystem, das speziellen Aspekt, aber nicht gesamtes Problem lösen kann
- interagieren nicht direkt, sondern ausschließlich über das Blackboard
- müssen das Vokabular des Blackboards verstehen (Standardformat, Ontologien)
- signalisieren, unter welchen Bedingungen sie zu einer (Teil–)Lösung beitragen können = Bedingungsteil
- erzeugen ein Ergebnis, das den Inhalt des Blackboards verändert = Aktionsteil

Verfeinerung der Architektur:

Kontrolle

- durchläuft einen Arbeitszyklus, in dem Änderungen des Blackboards beobachtet werden
- entscheidet, wie der Problemlösungsprozess fortgesetzt wird auf Basis von
  - Inhalt des Blackboards
  - Strategie / Heuristik zur Auswahl geeigneter Wissensquellen
  - Kontrollwissensbasis, die Kontrolldaten auf Blackboard schreibt

Verfeinerung der Architektur:



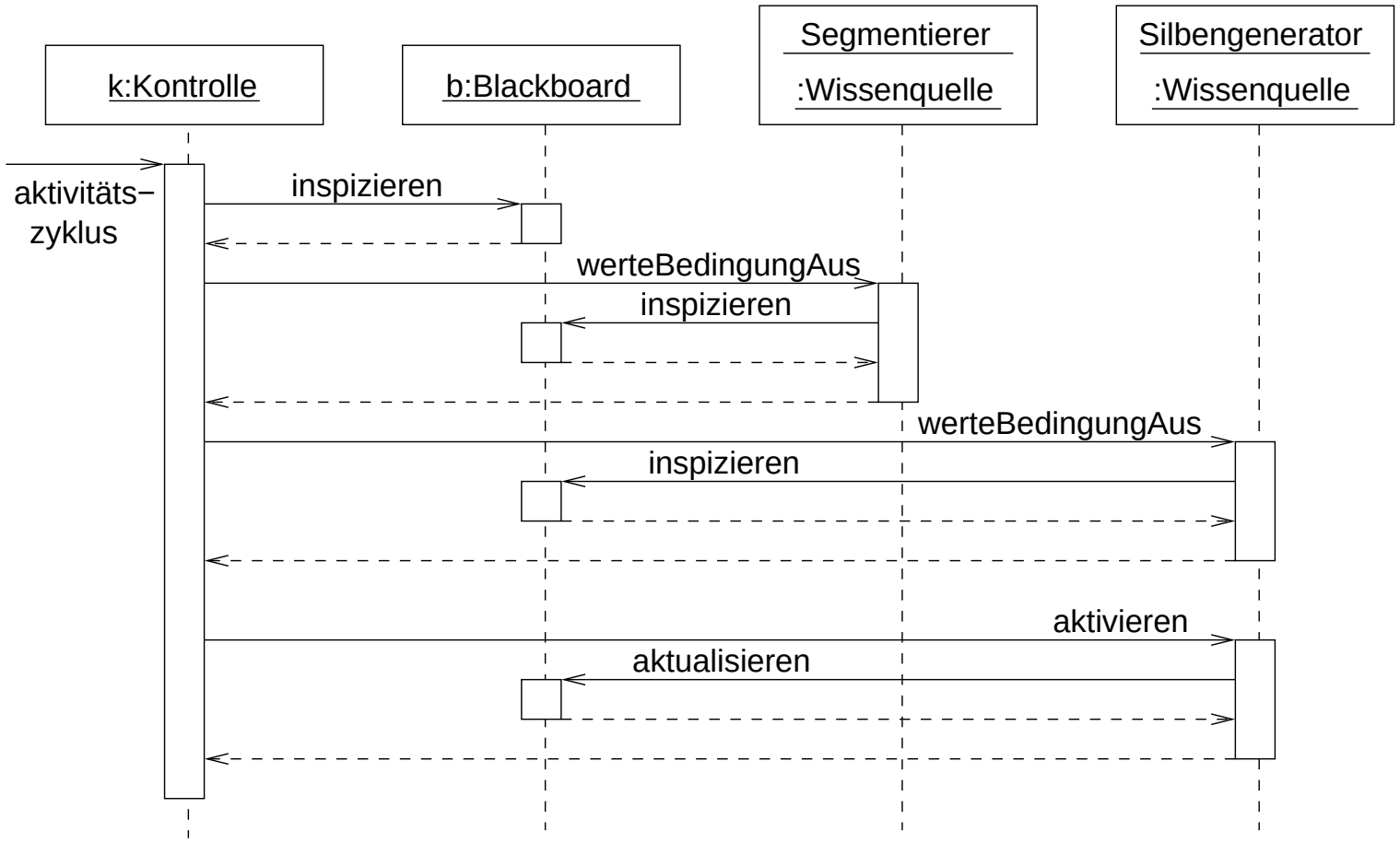
Beispiel:

System zur Spracherkennung (HEARSAY–II)

Hat Komponenten zur

- Segmentation von phonetischen Einheiten
- Silbenerzeugung
- Worterzeugung
- Phrasenerzeugung

Interaktion in der Blackboard–Architektur:



Implementierung:

- Problem definieren
- Lösungsraum definieren (Abstraktionsstufen, Teilprobleme)
- Vokabular definieren
- Kontroll–Komponente implementieren (Heuristiken)
- Wissensquellen implementieren oder ggf. existierende integrieren (Adapter– und WrapperFacade–Muster)

Varianten:

- Blackboard–Architektur als Verallgemeinerung von
  Produktionssystemen: Regeln aus Bedingungs– und Aktionsteil durch Konfliktauflösungsmodul gesteuert
- Vereinfachung der Blackboard–Architektur:
  Repositorium–Architektur
  - Repositorium ist zentrale Datenstruktur: Blackboard
  - keine Kontrollkomponente
  - Anwendungsprogramme, die Repositorium manipulieren: Wissensquellen

Gegenanzeigen:

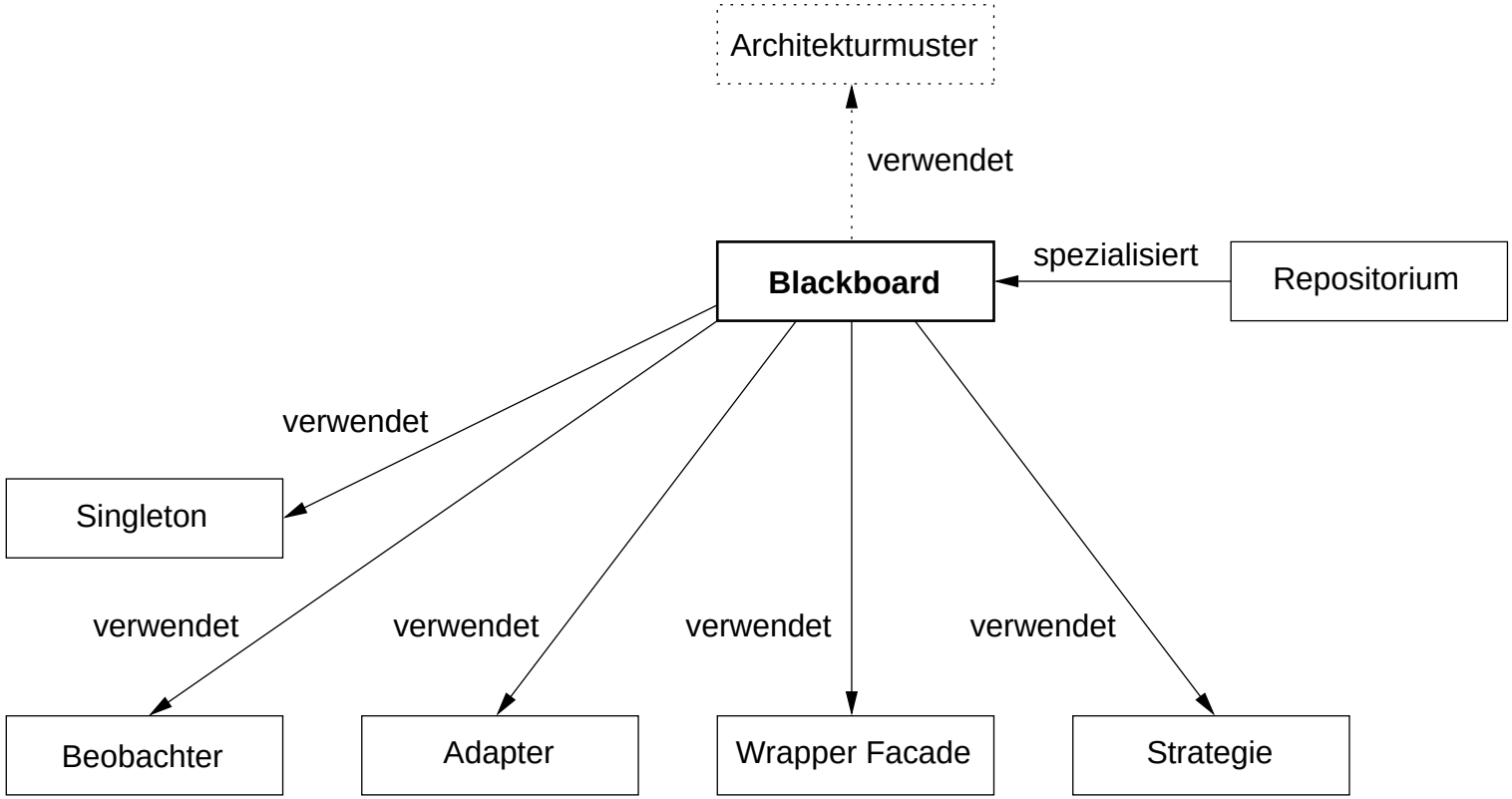
- Ergebnisse sind zum Teil nicht reproduzierbar und damit schwer zu testen
- Lösung nicht garantiert, Güte der Lösung ebenfalls nicht
- Geringe Effizienz da falsche Hypothesen verfolgt werden können
- Entwicklungsaufwand hoch: Vokabular definieren, Heuristiken und Kontrollstrategien finden

Falls für Problemgebiet geschlossener Lösungsalgorithmus entwickelt wird, ist ein darauf basierendes System einem Blackboard–System vorzuziehen. Für Experimentieren in schlecht erschlossenen Domänen ist Blackboard geeignet.

Klassifizierung des Blackboard–Musters:

- **Art:** Software–Muster: Architekturmuster
- **Einsatzgebiet:** schlecht strukturierte Gebiete
- **Konzept:** Interaktionsregelung, Dekomposition
- **Komplexität:** einfach, einige Diskussionspunkte, variantenarm

Beziehungen zu anderen Mustern:



Zusammenfassung:

- Software–Architektur
- Architekturmuster – Allgemeines
- Architekturmuster zur Strukturierung von Systemen
  - Schichten–Architektur
  - Blackboard–Architektur

Literatur zur heutigen Vorlesung:

- Buschmann, F., R. Meunier, H. Rohnert, P. Sommerlad, M. Stal: *A System of Patterns*, Wiley & Sons. 1996, Kapitel 2.2
- Tanenbaum, A.: Computer–Netzwerke, Wolfram's Verlag. 1992, Kapitel 1