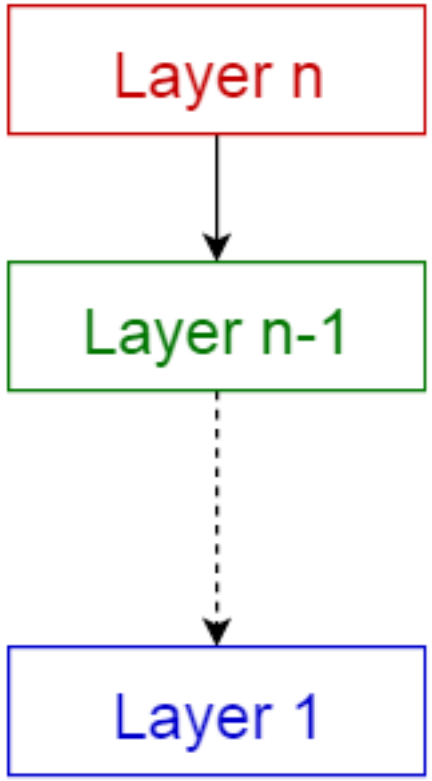
Ausgewählte Architekturpatterns

Pattern	Beschreibung
Layered Pattern	Strukturierung eines Programms in Schichten
Client-Server Pattern	Ein Server stellt Services für mehrere Clients zur Verfügung
Master-Slave Pattern	Ein Master verteilt die Arbeit auf mehrere Slaves
Pipe-Filter Pattern	Verarbeitung eines Datenstroms (filtern, zuordnen, speichern)
Broker Pattern	Meldungsvermittler zwischen verschiedenen Endpunkten
Event-Bus Pattern	Datenquellen publizieren Meldungen an einen Kanal auf dem Event-Bus. Datensenken abonnieren einen bestimmten Kanal
MVC Pattern	Eine interaktive Anwendung wird in 3 Komponenten aufgeteilt: Model, View – Informationsanzeige, Controller – Verarbeitung der Benutzereingabe

Schichtenkonzept (Layered Pattern) (1/3)

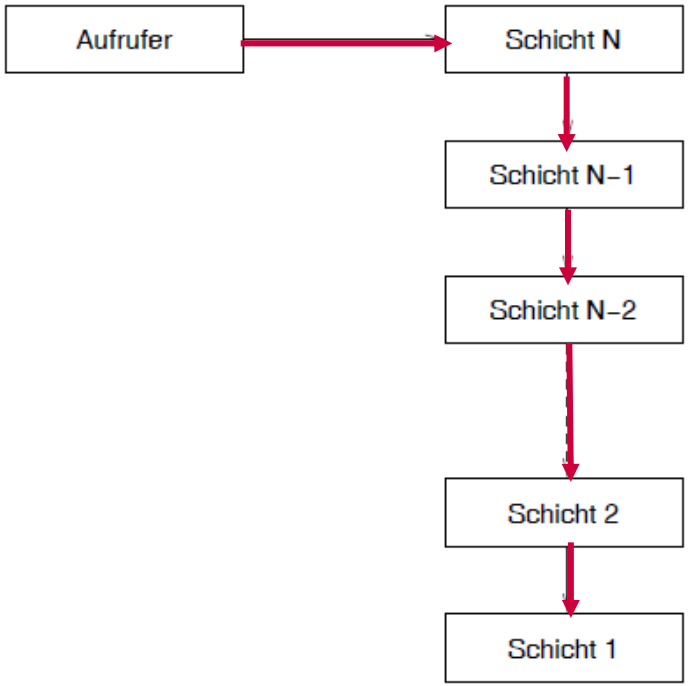
- Zerlegung des Gesamtsystems in Schichten
- Je weiter unten, desto allgemeiner
- Je höher, desto anwendungs-spezifischer
- Zuoberst ist das Benutzerinterface
- Kopplung nur von oben nach unten, NIE von unten nach oben



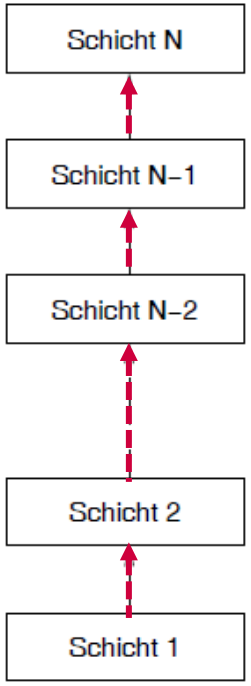
Schichtenkonzept (Layered Pattern) (2/3)

- Aufrufszenarien

höherer Schichten rufen Funktionalität in unteren Schichten direkt auf



untere Schicht benachrichtigt obere Schicht über Ereignis (Observer)

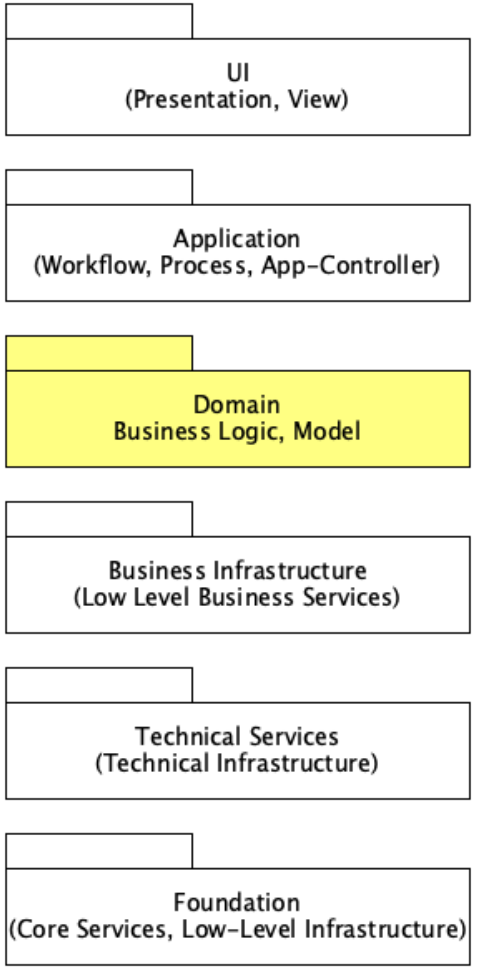


Kein direkter Aufruf!

Schichtenkonzept (Layered Pattern) (3/3)
Beispiel Enterprise Architektur

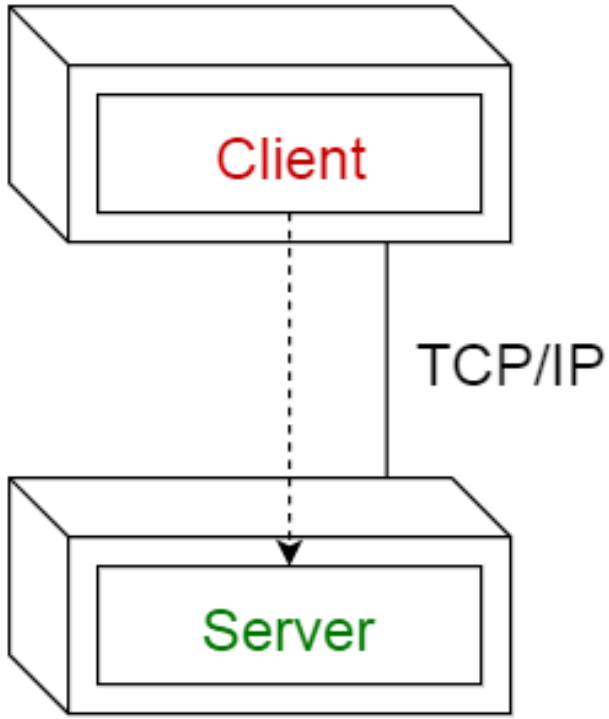
Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- UI
  - Presentation, Windows, Dialoge, Reports, WEB, Mobile
- Application
  - behandelt Requests von UI Layer, Workflow, Sessions
- Domain
  - behandelt Requests von Application Layer, Domain Rules und Services
- Business Infrastructure
  - Low Level Business Services, wie z.B. CurrencyConverter
- Technical Services
  - Persistence, Security, Logging
- Foundation
  - Datenstrukturen, Threads, Dateien, Network IO



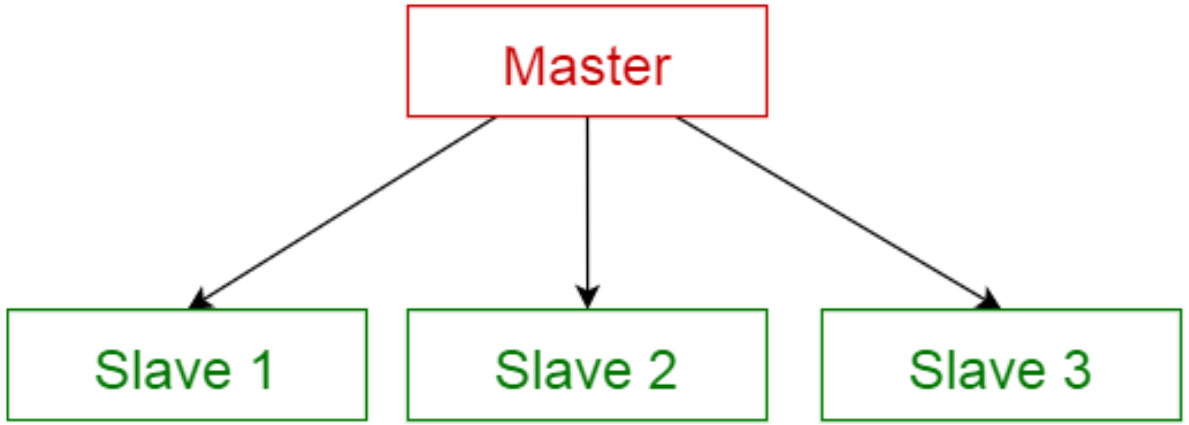
Client-Server

- Ein Server und mehrere Clients
- Ein Server stellt einen oder mehrere Services zur Verfügung
- Der Client macht eine Anfrage (Request) zum Server
- Der Server sendet eine Antwort (Response) zurück



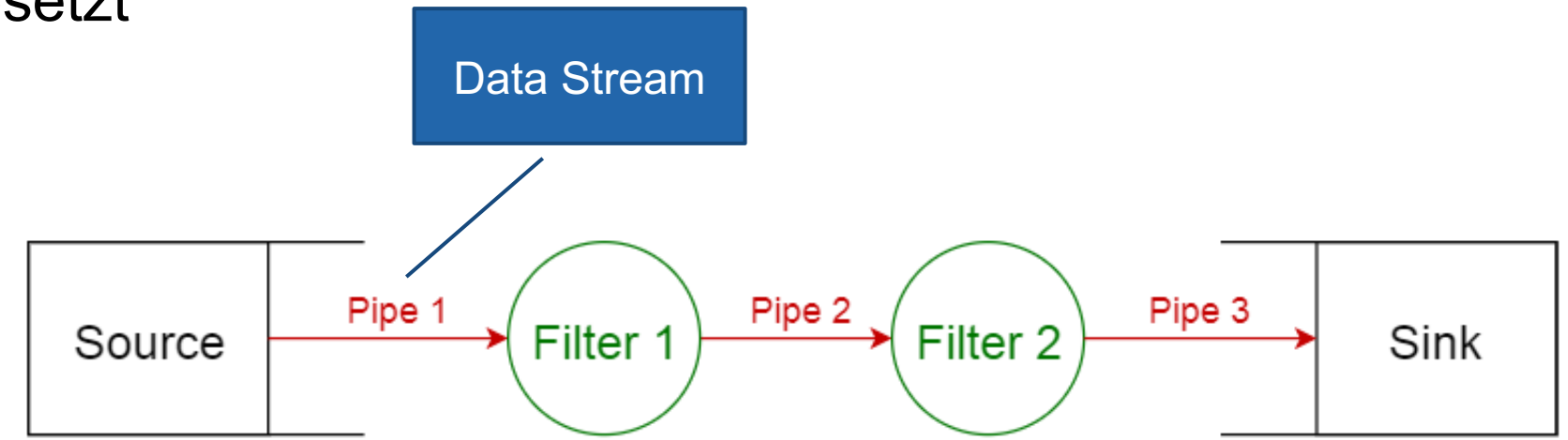
Master-Slave Pattern

- Der Master verteilt die Aufgaben auf mehrere Slaves
- Die Slaves führen die Berechnung aus und senden das Ergebnis zum Master
- Der Master berechnet das Endergebnis



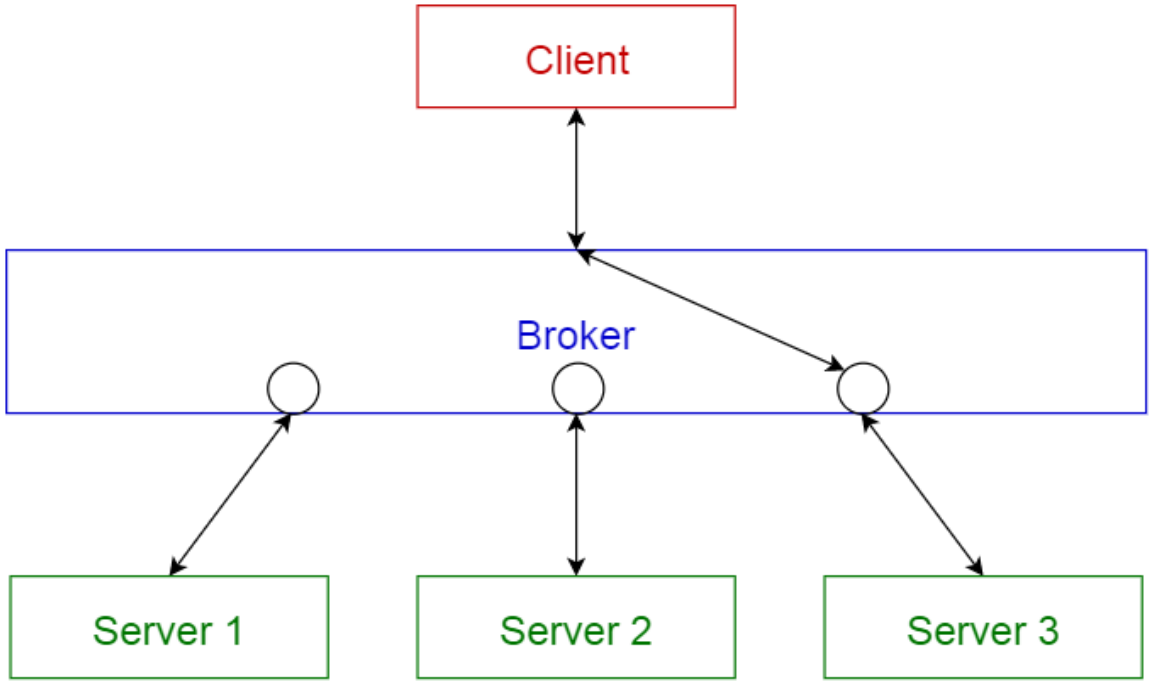
Pipe-Filter-Pattern

- Das Pattern kommt bei der Verarbeitung von Datenströmen zum Einsatz (Linux Pipe, RxJS Observable Streams, Java Streams, …)
- Jeder Verarbeitungsschritt wird durch einen Operator wie Filter, Mapper, etc. umgesetzt



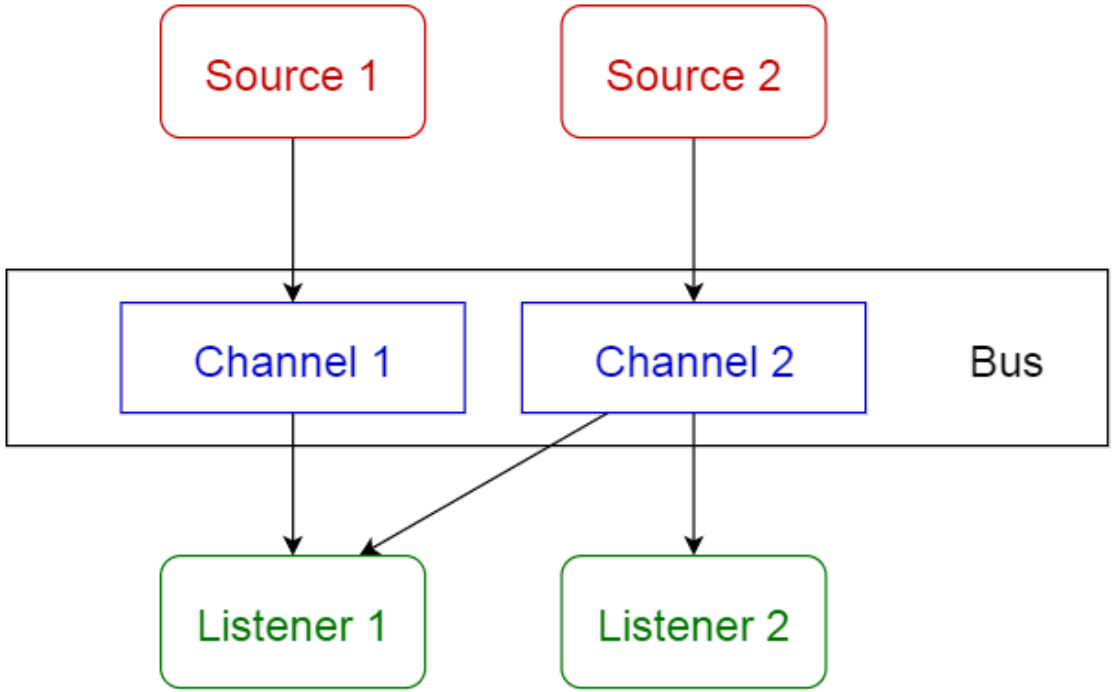
Broker Pattern

- Das Pattern wird eingesetzt, um verteilte Systeme mit entkoppelten Subsystemen zu koordinieren.
- Der Broker (Vermittler) vermittelt die Kommunikation zwischen einem Client und dem entsprechenden Subsystem
- Bsp.: Message Broker



Event-Bus-Pattern

- Der Pattern umfasst vier Hauptkomponenten: EventSource, EventListener, Channel und Event Bus.
- Die Event Sources publizieren Meldungen zu einem bestimmten Kanal auf dem Event Bus
- EventListeners
  - Melden sich für bestimmte Events an
  - werden informiert, sobald sich entsprechende Meldungen auf dem Kanal befinden



Model View Controller Pattern

- Eine interaktive Anwendung wird in drei Komponenten aufgeteilt:
  - **Model**: Daten und Logik,
  - **View**: Informationsanzeige
  - **Controller**: Verarbeitung der Benutzereingabe
- Bewirkt eine Entkopplung von UI und Logik
- Erlaubt Austauschbarkeit des Uis
- Alternativen
  - MVVM: Model View View Model
  - MVP: Model View Presenter
  - (mehr dazu in SWEN1-Vertiefung GUI-Architekturen)