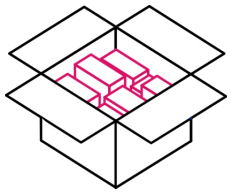
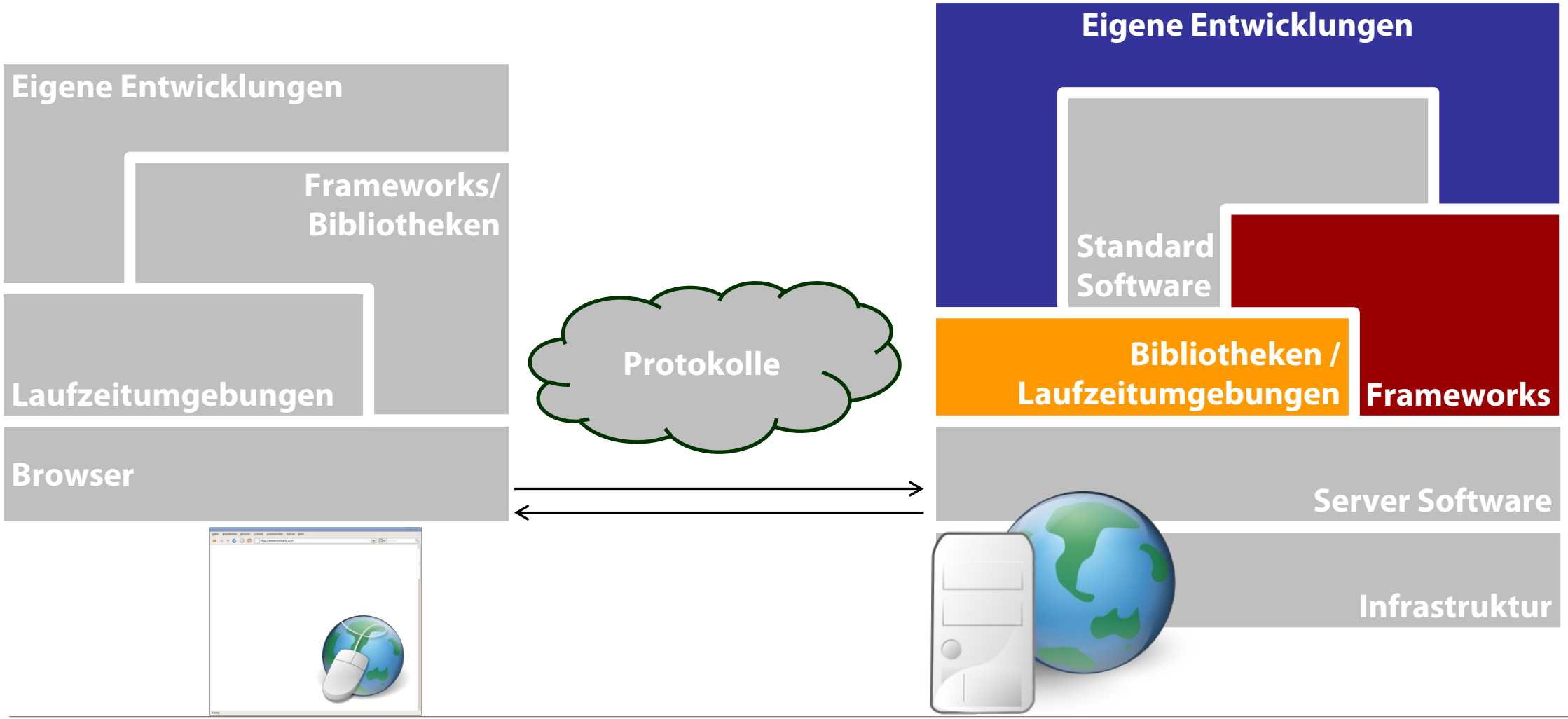
Lose Kopplung in Softwaresystemen

Hoai Viet Nguyen – TH Köln

Technology
Arts Sciences
TH Köln



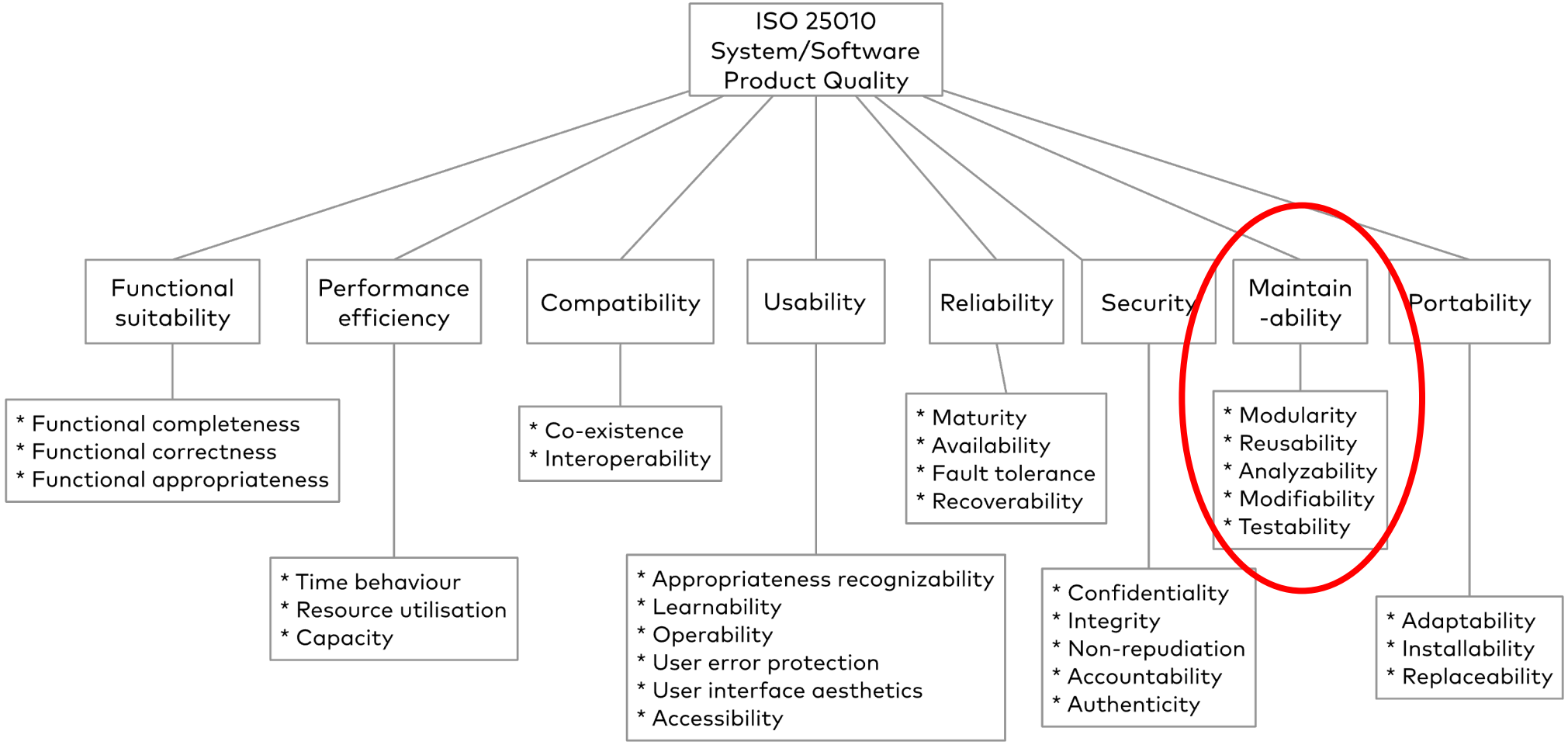
Wo befinden wir uns gerade?



Recap letztes Video

- Domain Driven Design:
  - Modellierung komplexer Software Systeme
  - Aufteilung von Software in Domänen bzw. Fachlichkeiten
  - Bounded Context definiert Building Blocks wie z.B.
    - Entities
    - Value Objects
    - Repositories
- Implementierung Repository für Speicherung von Daten in DB

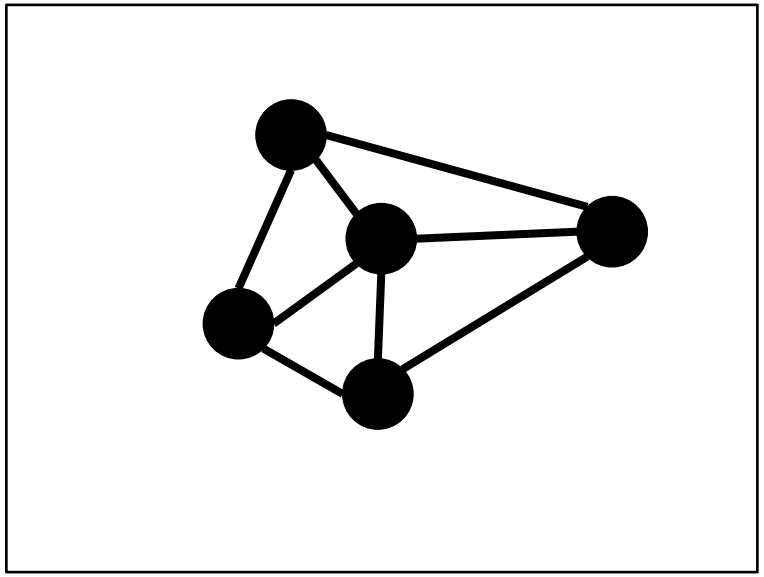
Softwarequalität



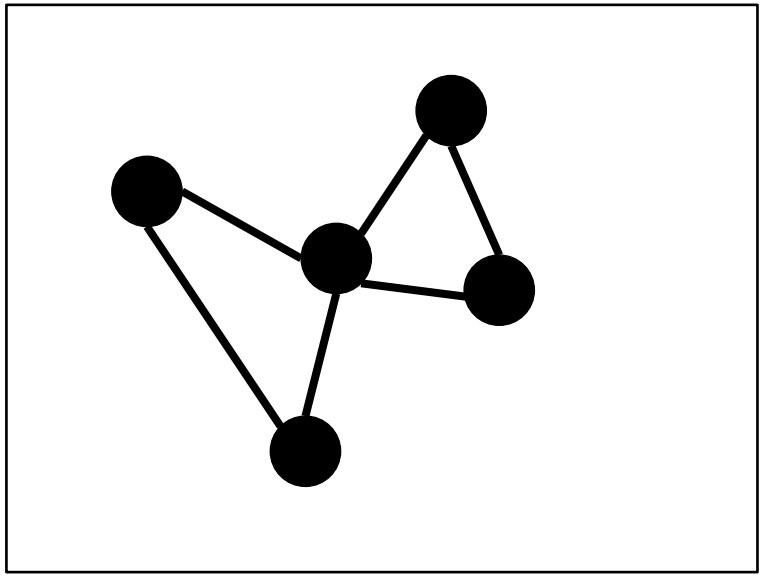
Quelle: https://www.innoq.com/en/articles/2023/02/iso-25010-shortcomings/

Kopplung und Kohäsion

- **Kopplung:** Abhängigkeit von Modulen untereinander
- **Kohäsion:** Zusammengehörigkeit von Klassen innerhalb eines Moduls



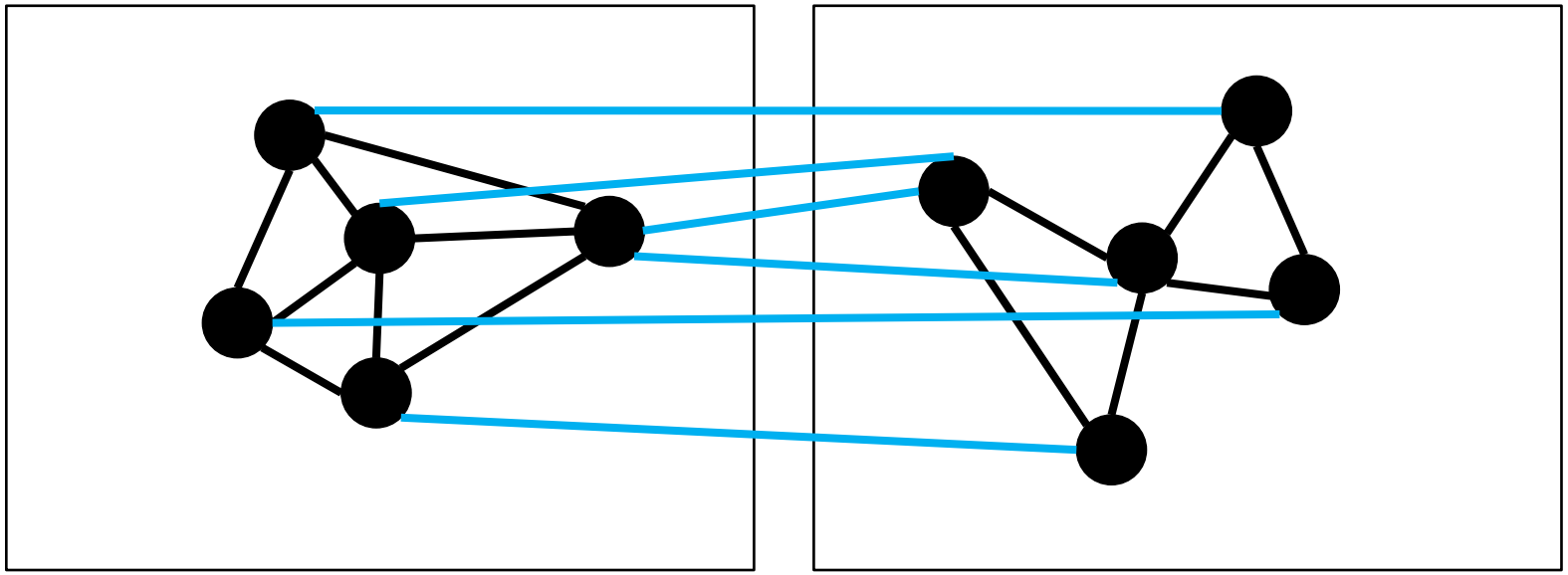
Modul A



Modul B

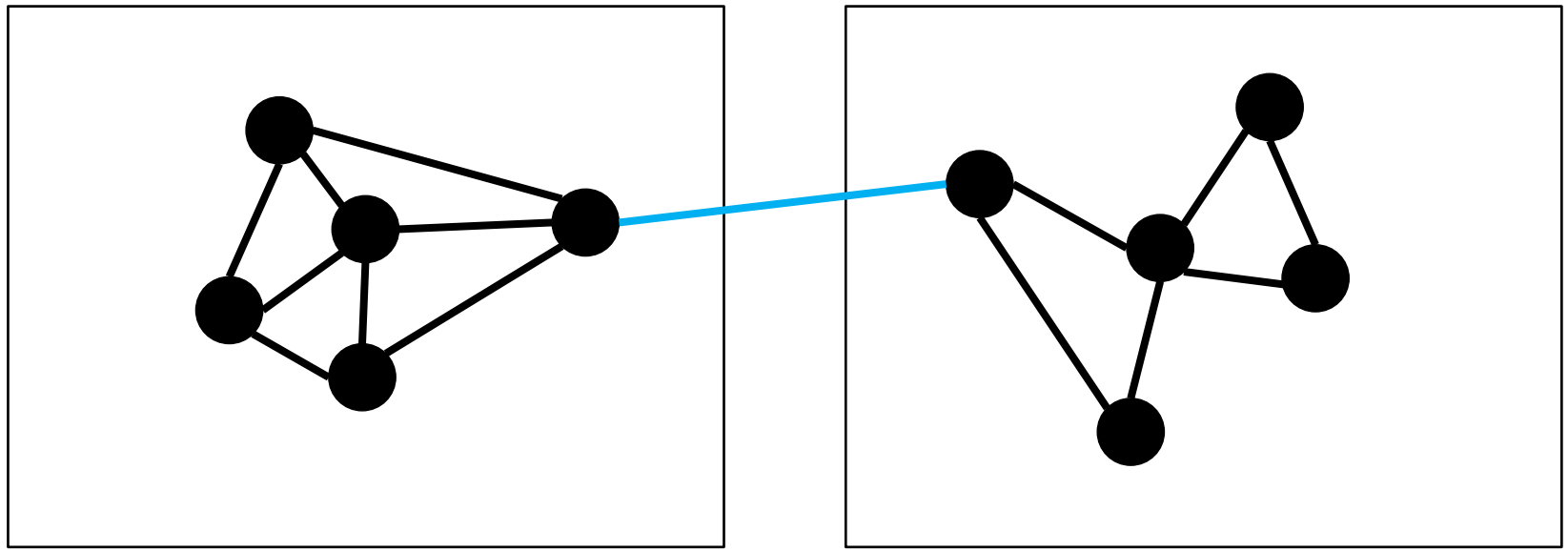
Enge Kopplung in der Softwarearchitektur

- Hohe Abhängigkeiten der Klassen zwischen Modulen untereinander
- Austausch oder Anpassung von Modul verursacht viele Änderungen des anderen Moduls



Lose Kopplung und hohe Kohäsion in der Softwarearchitektur

- **Lose Kopplung:** Geringe Abhängigkeiten der Klassen zwischen Modulen untereinander
- **Hohe Kohäsion:** Hohe Zusammengehörigkeit von Klassen innerhalb eines Moduls



SOLID Principles [OOD]

- **S**ingle Responsibility Principle (**SRP**)
- **O**pen-Close (**OCP**)
- **L**iskov Substitution Principle (**LSP**)
- **I**nterface Segregation Principle (**ISP**)
- **D**ependency Inversion Principle (**DIP**)
- **Ziel:**
  - **Wartbarkeit**
  - **Portierbarkeit**
  - **Lose Kopplung**
  - **Hohe Kohäsion**

[OOD] Robert C. Martin, Principles of OOD, http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

Single Responsiblity Principle (SRP) [OOD] [Bente] [Starke]

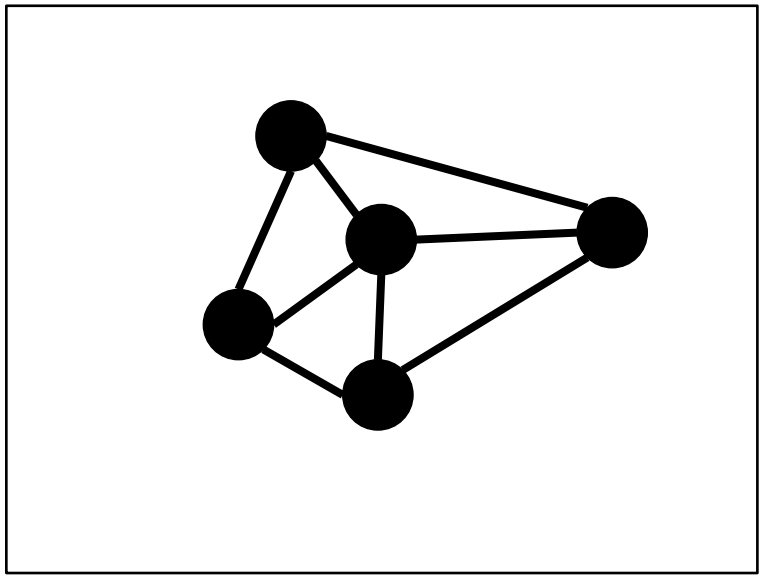
- Auch bekannt als Seperation of Concerns
- Trenne Verantwortlichkeiten
- Jedes Modul / Paket / Klasse / … ist genau für einen Zweck zuständig

[OOD] Robert C. Martin, Principles of OOD, http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod

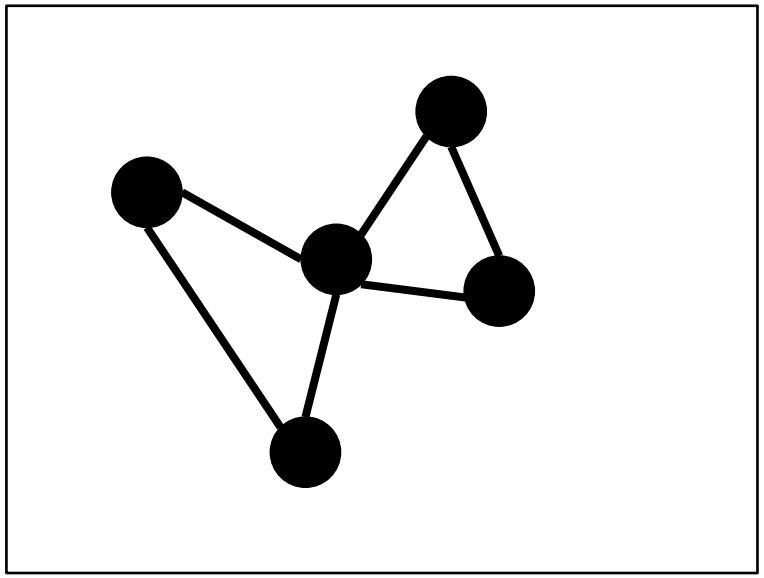
[Bente] Stefan Bente, Vorlesungskript Softwaretechnik 2, https://ilias.th-koeln.de/goto.php?target=file_1854728_download&client_id=ILIAS_FH_Koeln

[Starke] Gernot Starke, Effektive Softwarearchitekturen, Carl Hanser Verlag, 2020

SRP in der Architektur

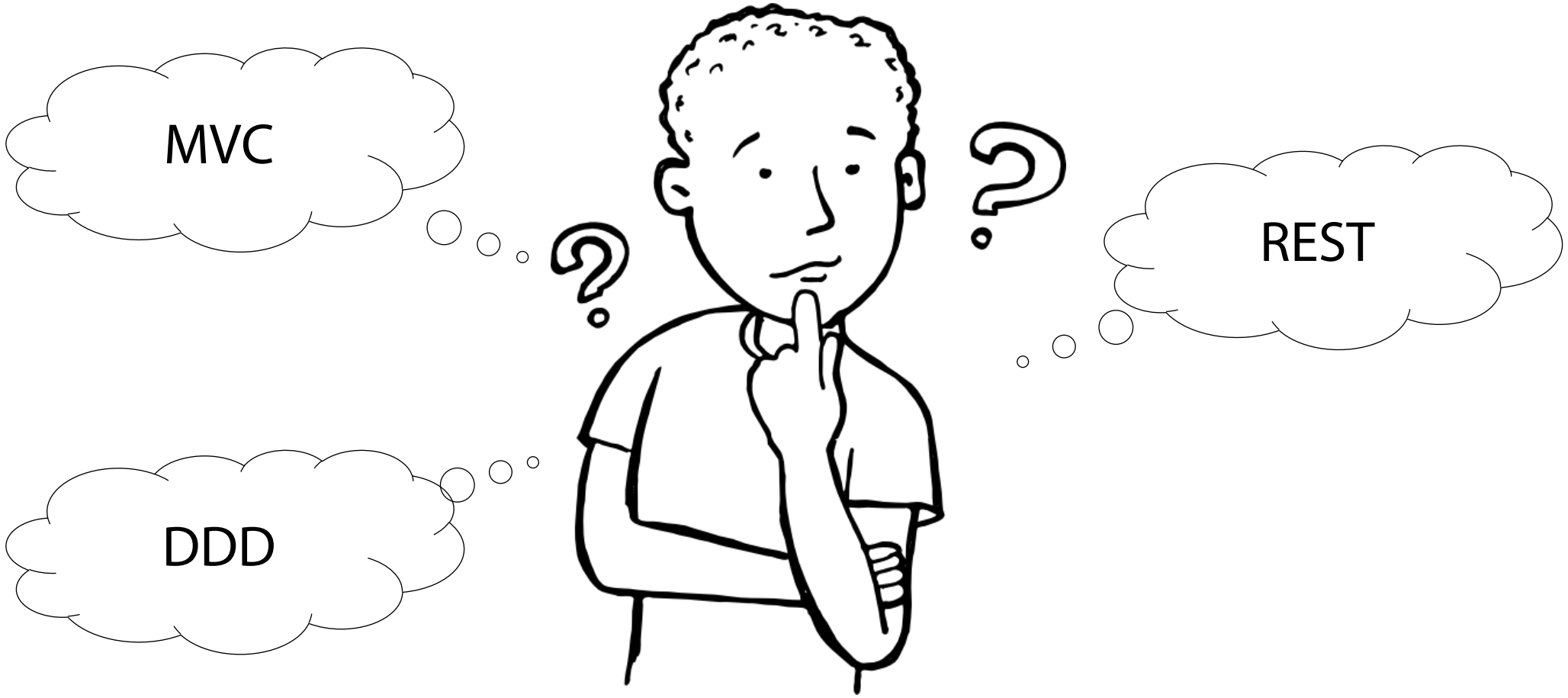


Modul A
(Verantwortlichkeit A)

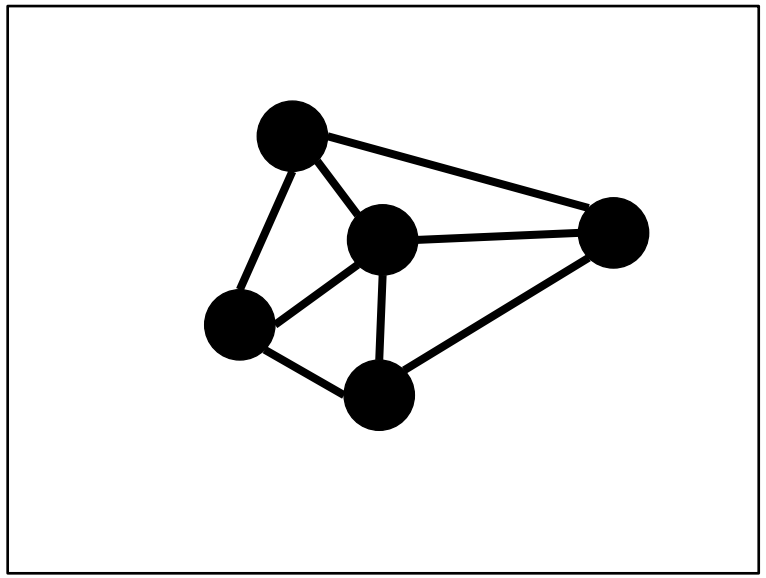


Modul B
(Verantwortlichkeit B)

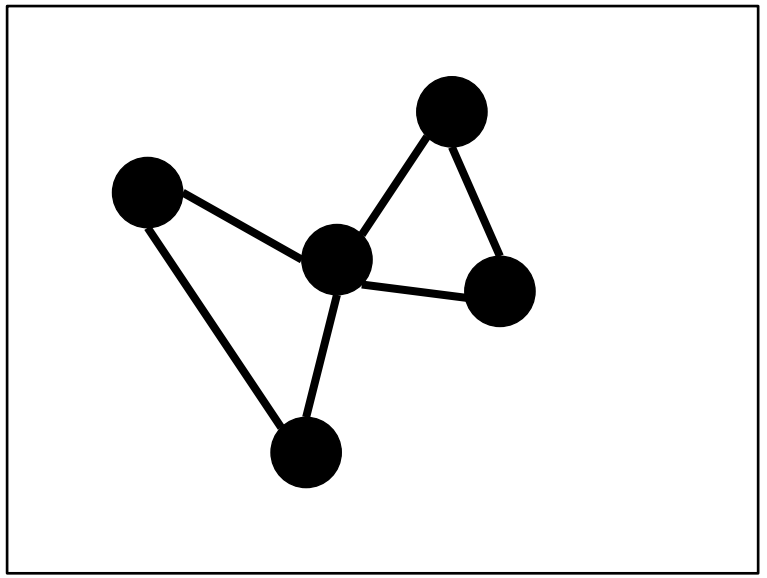
Welche Konzepte kennen Sie, die SRP anstreben?



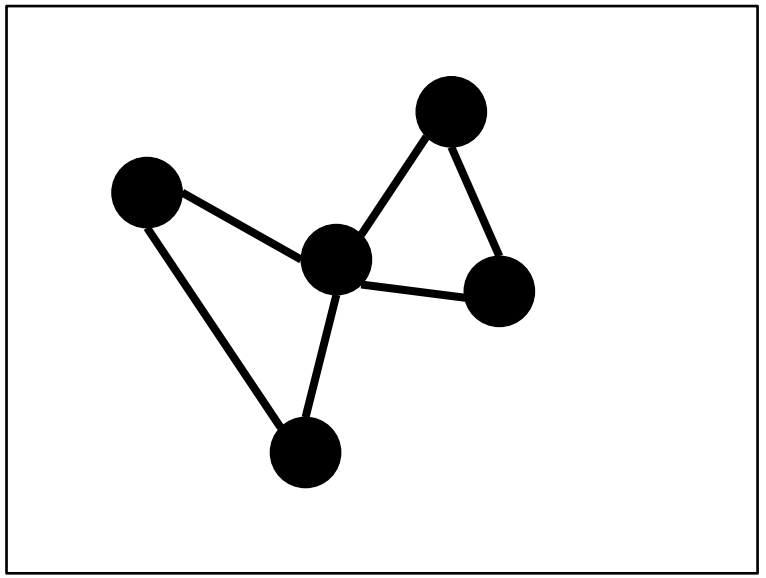
SRP in MVC



Model
(Verantwortlichkeit
Datenverwaltung)



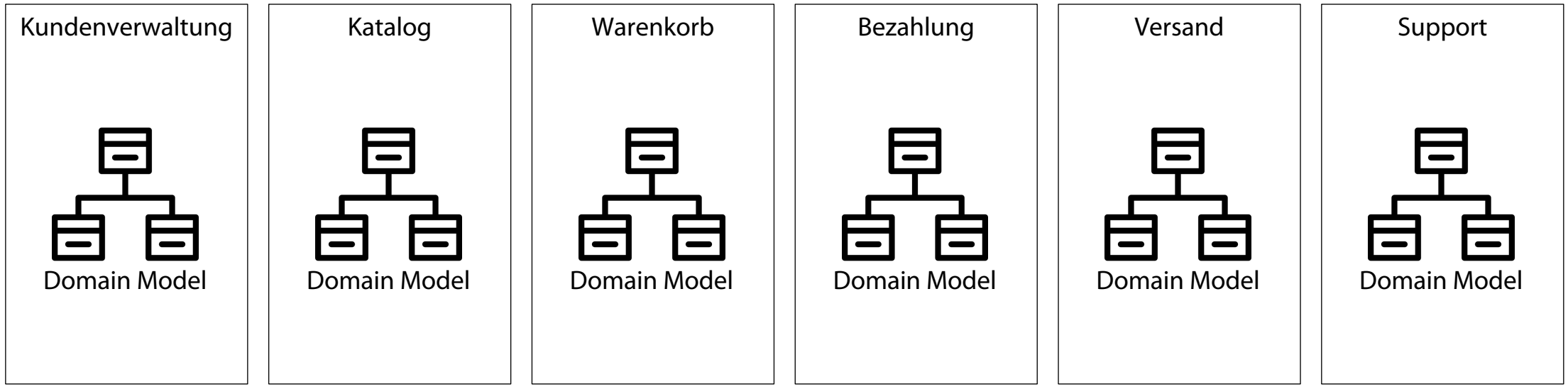
View
(Verantwortlichkeit Darstellung)



Controller
(Verantwortlichkeit
Anwendungslogik)

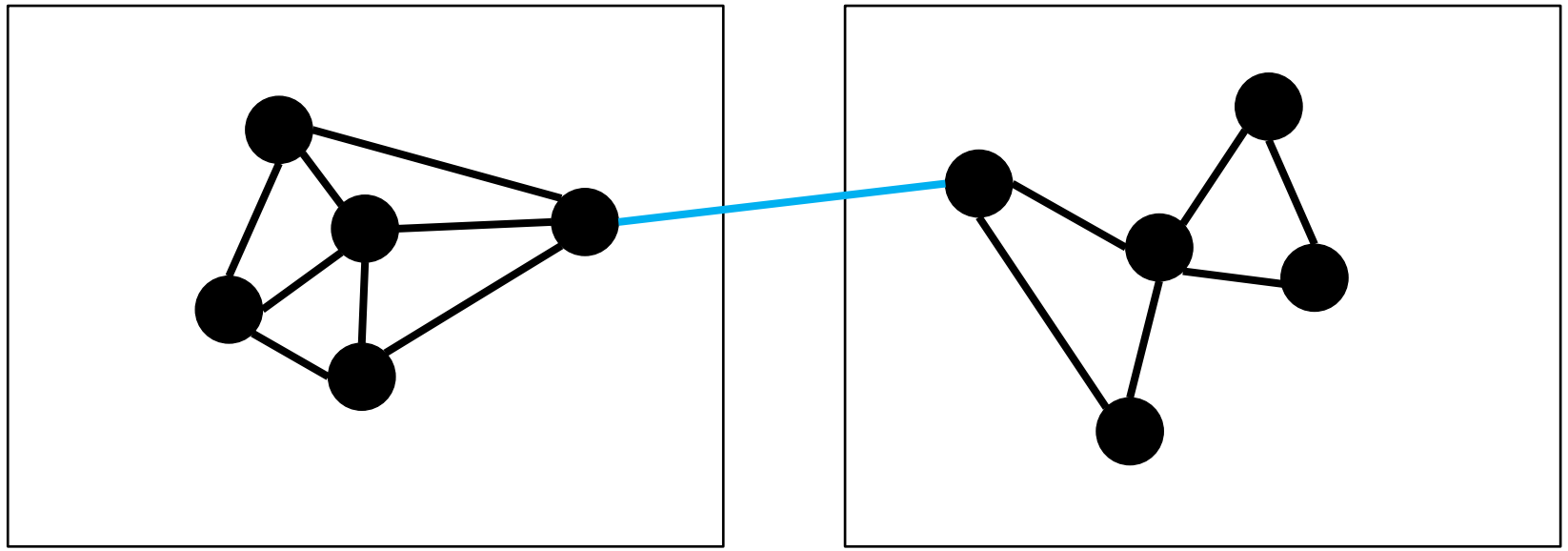
SRP in DDD anhand eines Beispiels „großer“ Onlineshop

Bounded Contexts



Open-Closed Principle (OCP)

- Offen für Erweiterungen, aber geschlossen für Änderungen
- Erweiterungen in einem Modul sollte ohne die Änderungen des anderen Moduls möglich sein



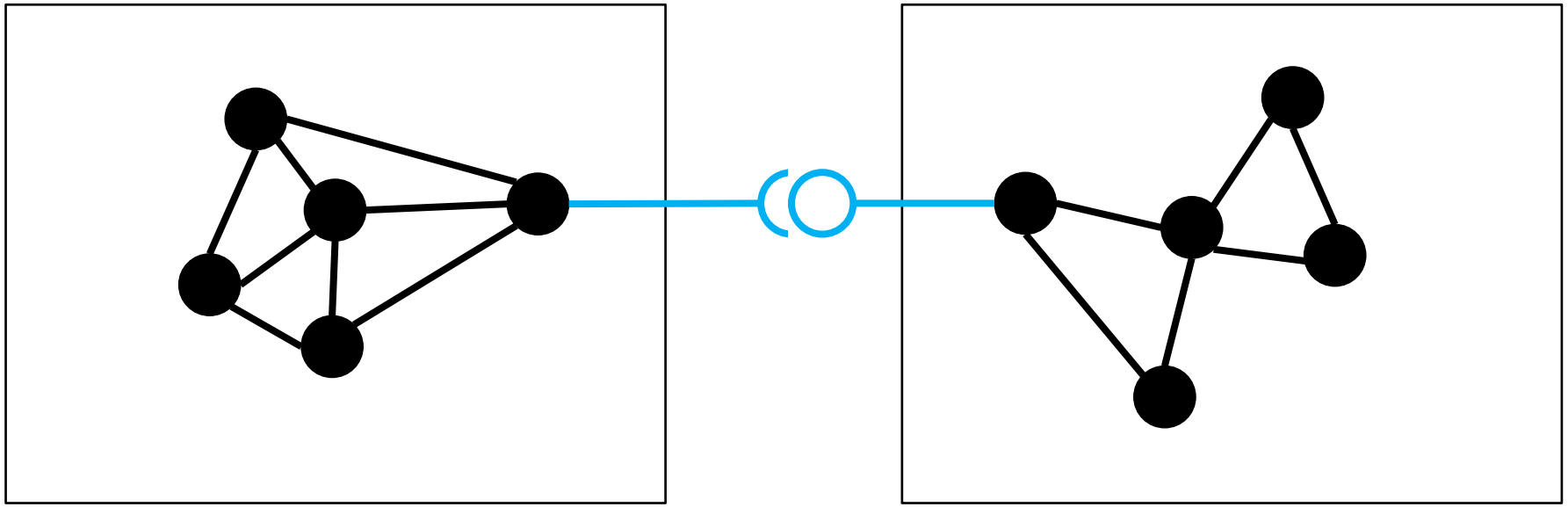
Dependency Inversion Principle (DIP)

- Abhängigkeit über Abstraktionen und Interfaces, nicht von konkreten Implementierungen
- Ziele DIP:
  - Optimierung der losen Kopplung
  - Austausch von Modulen ohne die Beeinflussung von abhängigen Modulen

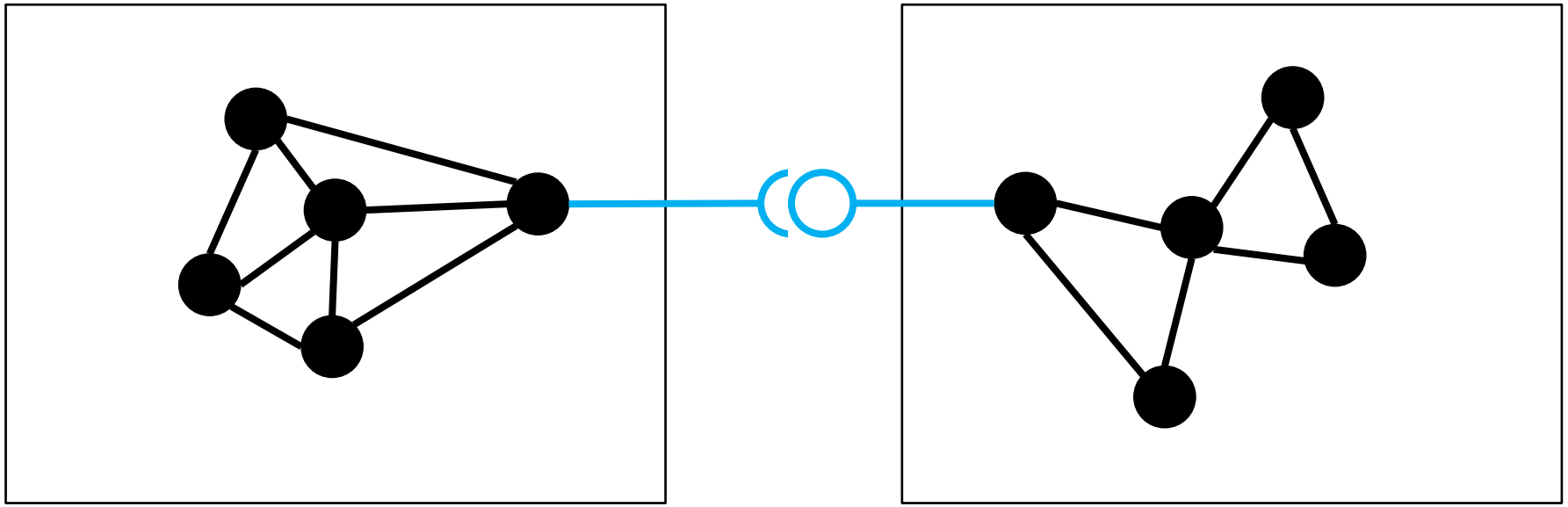
SRP, OCP und DIP am Beispiel Steckdose

Quelle: https://www.pexels.com

Lose Kopplung durch Interfaces



Lose Kopplung durch Interfaces am Beispiel Anwendung und Datenbank



Warum macht es Sinn, dass Komponenten über Interfaces miteinander kommunizieren?

Lose Kopplung am Beispiel DB-Konfiguration in application.properties

```
spring.datasource.url=jdbc:h2:mem:todo
spring.datasource.username=sa
spring.datasource.password=password
…
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/todo
spring.datasource.username=sa
spring.datasource.password=password
…
```

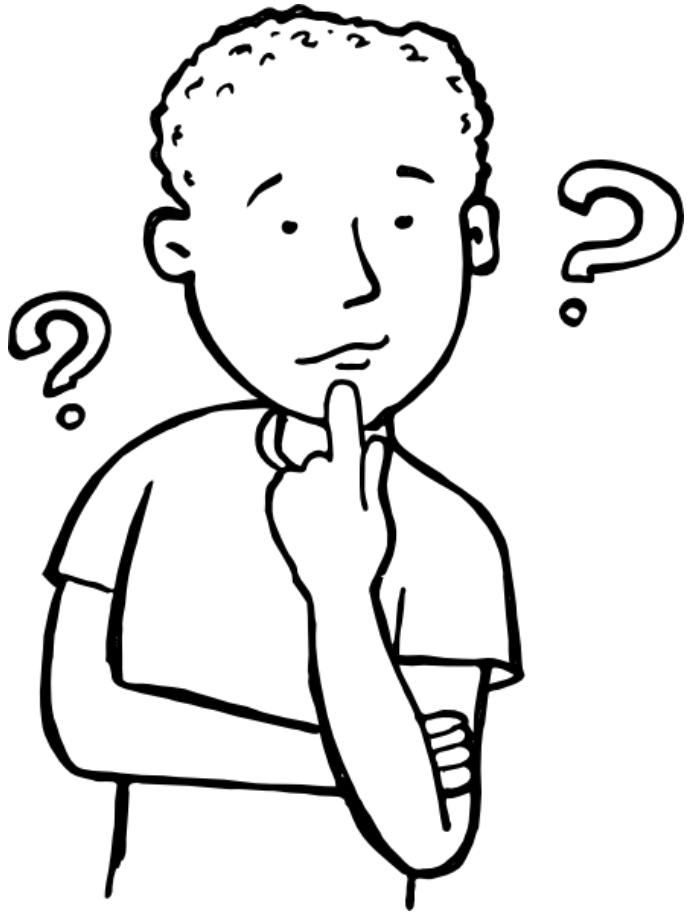
```
spring.datasource.url=jdbc:postgresql://localhost:5432/todo
spring.datasource.username=sa
spring.datasource.password=password
…
```

Aufteilung Bounded Context in Building Blocks

- **Entities** (Objekte mit ID, identifizieren sich über ihre ID)
- **Value Objects** (Objekte ohne ID, identifizieren sich über ihren Wert)
- **Aggregates** (Strukturen aus Entities und Value Objects)
- **Services** (Geschäftslogik)
- **Repositories** (Ermöglicht Zugriff auf Entities und Value Objects)
- **Factories** (Ermöglicht das Erstellen eines neuen Aggregates)

Live-Coding: Integration von Services in Spring

Was ist der Unterschied zwischen Controller und Service in Spring? Beide sind doch verantwortlich für die Geschäftslogik?



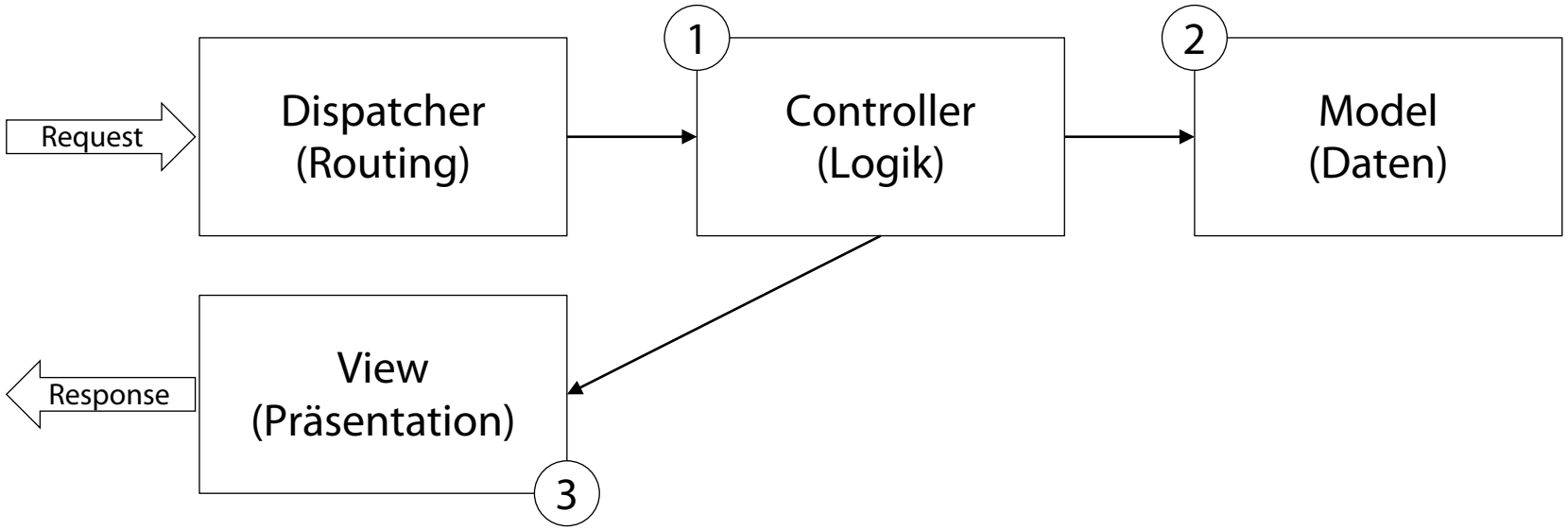
Controller vs Service [Cleary]

- Beide Elemente sind für Geschäftslogik zuständig
- Controller
  - Vorbehandlung eingehende Requests
  - Teilt die Arbeit passenden Einheiten auf
  - Entscheidet welcher Service welche Arbeit machen soll
  - Orchestriert die Arbeit an mehrere Services
  - Entscheidet welche Antwort, er dem Client zurückgeben soll
- Service
  - Nimmt Aufträge vom Controller an
  - Trift keine „großen“ Entscheidungen
  - Erledigt die erforderlichen Aufgaben, damit die Arbeit erledigt ist

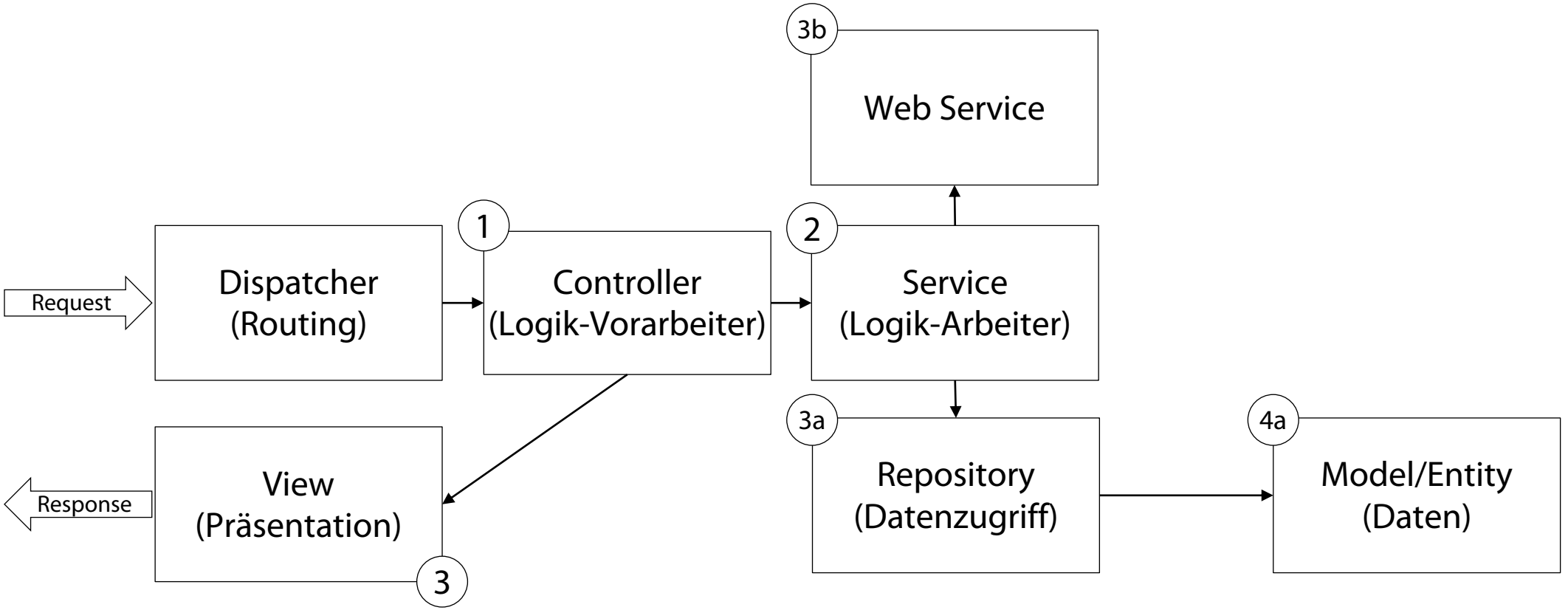
[Cleary] https://www.coreycleary.me/what-is-the-difference-between-controllers-and-services-in-node-rest-apis#

Model-View-Controller Architekturmuster

- **Trennung von Logik (Programmierer), Daten (Datenmanager) und Präsentation (Designer)**



MVC-Umsetzung in Spring



Beispiel: Controller-Funktion mit Aufruf zwei Services

```
@Controller
class UsersController(private val usersService: UsersService, private  val emailService: EmailService) {

    @PostMapping("/users")
    @ResponseStatus(HttpStatus.CREATED)
    fun saveUser(email: String, password: String, passwordConfirm: String){
        if(password == passwordConfirm){
            usersService.save(email, password)
            emailService.sendRegistrationNotification(email)
        } else {
            throw ResponseStatusException(HttpStatus.BAD_REQUEST)
        }

    }
}
```

Vorbehandlung eingehender Request

Orchestrierung der Geschäftslogik

Diese Fragen sollten Sie nach der heutigen Sitzung beantworten können

- Was ist lose Kopplung?
- Was ist SRP, OCP und DIP?
- Warum macht es Sinn über Interfaces zu kommunizieren?
- Was ist der Unterschied zwischen Controller und Service?
- Wie implementiere ich ein Service in Spring?

Zusammenfassung

- **Lose Kopplung:** Geringer Grad der Abhängigkeit zwischen Komponenten untereinander
- **SRP, OCP und DIP sind Prinzipien die lose Kopplung anstreben durch:**
  - **Trennung von Verantwortlichkeiten**
  - **Offen für Erweiterungen, aber geschlossen für Änderungen**
  - **Abhängigkeiten zwischen Modulen durch Interfaces**
- **Einhaltung der SOLID-Prinzipien ermöglicht höhere u.a. Wartbarkeit**
- **Implementierungen können sich über die Zeit ändern, Interfaces sollen gleich bleiben**

Danksagung

Die verwendeten Icons stehen unter der Flaticon Basic Lizenz (flaticon.com) und stammen von

- Freepik
- Atif
- juicy fish
- BZZRINCANTATION