Entwurfsprinzipien

Stand April 2020

http://arc42.de

Ziele und Inhalt

Sie lernen:

- Kopplung
- Kohäsion
- Konsistenz
- SOLID-Prinzipien
- Entwurfsmuster

Lernziele gemäß iSAQB CPSA-F:

- **LZ 2-6: Entwurfsprinzipien erläutern und anwenden (R1)**
- **LZ 2-7: Abhängigkeiten von Bausteinen planen (R1)**

Modularität

modularity — *The degree to which a system or computer program is composed of discrete components such that a change to one component has minimal impact on other components.*

IEEE Std 610.12 (1990)

modular decomposition — *The process of breaking a system into components to facilitate design and development*

IEEE Std 610.12 (1990)

Vorteile hoher Modularität:

- Implementierung von Modulen austauschbar und unabhängig voneinander
- (viele) Änderungen ohne Modifikation der Schnittstelle möglich
- Module einzeln testbar

Geheimnisprinzip

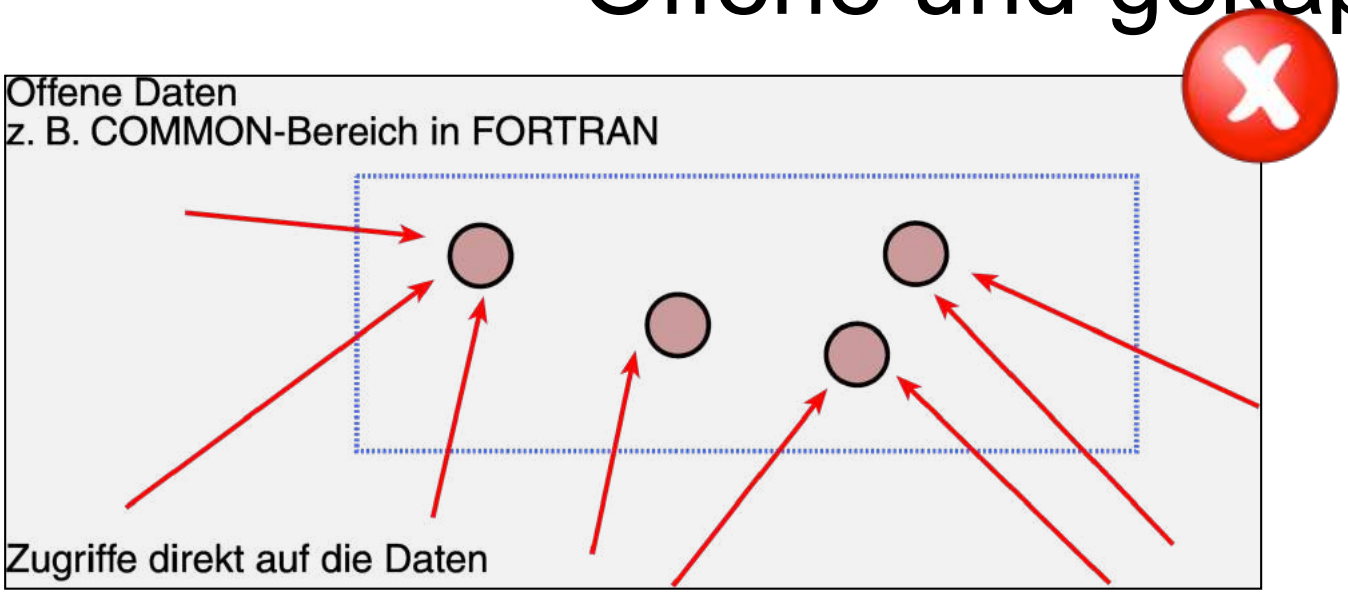
(Kapselung, Modularisierung, Information Hiding)

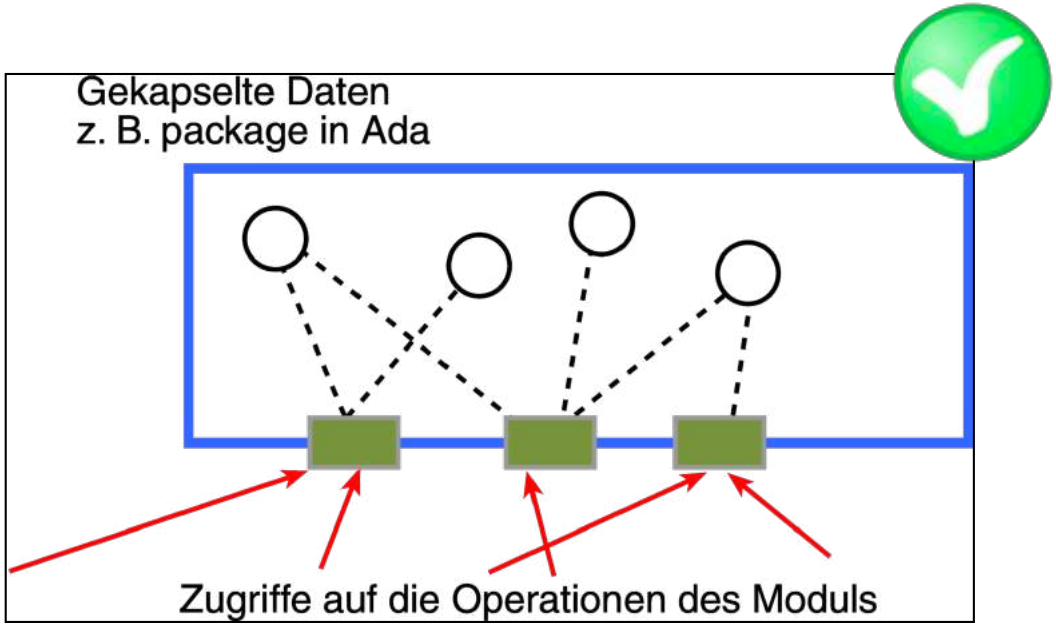
Information Hiding — *A software development technique in which each module's interfaces reveal as little as possible about the module's inner workings, and other modules are prevented from using information about the module that is not in the module's interface specification.*

IEEE Std 610.12 (1990),
ursprünglich von David Parnas

- Bausteine verkapseln Geheimnisse
- Schutz von Daten und inneren Abläufen eines Bausteins gegen Zugriff von außen
- Dienste und Daten von Bausteinen nur über öffentliche Schnittstellen nutzbar

Offene und gekapselte Daten

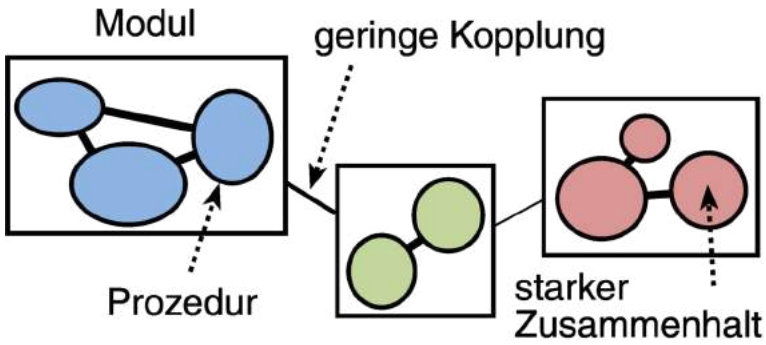




Basiert auf Ludewig, J., H. Lichter: Software Engineering – Grundlagen, Menschen, Prozesse, Techniken. 2. Aufl., dpunkt.verlag, 2010.

Lose (niedrige, geringe) Kopplung

- Kopplung definiert den Grad der Abhängigkeit
  - … zwischen zwei oder mehr Bausteinen
- Hohe (starke) Kopplung führt zu…
  - Änderungskaskaden
  - Geringer Verständlichkeit -> hohem Änderungsaufwand
  - Schwierigkeiten bei arbeitsteiliger Entwicklung
  - Geringe Wiederverwendbarkeit



Arten von Kopplung:

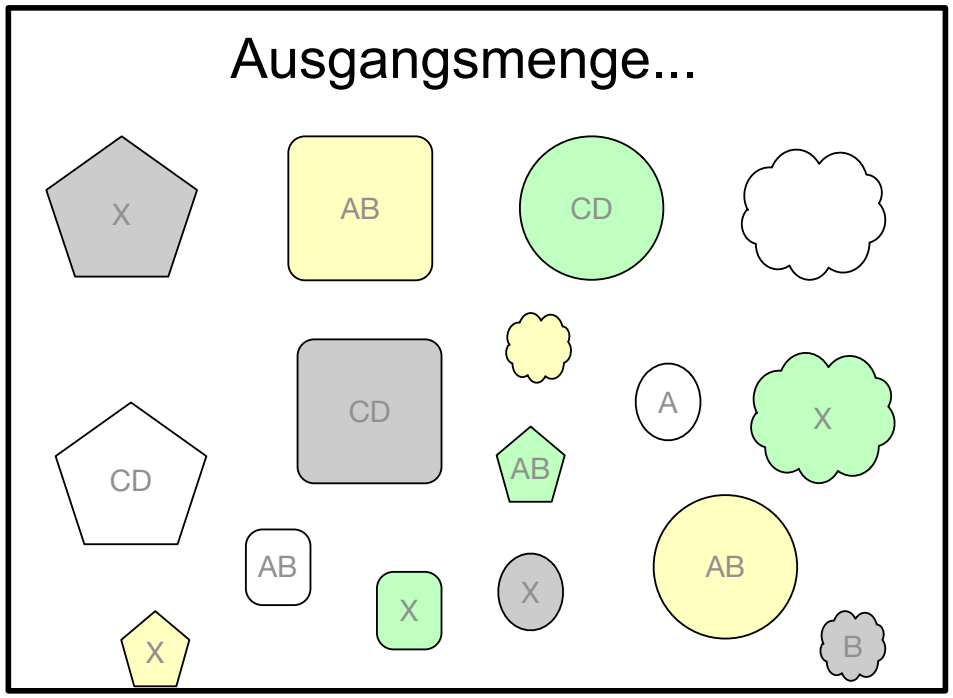
- Aufruf
- Erzeugung
- Daten
- Hardware
- Laufzeitumgebung
- Zeit
- …

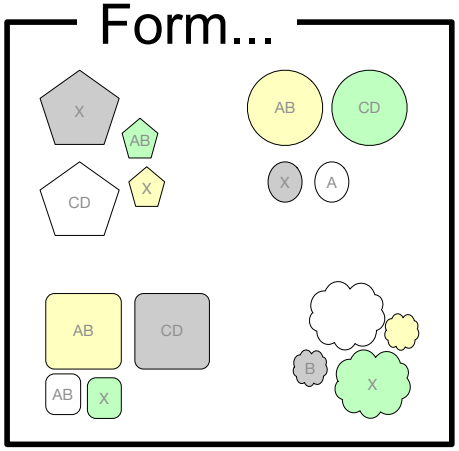
Hohe Kohäsion

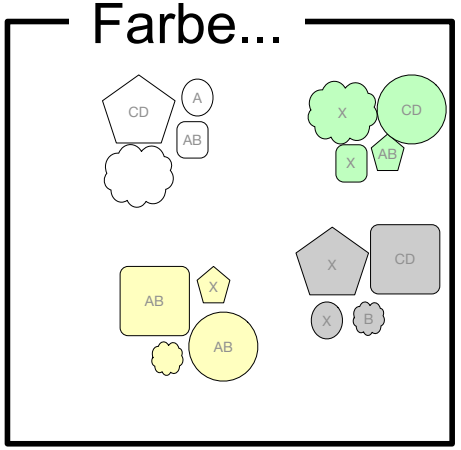
Definiert, wie eng Bausteine **inhaltlich** zusammen gehören

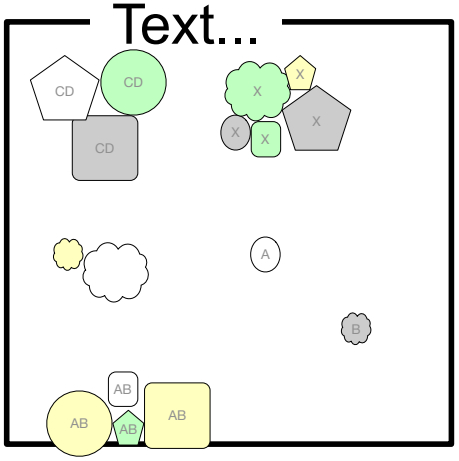
- Baustein hoher Kohäsion erledigt inhaltlich zusammengehörige Aufgaben
- Baustein mit niedriger Kohäsion erledigt:
  - keine zusammenhängende Aufgaben
  - zu viele Aufgaben

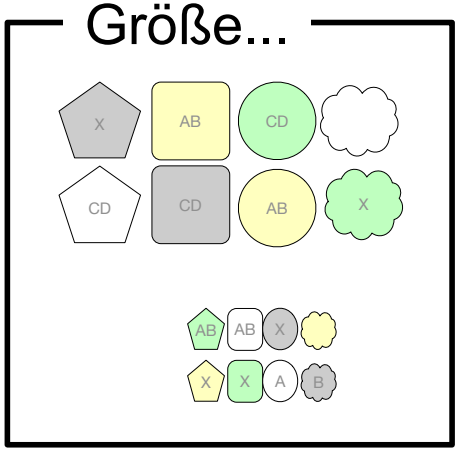
Kohäsionskriterien sind Entwurfsentscheidung











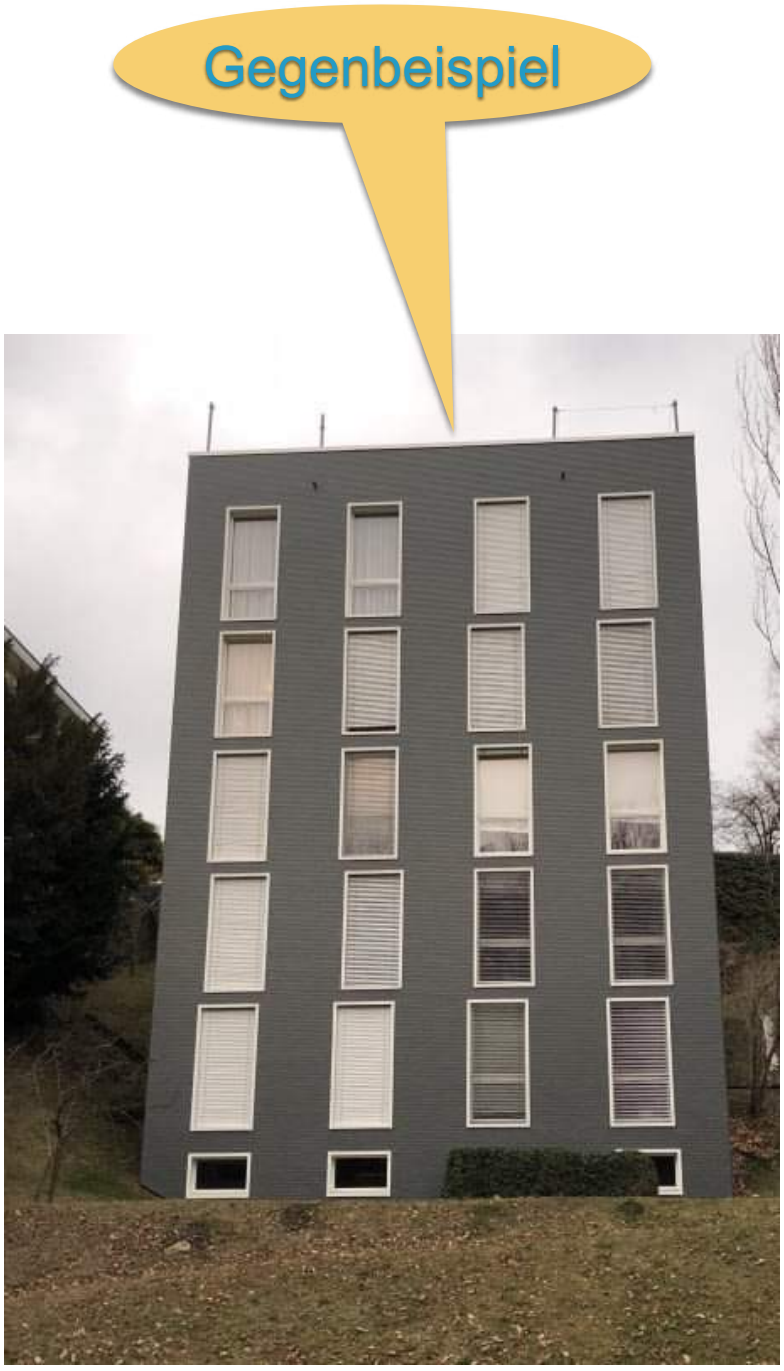
Mögliche Kohäsionskriterien...

- Daten
- Datentypen
- Abläufe / Prozesse / Services
- Größe der einzelnen Teile
- Volumen / Menge / Anzahl
- Klienten
- Vertraulichkeit
- Qualitätsanforderungen

- Verteilung / Hardware
- Implementierungs-technologie
- Datenbank, Middleware
- Zugänge (online, API...)
- Organisatorische Verantwortung
- Ertrag, Umsatz
- Geographie
- <u.v.a.m>

Konsistenz (Homogenität)

- “Konzeptionelle Integrität”
- Identische Probleme identisch lösen sofern angemessen.
- Angemessene Standardisierung von Lösungsansätzen
- Ziel: hohe Konsistenz



SOLID-Prinzipien

S	SRP	Single Responsibility Principle „A class should have one, and only one, reason to change“
O	OCP	Open Closed Principle „You should be able to extend a classes behavior, without modifying it“
L	LSP	Liskov Substitution Principle „Derived classes must be substitutable for their base classes“
I	ISP	Interface Segregation Principle „Make fine grained interfaces that are client specific“
D	DIP	Dependency Inversion Principle „Depend on abstractions, not on concretions“

Robert Martin (aka Uncle Bob):

http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod
http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf

Single Responsibility-Prinzip (SRP)

Jeder Baustein sollte nur **eine** fest definierte Aufgabe erfüllen

- auch: Separation-of-Concern
- Robert C. Martin:
  - „Es sollte nie mehr als einen Grund geben einen Baustein zu ändern“

- Verstoß: Verantwortlichkeit auf mehrere Bausteine verteilt
  - Enge Kopplung, Änderung wirkt sich auf viele Bausteine aus
- Verstoß: Ein Baustein hat mehrere Verantwortlichkeiten
  - Kopplung der Verantwortlichkeit, Änderung einer Verantwortlichkeit beeinflusst die anderen („Gottklassen”)

Open-Closed-Prinzip (OCP)

Bausteine sollen

- offen für Erweiterungen sein, aber
- geschlossen für Modifikationen

- Erweiterung: Ändert bestehenden Code nicht
- Modifikation: Ändert bestehenden Code

Anti-Beispiel (wie man es nicht machen sollte…):

```
void draw(Form f) {
   if (f.type == circle) drawCircle(f);
   else if (f.type == square) drawSquare(f);
    ...
```

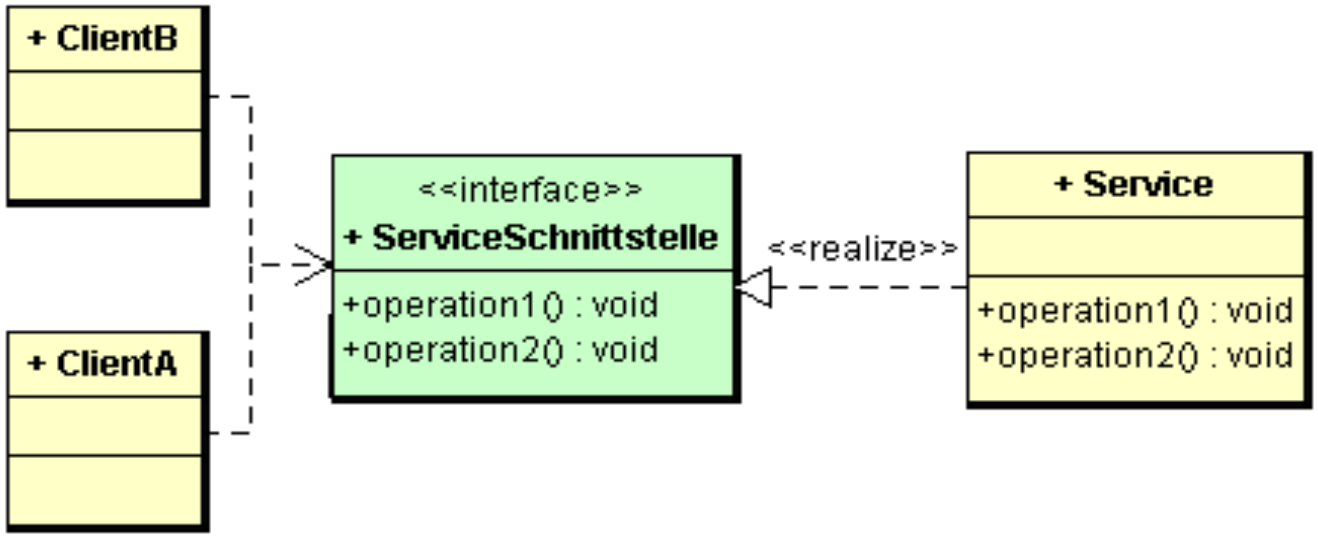
Liskov ‘sches Substitutionsprinzip

Instanzen von Unterklassen sind immer für ihre Oberklassen einsetzbar

- Wenn Sie Vererbung einsetzen, dann bitte richtig!
- Indikatoren für LSP-Verletzungen:
  - Unterklassen überschreiben Methoden der Oberklasse durch leere Implementierungen (d.h. sie entfernen Funktionalität der Oberklassen)
  - Unterklassen werfen Exceptions, die Oberklassen nicht werfen.

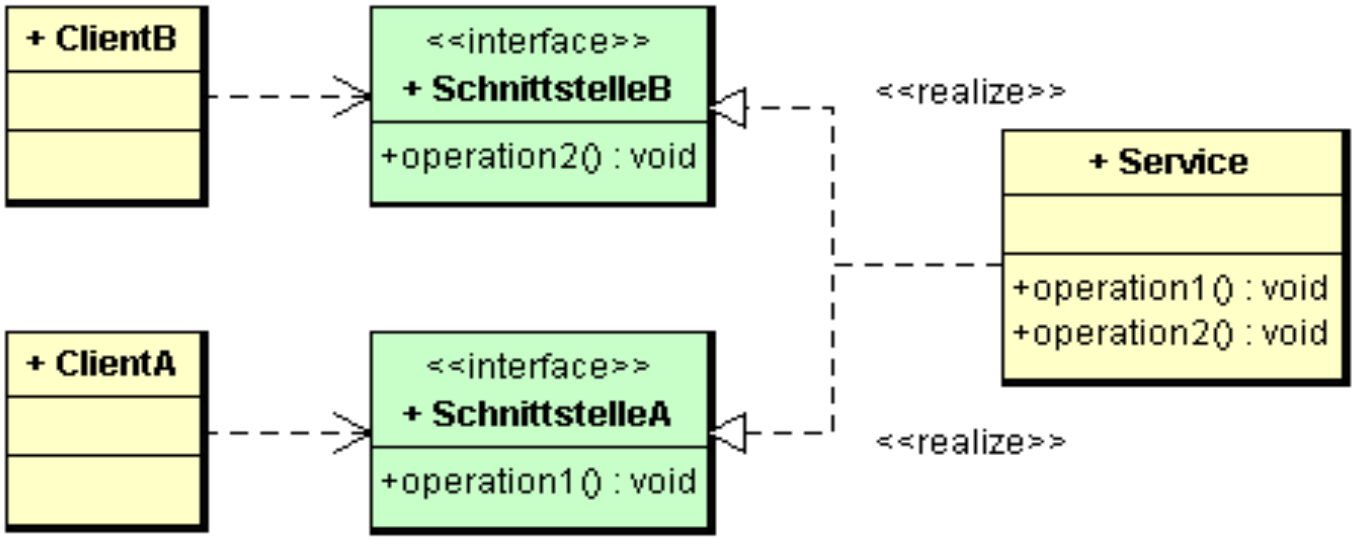
Interface Segregation-Prinzip (ISP)

- Kopplung der Clients durch Verwendung einer globalen Schnittstelle
- Änderungen betreffen u.U. beide Clients



Interface Segregation-Prinzip (ISP)

- Besser: Spezielle Schnittstellen

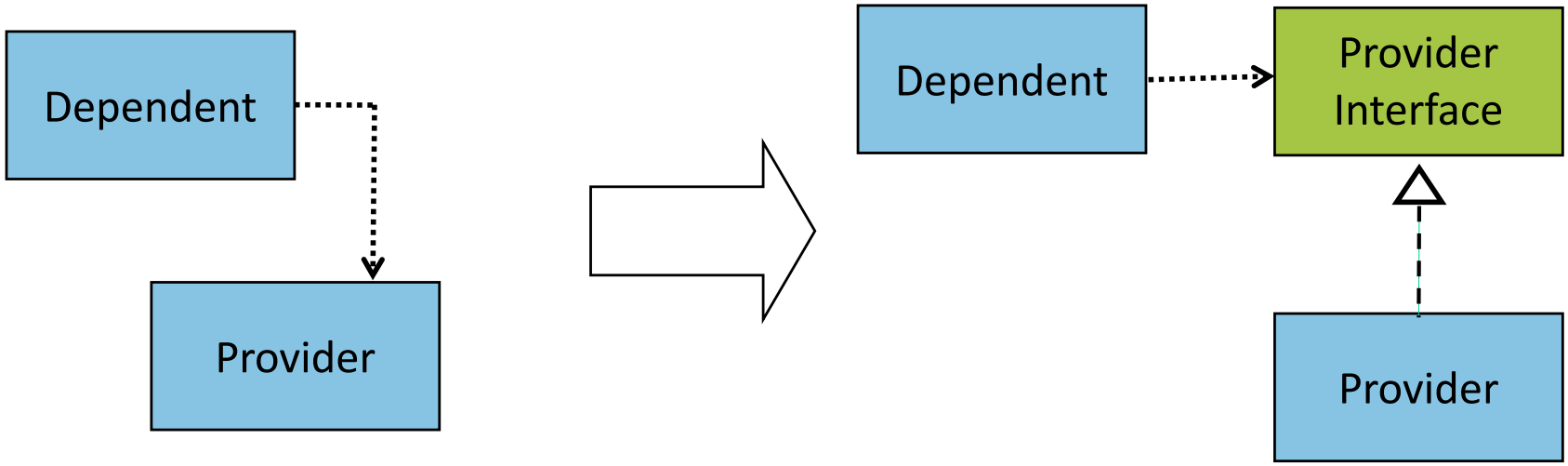


„Clients sollten nicht von Methoden abhängen, die sie nicht benutzen“

Dependency Inversion-Prinzip (DIP)

Abstraktionen sollten nicht von Details abhängen,
Details sollten von Abstraktionen abhängen.

- Bausteine höherer Ebenen sollen nicht von Bausteinen niederer Ebenen abhängen, beide sollten von Abstraktionen abhängen.



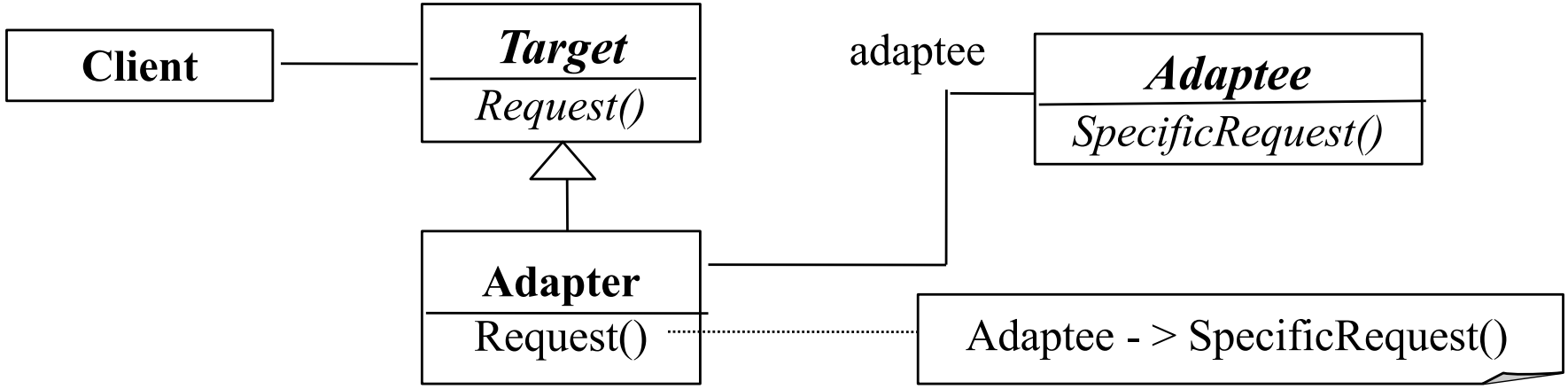
Design Patterns

Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, “Design Patterns”, Addison-Wesley 1995.

- **5 Creational Patterns**: Abstract Factory, Builder, Factory Method, Prototype, Singleton
- **7 Structural Patterns**: Adapter, Bridge, Composite, Decorator, Façade, Flyweight, Proxy
- **11 Behavioral Patterns**: ChainOfResponsibility, Command, Interpreter, Iterator, Mediator, Memento, Observer, State, Strategy, Template Method, Visitor

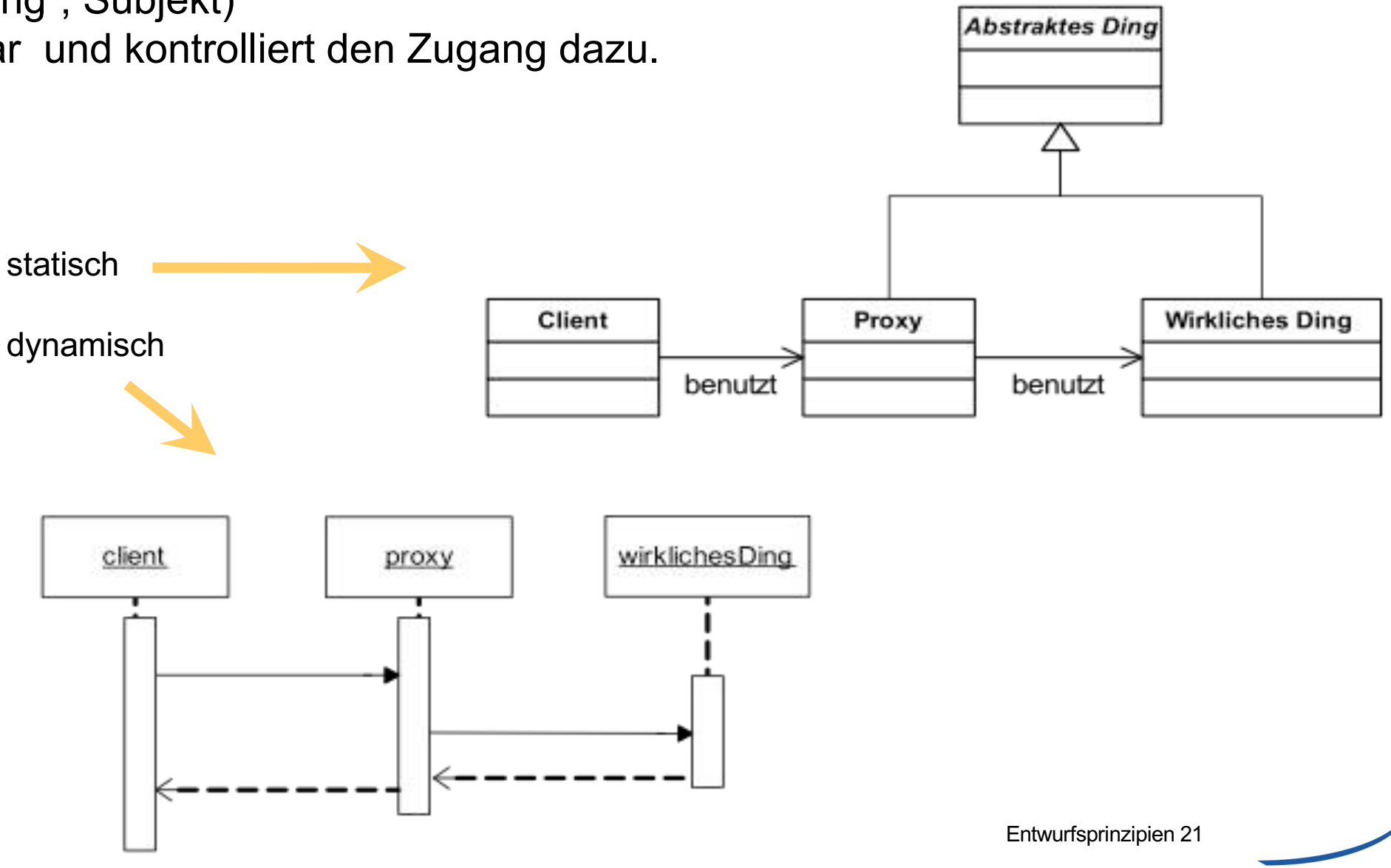
Adapter (a.k.a. Wrapper)

Clients call operations on an Adapter instance. In turn, the adapter calls Adapter operations that carry out the request.



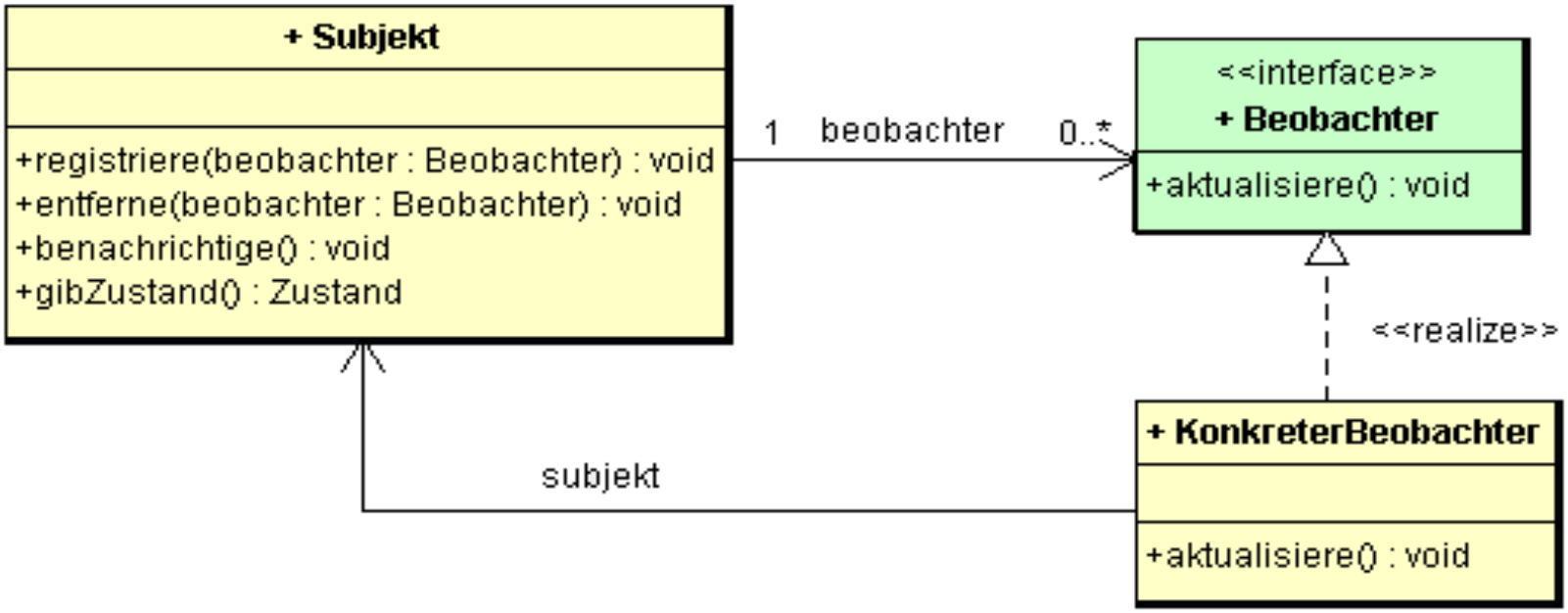
Entwurfsmuster: Proxy

- Proxy stellt einen Platzhalter für eine andere Komponente („Wirkliches Ding“, Subjekt)
  dar und kontrolliert den Zugang dazu.



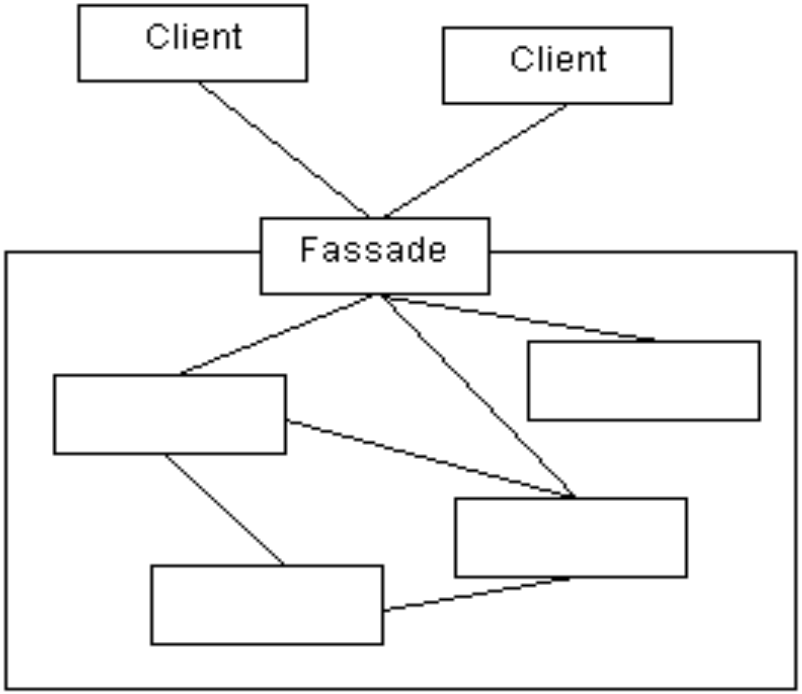
Muster: Observer

- ermöglicht einem oder mehreren Beobachtern, automatisch auf Zustandsänderung eines bestimmten Objekts zu reagieren, um den eigenen Zustand anzupassen



Muster: Facade (Fassade)

- Problemstellung
  - Einfache Schnittstelle zu einem komplexen Subsystem
  - Aufteilung eines Systems in Schichten und Definition eines Eintrittspunkts in die Schicht
- Lösung: Facade



Zusammenfassung

- Wichtige Designprinzipien:
  - Coupling & Cohesion / Geheimnisprinzip / Modularität
  - SOLID-Prinzipien
- Kohesionskriterien sind Entwurfentscheidungen!
- Architektur- und Entwurfsmuster können bei Strukturentwurf helfen

  Tipps:
  - Muster NIE zum Selbstzweck