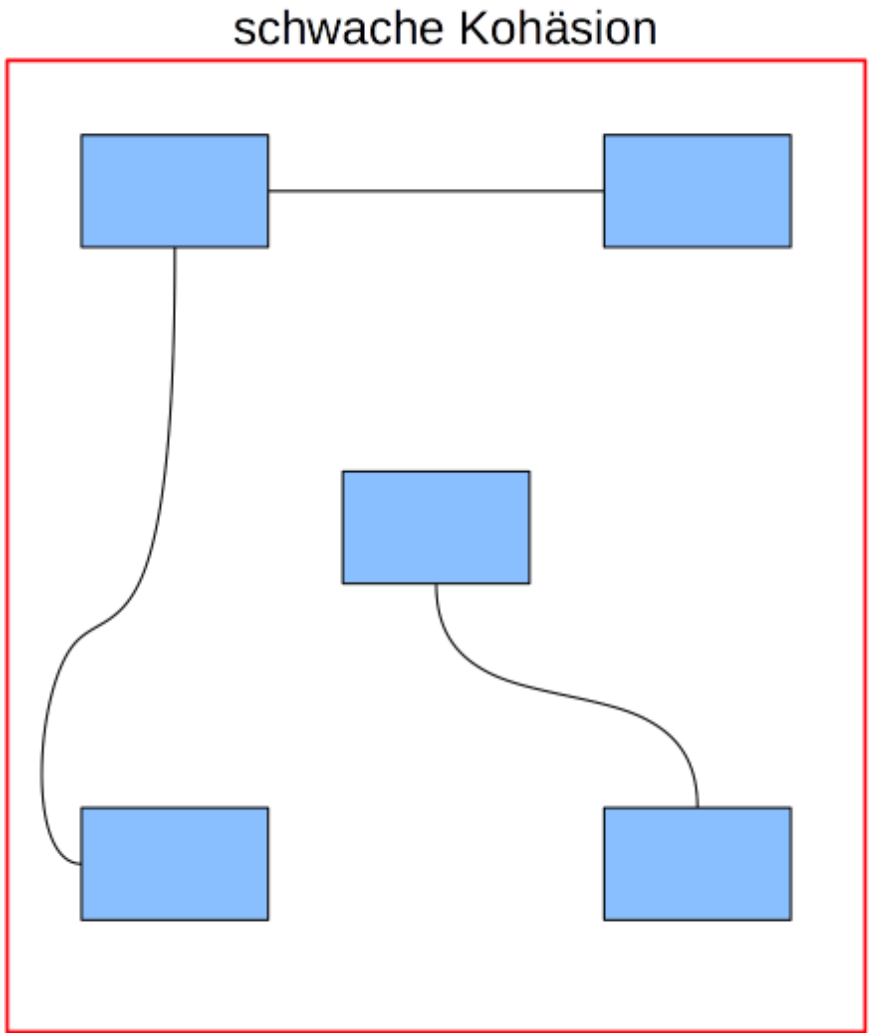
Übung 05

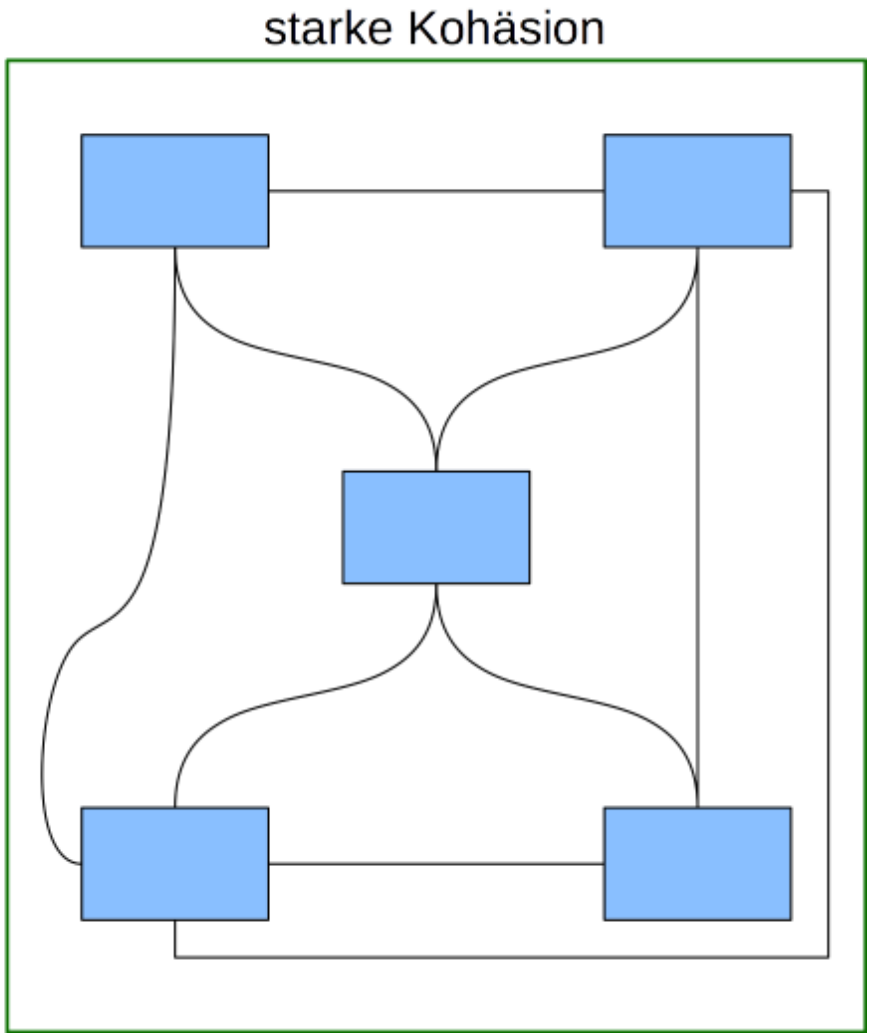
Design und Architektur

1.1 Kohäsion

- qualitatives Maß für die Kompaktheit einer Komponente
- **Idealzustand:** jede Komponente (Methode, Klasse) für eine Aufgabe verantwortlich, die sie eigenständig lösen kann
- **ausschlaggebend:** Beziehungen zwischen Elementen *innerhalb einer Komponente*
- **starke Kohäsion**: alle Elemente sind nötig für die Funktionstüchtigkeit der anderen internen Elemente, d.h. keine isolierten Elemente
- **schwache Kohäsion**: Komponente erfüllt viele verschiedene „responsibilities“ oder Elemente sind nur zusammengefasst, weil sie ähnliche Funktionalitäten anbieten
  - ➢ Ausgliedern in neue Komponente bietet sich an

1.1 Kohäsion



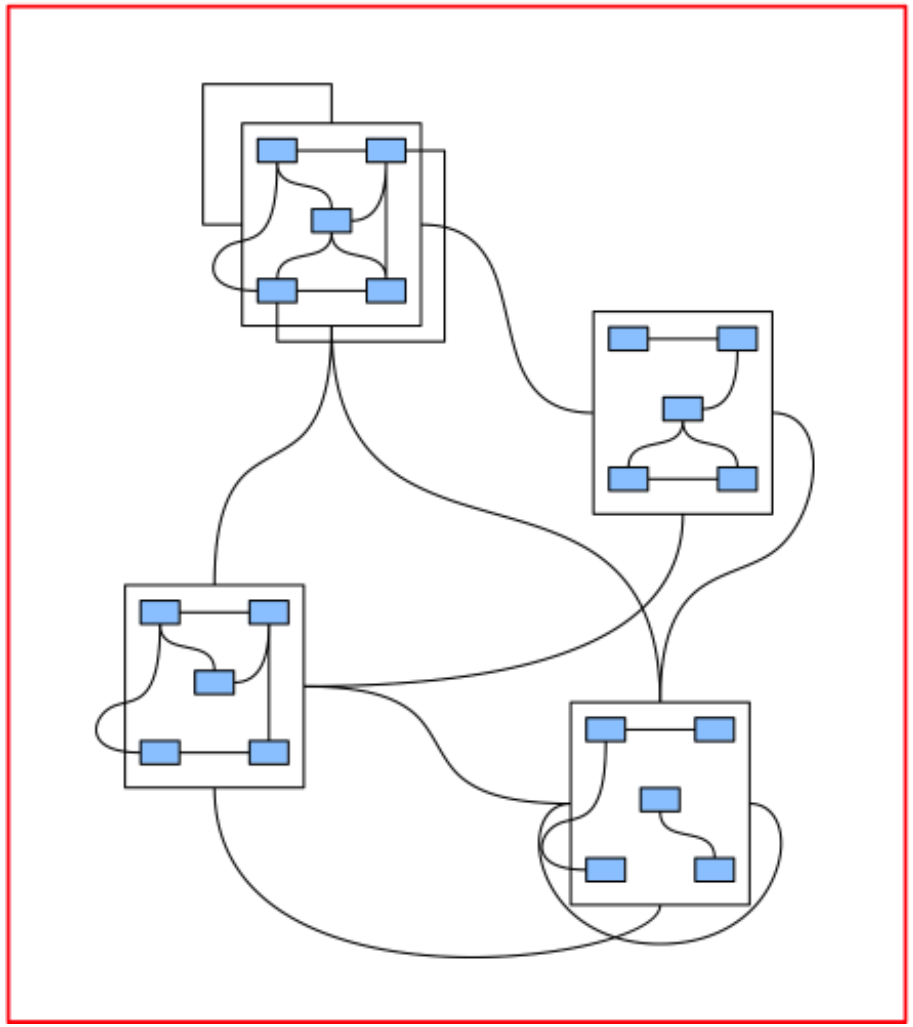


1.2 Kopplung

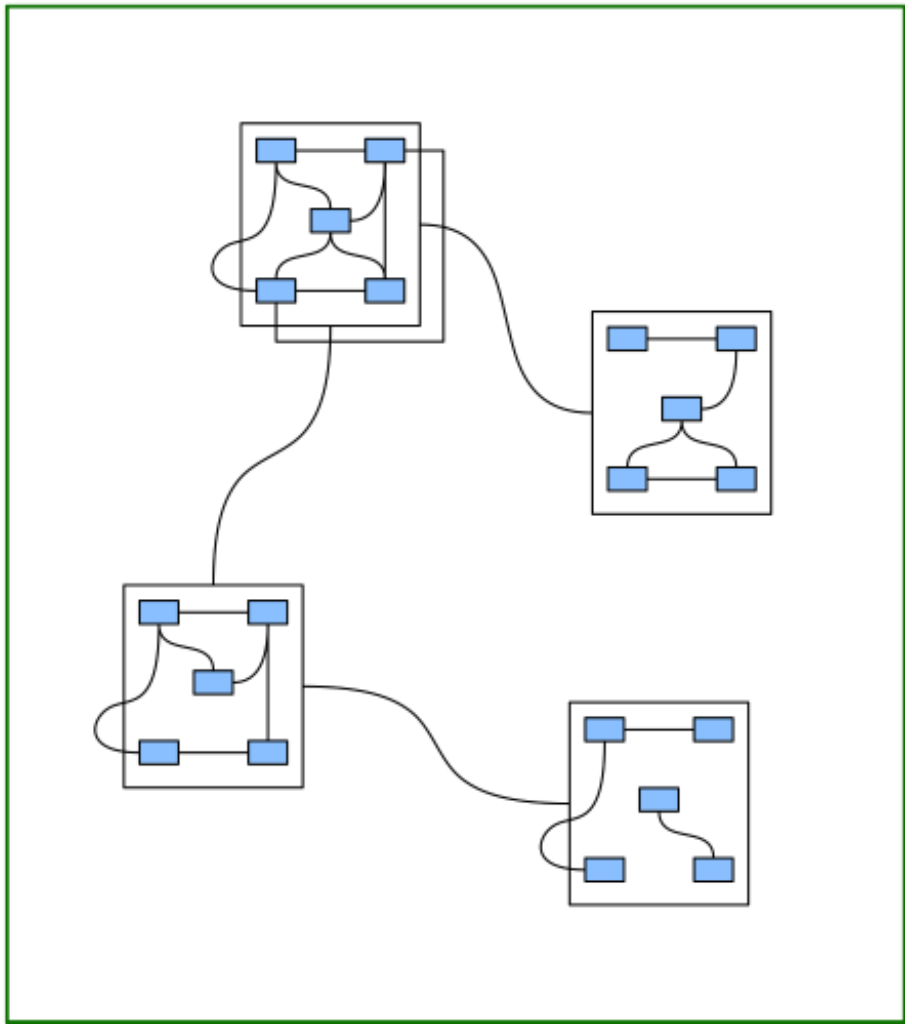
- qualitatives Maß für die Schnittstellen zwischen Komponenten
- **Idealzustand:** jede Komponente (Methode, Klasse) nur lose mit anderen verbunden
- **ausschlaggebend**:
  - Kopplungsmechanismus
  - Schnittstellenbreite
  - Kommunikationsart
- **lose Kopplung:** Komponenten besitzen wenige Abhängigkeiten untereinander
- **enge Kopplung:** Komponenten besitzen viele Abhängigkeiten untereinander
  - ➢ Eine Änderung hat möglicherweise Auswirkung auf viele andere Komponenten!

1.2 Kopplung

enge Kopplung



lose Kopplung



1.2 Kopplung

enge Kopplung

```
A::doSomething() {
    B b = new B();

    b.doSomethingElse();
}
```

lose Kopplung

```
A::doSomething(B b) {


    b.doSomethingElse();
}
```

1.2 Kopplung

enge Kopplung

```
Person::writeToDatabase() {
   DB db = new SQLiteDB();
   db.updateRecord(this);
}

…

/* client code */
Person p = new Person("v", “n“);

p.writeToDatabase();
```

lose Kopplung

```
Person::writeToDatabase(DB db) {
   db.updateRecord(this);
}

…

/* client code */
Person p = new Person(“v“, “n“);
DB db = new SQLiteDB();

p.writeToDatabase(db);
```

1.3 Kohäsion & Kopplung

- hohe Kohäsion und geringe Kopplung fördert Wartbarkeit und Anpassbarkeit
  - Effekte von Änderungen auf engem Kreis von Komponenten beschränkt
- geringe Kohäsion und starke Kopplung
  - in wie vielen und welchen Komponenten sind Änderungen nötig ..?

➢perfekter Zustand unerreichbar: Blob vs. einsame Klasse

Visitor Pattern:

- Kohäsion:
  - ✓ hoch, da jeder Visitor genau eine Funktionalität anbietet
  - ✓ hoch, da die Objektstruktur genau eine Funktionalität anbietet
- Kopplung:
  - ✗ hoch, da Visitor Zugriff auf interne Datenstruktur benötigt
  - ✗ hoch, da Visitor durch die Objektstruktur iterieren können muss

http://www.hwk-chemnitz.de/fileadmin/user_upload/ausbildung/Pruefungen/Gesellenpruefungen/Fragezeichen.JPG

2. Model-View-Controller (MVC)

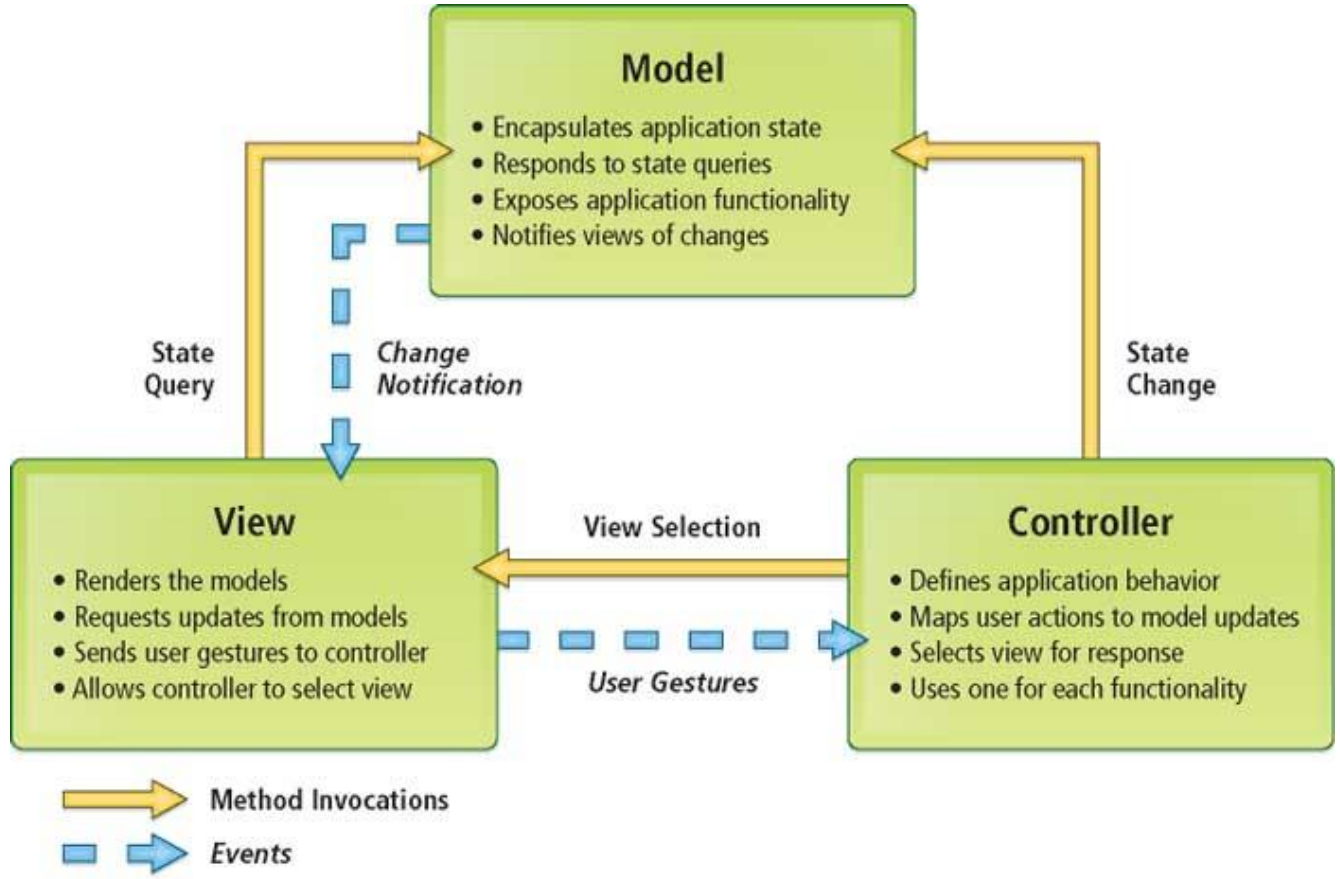
- erstmals beschrieben und implementiert in Smalltalk-80, ca. 1980

- **Ziele** der Entkopplung von Modell und View sind:
  - ✓ Komplexität der gesamten Architektur wird verringert
  - ✓ Wiederverwendbarkeit, Flexibilität und Wartbarkeit des Programms wird erhöht

- **Vorteile** der Entkopplung von Modell und View sind:
  - ➢ höhere Kohäsion und geringere Kopplung als mit naivem/monolithischem Ansatz
  - ➢ einfaches *unit-testing* möglich durch *separation of concerns*
  - ➢ gleichzeitige Anzeige mehrerer Views pro Modell möglich
  - ➢ Anpassung (neuer roter Button) an View möglich, ohne Modell ändern zu müssen
  - ➢ verschiedene Views in Abhängigkeit vom Ausgabegerät (PDA, Smartphone, etc.)

2. Model-View-Controller (MVC)

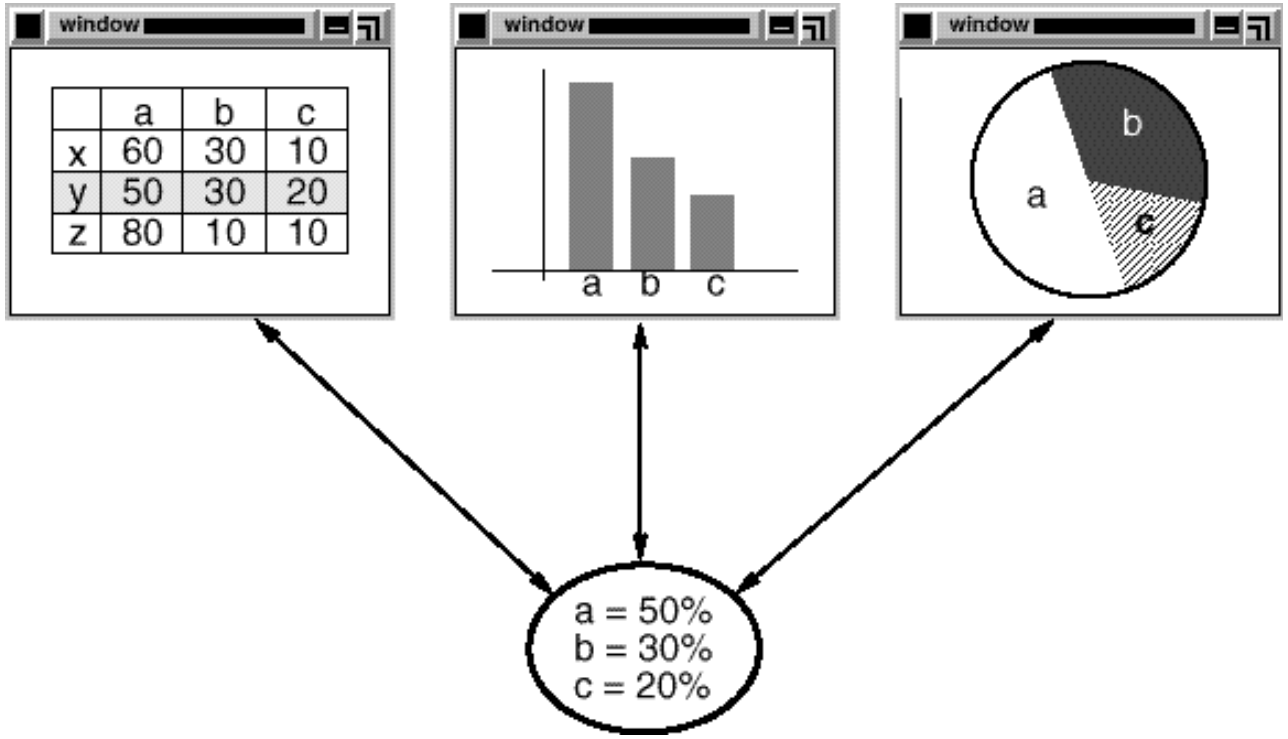
- **Modell**
  - kapselt Zustand der Applikation
  - nimmt Zustandsanfragen entgegen
  - benachrichtigt Views über Änderungen
- **View**
  - fordert Daten von Modell an
  - zeigt Modell an
  - sendet User-Aktionen an Controller
- **Controller**
  - definiert Verhalten der Applikation
  - verknüpft User-Aktionen mit Modell-Updates
  - wählt entsprechende View aus



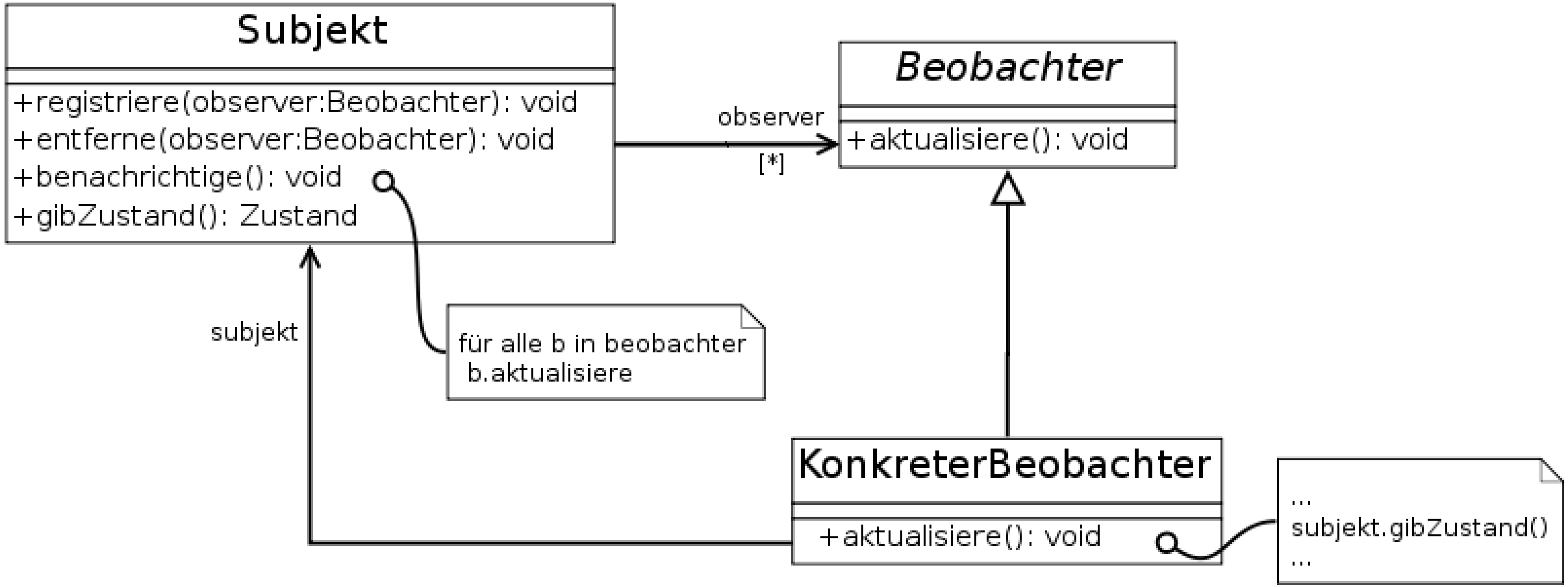
Implementierung bedient sich des **Observer-Patterns** und ggf. des **Composite-Patterns**

Zusätzlich: Observer-Pattern – Motivation

- mehrere visuelle Repräsentationen der Anwendungsdaten gewünscht
- Klassen sollen unabhängig voneinander sein
- Konsistenz der Repräsentationen muss jederzeit gewährleistet sein



Zusätzlich: Observer-Pattern – Struktur



http://www.hwk-chemnitz.de/fileadmin/user_upload/ausbildung/Pruefungen/Gesellenpruefungen/Fragezeichen.JPG