MVC — Ein wichtiges Konzept der Software-Architektur

Dr. Arthur Zimek

Ludwig-Maximilians-Universität München,
Institut für Informatik,
LFE Datenbanksysteme

Softwareentwicklungspraktikum – Wintersemester 2011/12

Modelle und Anwendungen

- Unsere Programme in der Vorlesung EIP waren oft eine Mischung aus Modellen und Anwendungen.
- Ein Modell stellt einen Ausschnitt der Welt in vereinfachter, schematisierter Form dar (z.B. die Klasse `Konto`).
- Eine Anwendung verwendet ein Modell um z.B. etwas zu berechnen, oft abhängig von Eingaben des Benutzers (z.B. eine Geschäftsabwicklung, bei der Konten angelegt und ihr Inhalt verändert wird).

Modelle und Anwendungen

- Als Anwendungen haben wir bisher `main`-Methoden kennengelernt.
- Modelle wurden bei uns bisher auf der Konsole dargestellt (`System.out.print`-Ausgaben).
- Interaktion war durch Eingabe auf die Konsole möglich (z.B. durch Start-Parameter).
- Die Konsole stellt damit eine bedeutende Schnittstelle zum Benutzer dar (Command-Line-Interface, CLI).
- Jede Klasse, die eine `main`-Methode zu Verfügung stellt, ist damit eine Anwendung (Applikation).
- Ein einziges Modell kann in sehr unterschiedlichen Anwendungen verwendet werden.

Verschiedene Arten von Anwendungen

- Eine Anwendung kann auf verschiedene Arten mit einem Benutzer interagieren.
- Neben CLI ist die bedeutenste Schnittstelle die GUI (Graphical User Interface).
- Andere Möglichkeiten sind die Interaktion über Web-Oberflächen – z.B. als Applets oder als Java-Server-Pages (JSP).
- Wir werden in diesem Praktikum GUIs als sehr zentrale und Applets als java-typische, sehr schlanke Anwendungen kennenlernen.

Trennung von Modell und Benutzer-Interaktion

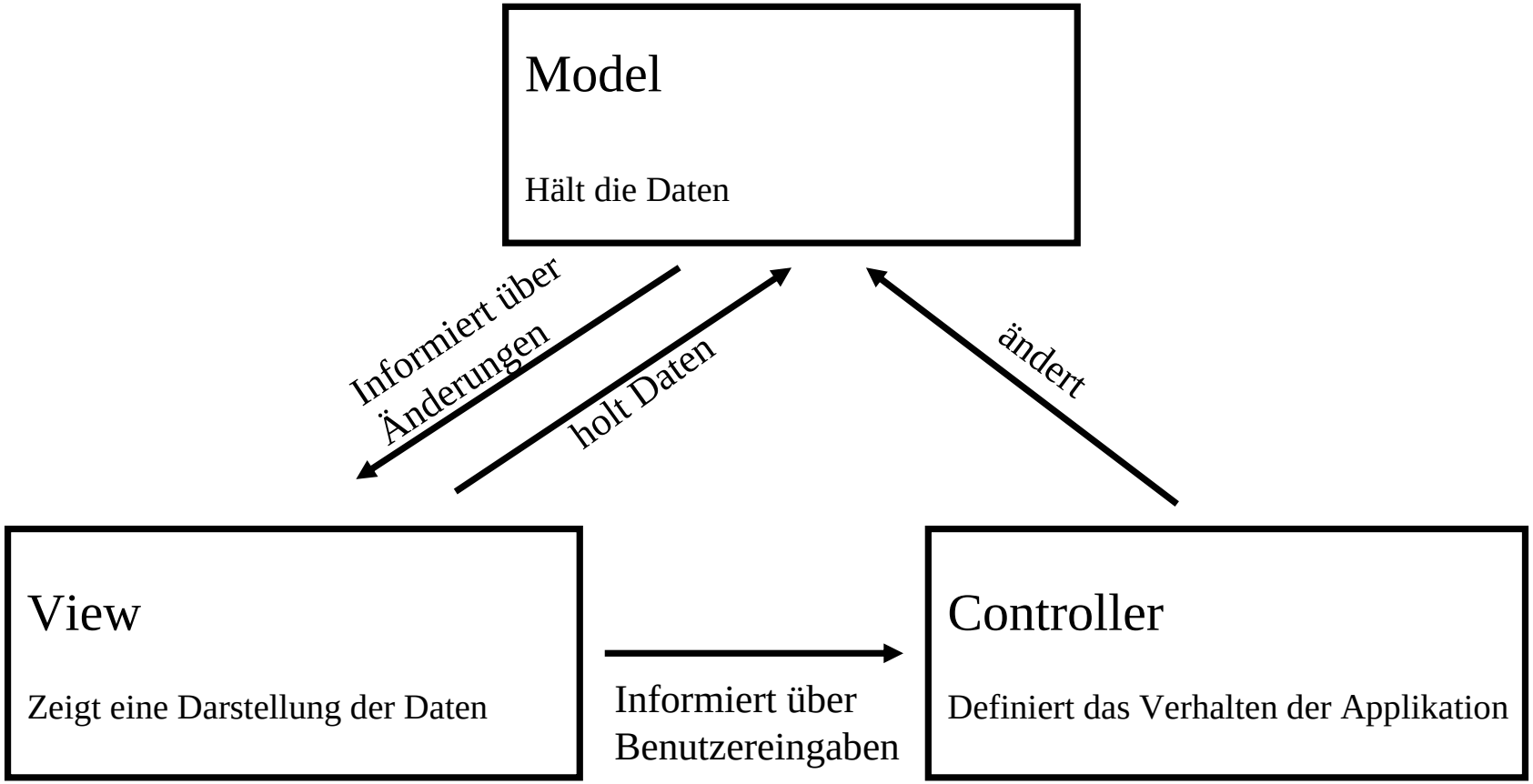
- Es ist eine wichtige Design-Hilfe für gute Programme, die Modellierung eines Ausschnitts der Wirklichkeit (das *Modell*) von den Möglichkeiten der Benutzer-Interaktion getrennt zu halten.
- In der Software-Entwicklung hat man für diese grundlegende Trennung das *Model-View-Controller* (*MVC*) Konzept geprägt.

Model-View-Controller

- Drei getrennte Programm-Komponenten sind für
  - das Modell (*Model*),
  - die Darstellung des Modells (*View*) und
  - die Beeinflussung des Modells (*Controller*)

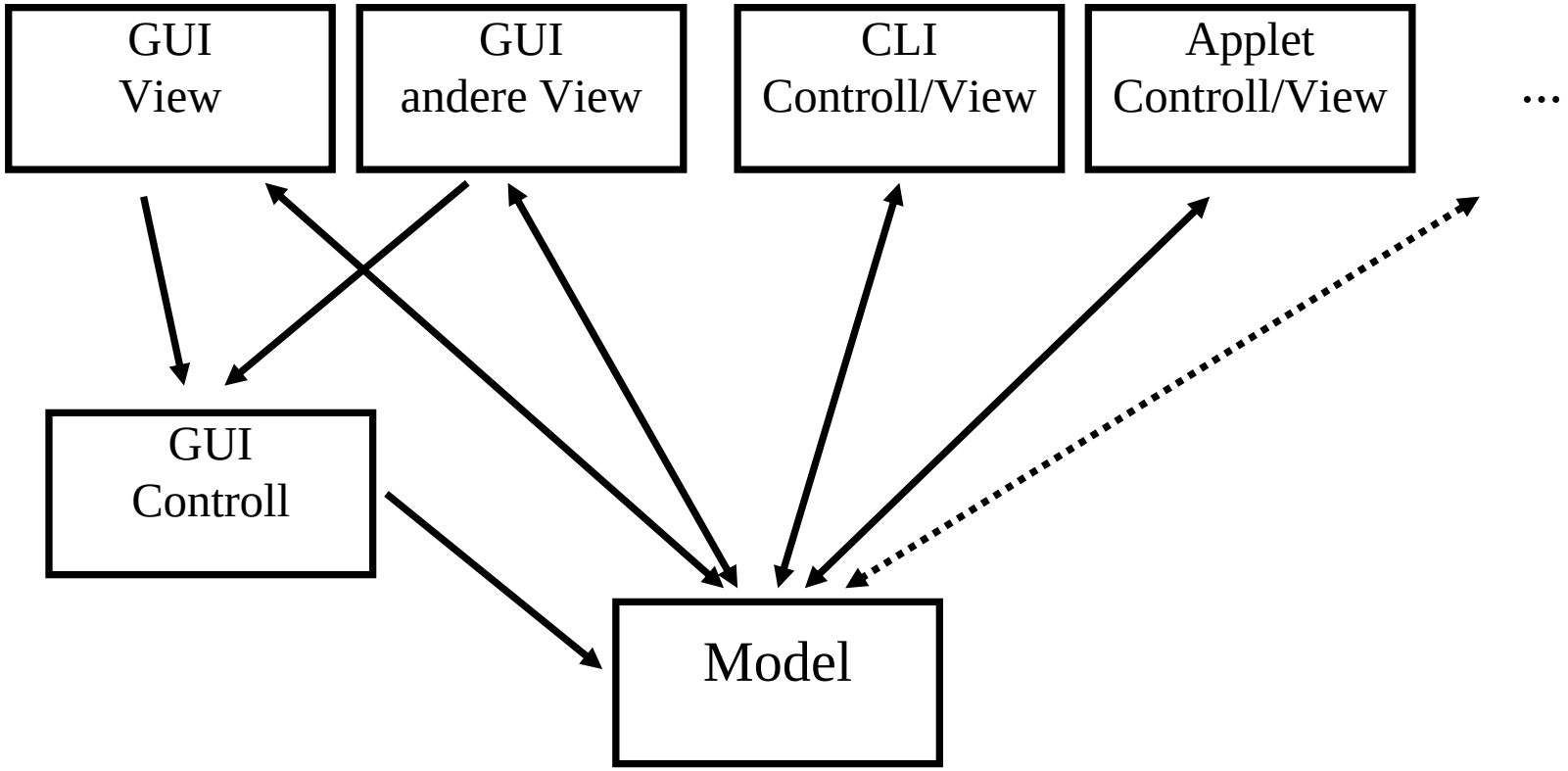
  zuständig.
- *View* und *Controller* hängen meist enger zusammen, da beide für eine bestimmte Art der Applikation (CLI, GUI, Applet, JSP, ...) bestimmt sind.
- Aber auch innerhalb des Anwendungsszenarios GUI ist es sinnvoll, View und Controller getrennt zu halten. Das Design der Darstellung ("Look-And-Feel") kann dadurch leicht ausgetauscht werden, ohne die Kontroll-Ebene zu beeinflussen.

Model-View-Controller



Model-View-Controller

Das Modell steht dem View-Controller-Paar eher unabhängig gegenüber und könnte auch von einem ganz anderen View-Controller-Paar verwendet werden.



Applikationen und Applets

- In der Vorlesung EIP haben wir implizit eine CLI-View-Controll verwendet.
- Um das MVC-Prinzip bei einfachen Aufgaben mit CLI einzuhalten, achtet man z.B. darauf, dass die Programmlogik (Methoden, die etwas berechnen) getrennt ist von der View (Methoden, die etwas ausgeben). Auf diese Weise könnte die Programmlogik auch von anderen View-Controller-Paaren verwendet werden.
- In diesem Praktikum werden wir nun auch Applets und GUI-Applikationen verwenden.

Weiterführende Informationen

- Java Sun:

  `http://java.sun.com/blueprints/patterns/MVC-detailed.html`
- S. Schubert, A. Schwill (2004): Didaktik der Informatik. Spektrum Akademischer Verlag, Heidelberg.
- G. Krüger, T. Stark (2009): Handbuch der Java-Programmierung. Addison-Wesley

  `http://www.javabuch.de`