MVC — Ein wichtiges Konzept der Software-Architektur

Julian Busch, Peer Kröger

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

Softwareentwicklungspraktikum – Wintersemester 2016/17
Folien basieren auf den Folien von Dr. Arthur Zimek, SoSe 2016

Modelle und Anwendungen

- 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 einfache Anwendungen kennen wir `main`-Methoden.
- Modelle werden z.B. auf der Konsole dargestellt (`System.out.print`-Ausgaben).
- Interaktion ist 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 Java-Server-Pages (JSP).
- Wir werden in diesem Praktikum GUIs als sehr zentrale Anwendungen implementieren, aber CLIs nicht zu sehr vernachlässigen.

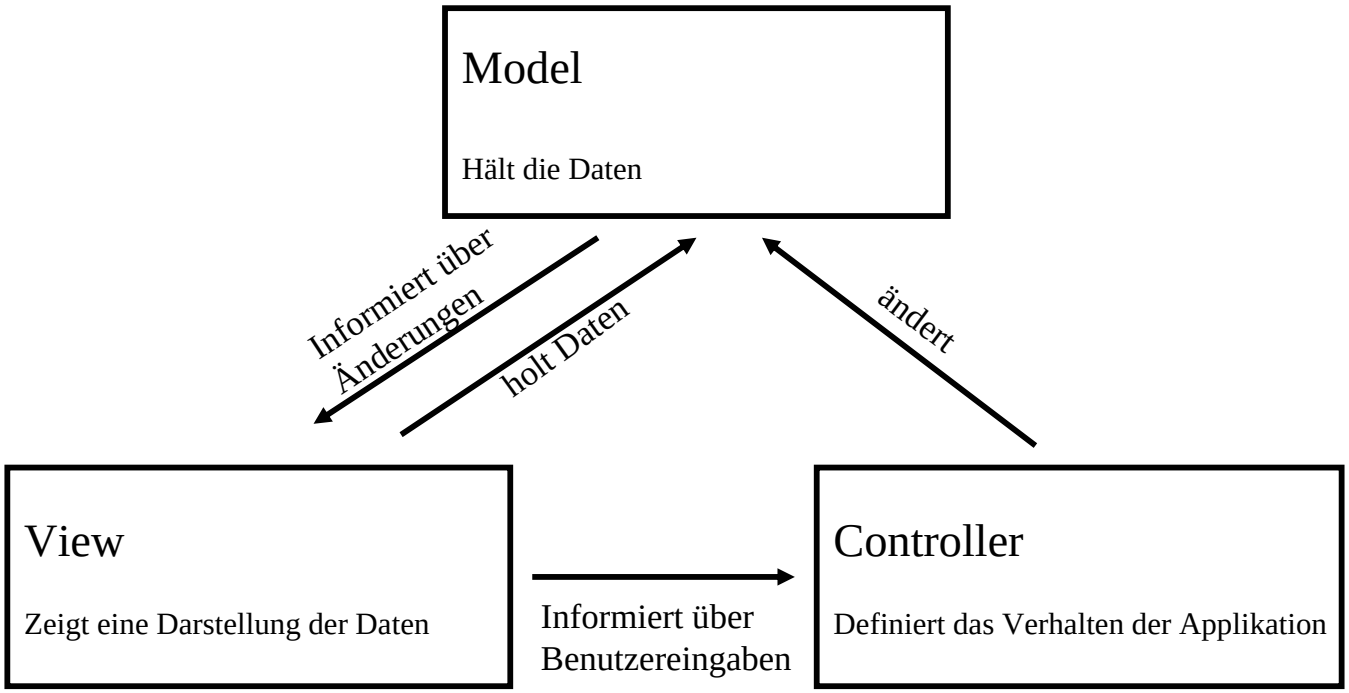
Trennung von Modell und Benutzer-Interaktion

- 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 zuständig für

- das Modell (*Model*),
- die Darstellung des Modells (*View*) und
- die Beeinflussung des Modells (*Controller*)

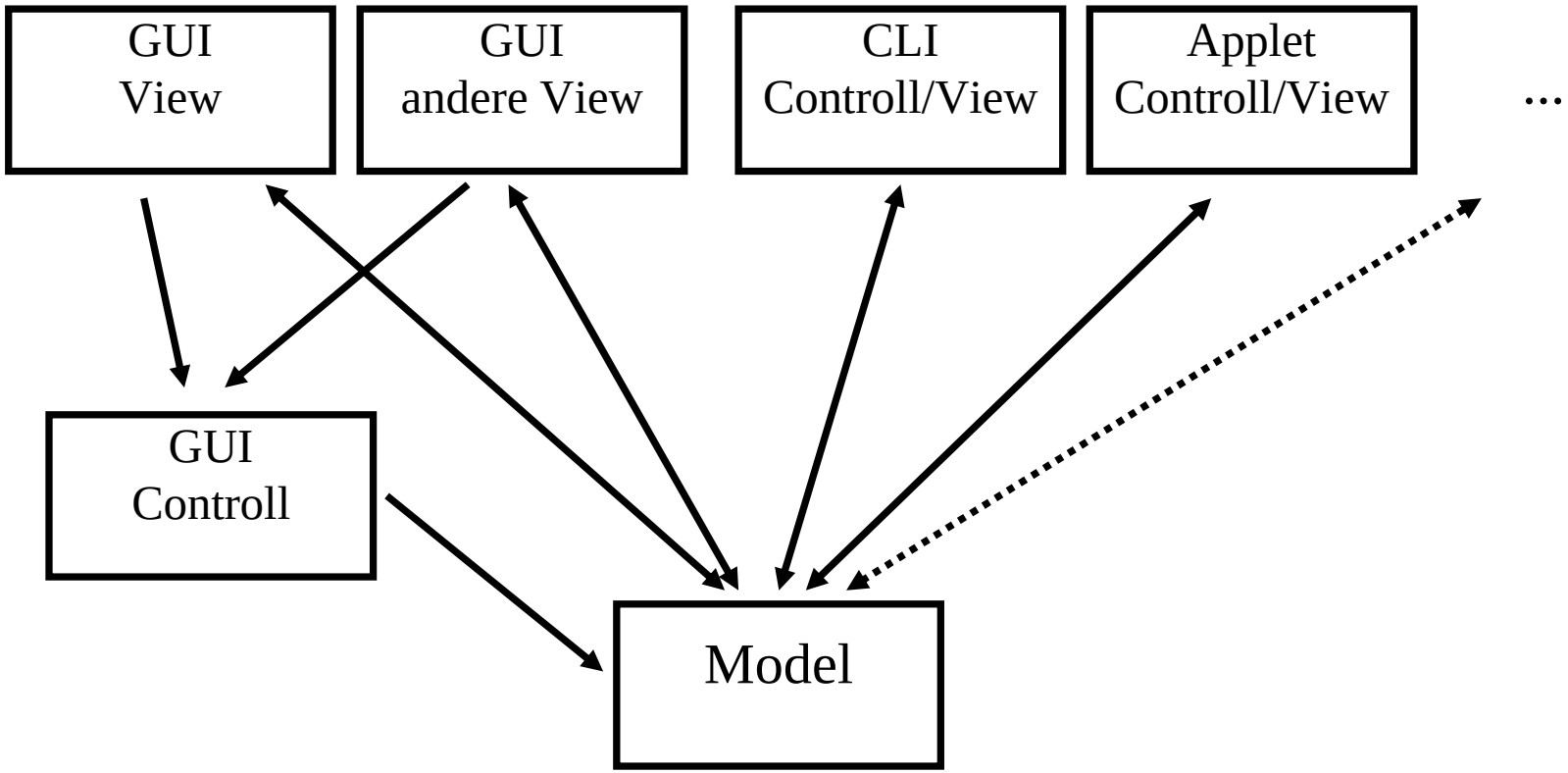


MVC in verschiedenen User Interfaces

- 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.
- *View* und *Controller* hängen meist enger zusammen, da beide für eine bestimmte Art der Applikation (CLI, GUI, 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

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



Varianten: Model - View - Presenter, Model - View - ViewModel

Weiterführende Informationen

- Java Sun:

  `http://www.oracle.com/technetwork/articles/javase/index-142890.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`