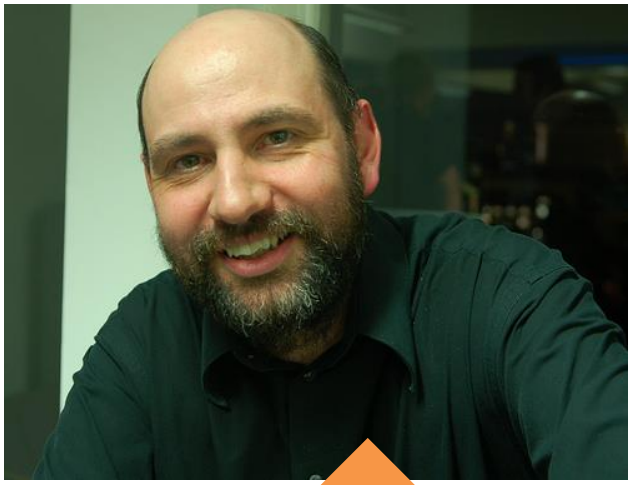
Model-View-Presenter

Übersicht

- Einführung
- Aufbau einer MVP-Anwendung
- Konkrete Problemstellung
- Mögliche Lösung
- Vor- und Nachteile
- Fazit
- Übung

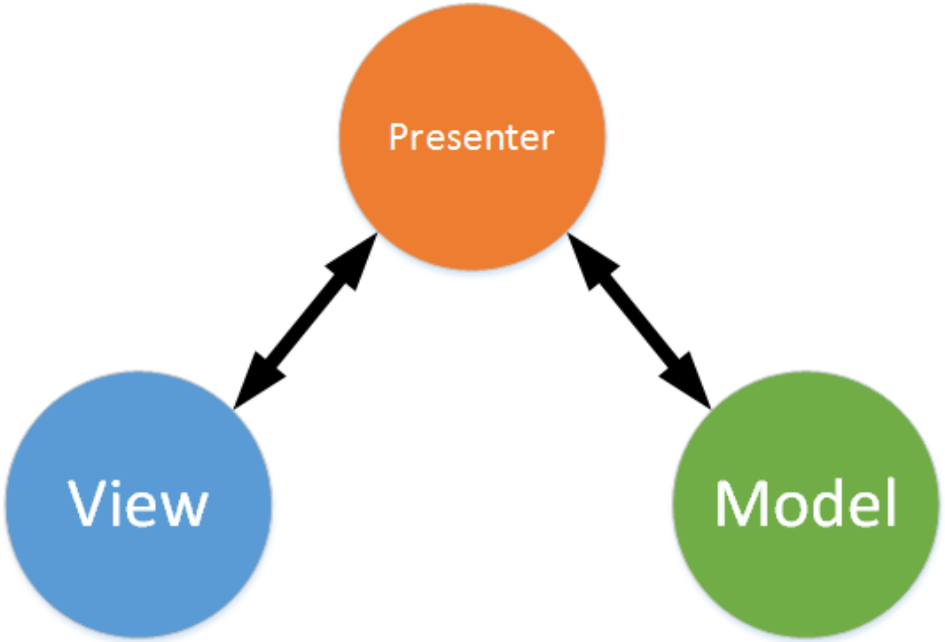
Was ist Model-View-Presenter?



- Weiterentwicklung von MVC
- Von Taligant und IBM entwickelt
- Weiterentwickelt durch Martin Fowler
- Keine genaue Definition

Unterschied zu MVC

Der Presenter ist einziges Bindeglied von Model und View -> View kennt das Model nicht!



Die zwei MVP-Arten (nach Fowler)

Supervising Controller (Überwachende Steuerung)

Passive View

Reduzierung der Programmlogik auf ein Minimum	Handler der GUI-Elemente werden in den Presenter ausgelagert

Wir betrachten nur die Implementierung mit passiver View

Aufgabe: Model

identisch zu MVC

Aufgabe: Der Presenter

- Bindeglied zwischen Model und View
- Interpretation von Benutzereingaben

(Verwendung: passive View)

…

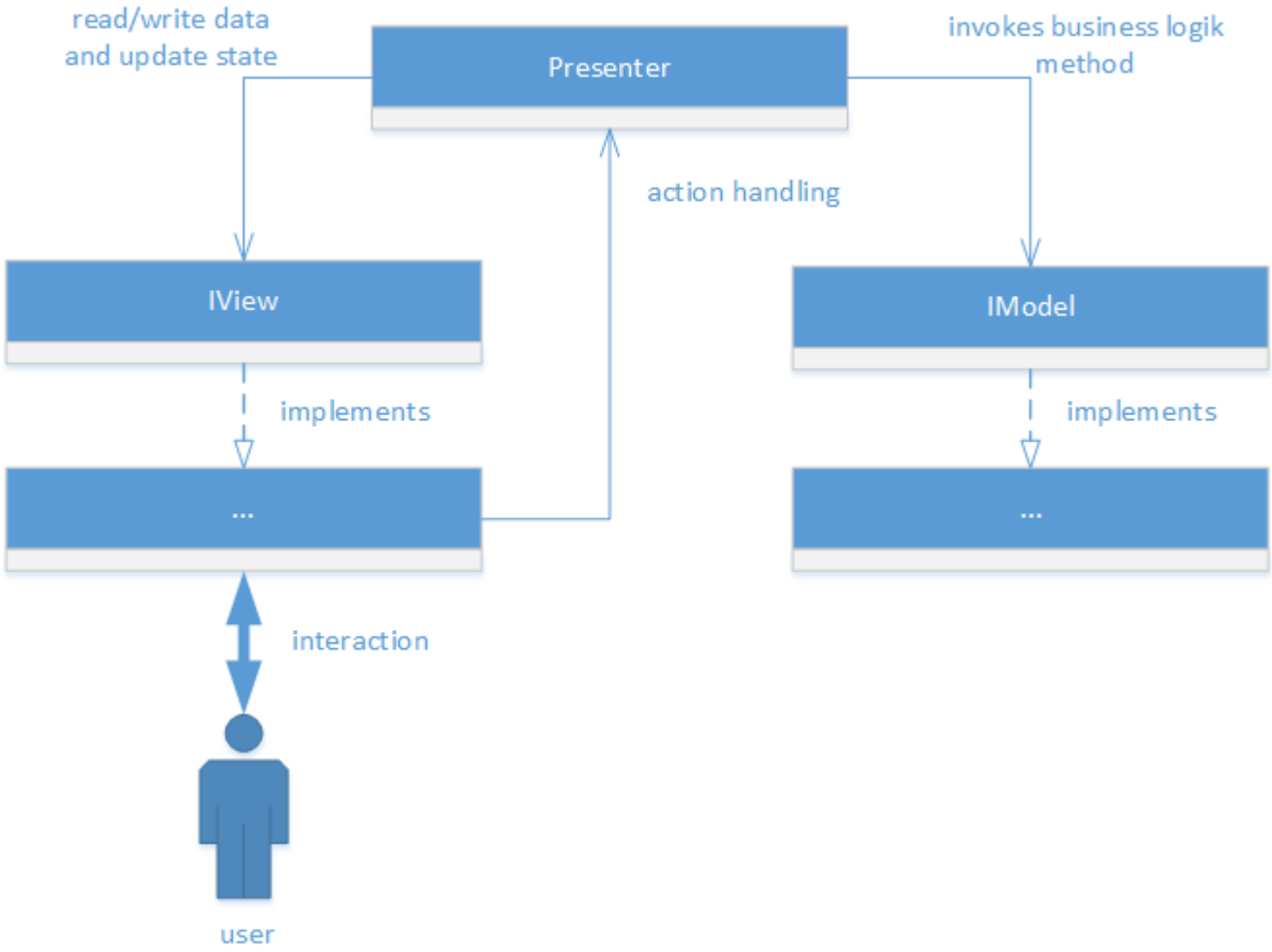
Modifikation des Models
Aktualisierung der View

Aufgabe: Die View (passiv)

Eingabeelemente + Ausgabeelemente

= Bereitstellung von GUI-Elementen für Interaktion mit Benutzer

Möglicher MVP-Aufbau



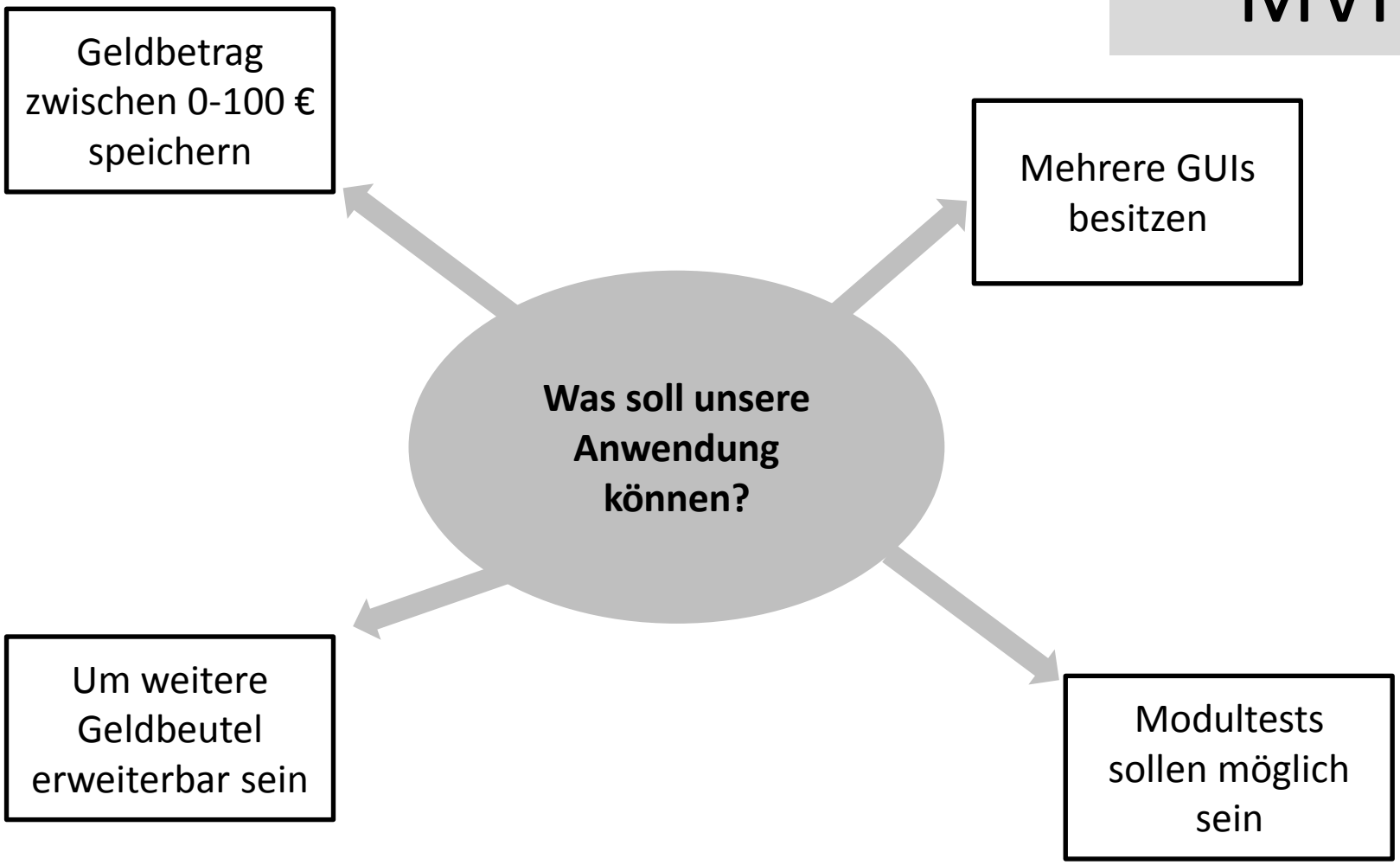
konkrete Problemstellung

Implementierung eines virtuellen Geldbeutels

Design Patterns: MVC und Derivate

FHWS

MVP



Wie gehen wir vor?

1. Interfaces festlegen

View

Model

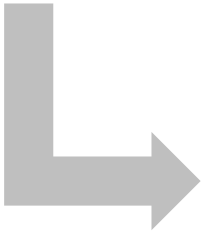
Was muss unsere View können?

- Aktuellen Geldbetrag anzeigen
- Betragsänderungen durch Benutzer entgegen nehmen

Bereitstellung von Methoden für das Action-Handling durch den Presenter

Was muss unser Model können?

- Geldbetrag speichern
- Aktuellen Geldbetrag vermitteln

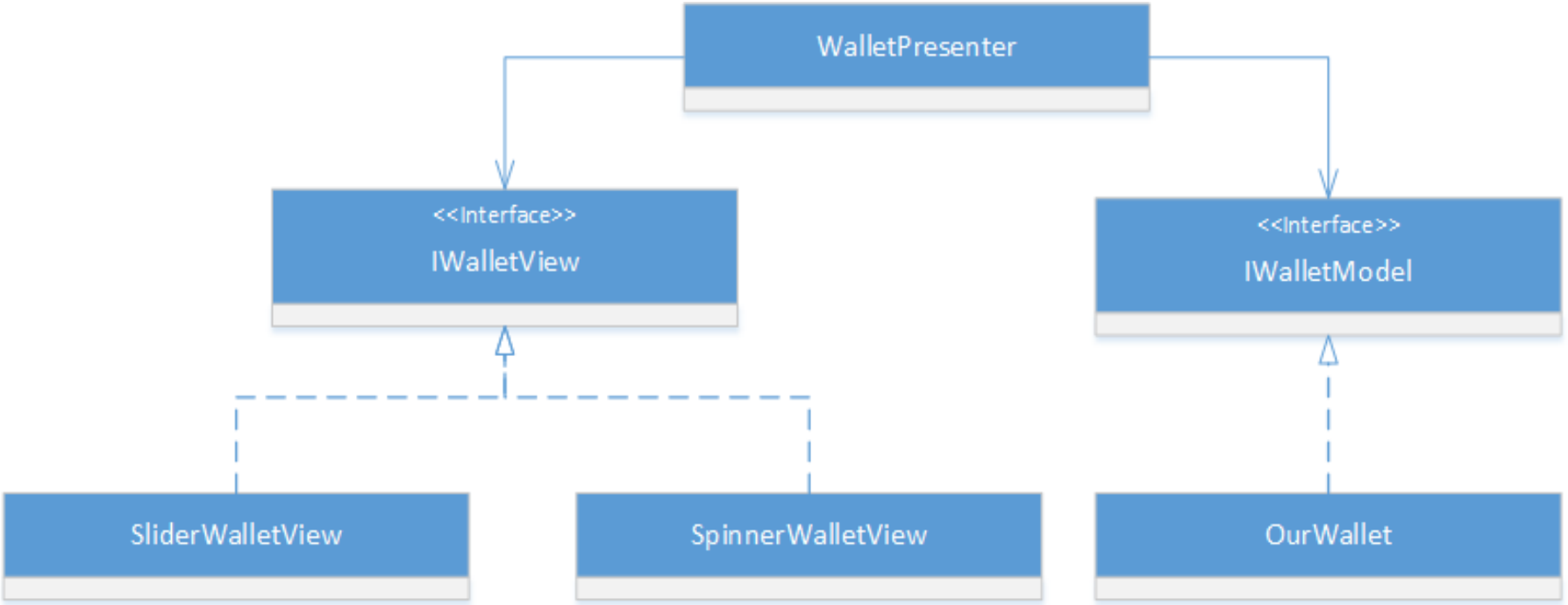


Benachrichtigung des Presenter über Betragsänderungen

Presenter definieren

- Auf Benutzereingaben durch die View reagieren, diese interpretieren und dem Model vermitteln -> Observer für View
- Anzeige des aktuellen Geldbetrags des Models in der View -> Observer für Model

Klassendiagramm einer Lösung



Beispielanwendung

Vorteile

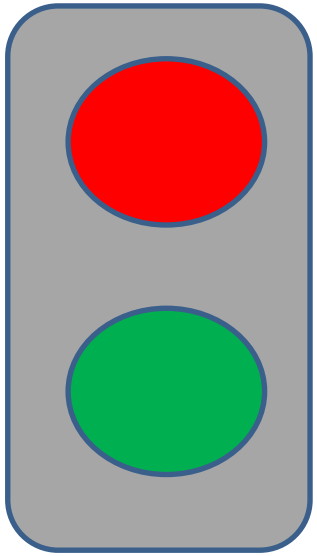
- Komponenten/Module sind austauschbar
- Testen der Anwendung vereinfacht
  (bessere Wartbarkeit)
- Einfach erweiterbar
- Klare Aufgabenteilung der Komponenten

Nachteile

- Hoher Design- und Implementierungsaufwand

- Nicht auf jedes Szenario anwendbar

(Subjektives) Fazit



kleine Anwendung

größere Anwendung

Möglicher Einsatz des Patterns muss bei der Analyse/Designs der Anwendung geprüft werden

Quellen

Paper von Taligant & IBM:
http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

Internetauftritt von Martin Fowler
http://www.martinfowler.com/

IEEE-Paper:
An Architecture and Implement Model for Model-View-Presenter Pattern