Praktikum – iOS-Entwicklung

Sommersemester 2018

Prof. Dr. Linnhoff-Popien

Markus Friedrich, Kyrill Schmid

Patterns & UI

Klassische Entwurfsmuster, MVC, UI Entwicklung mit XCode

Was sind Entwurfsmuster?

- Lösungsschablonen für wiederkehrende Entwurfsprobleme (Wikipedia)

- Wird meistens mit Codestrukturschablonen gleichgesetzt.
  - Aber: Muster existieren auch für andere Bereiche außerhalb der Softwareentwicklung.
  - Beispiel: HCI, aber auch fachfremd: Architektur, Maschinenbau

- Fokus: Entwurfsmuster in der Software Entwicklung
  - Wichtig: Die Beschreibung (der meisten) Muster ist **unabhängig** von einer bestimmten Programmiersprache.

- Wir unterscheiden für unsere Zwecke:
  - **Architekturmuster**: Beschreiben die Grobarchitektur der gesamten Anwendung
  - (klassische) **Entwurfsmuster**: Beschreiben einzelne Teilelemente der Architektur in größerem Detail

Entwurfsmuster - Kategorien

Man unterscheidet verschiedene Kategorien von (OOP) Entwurfsmustern:

- **Erzeugungsmuster:** Trennung der (komplexen) Erzeugung von Objekten von Ihrer Repräsentation
- **Strukturmuster:** Schablonen zur Modellierung von Beziehungen zwischen Objekten
- **Verhaltensmuster:** Muster zur Modellierung des Verhaltens von Objekten

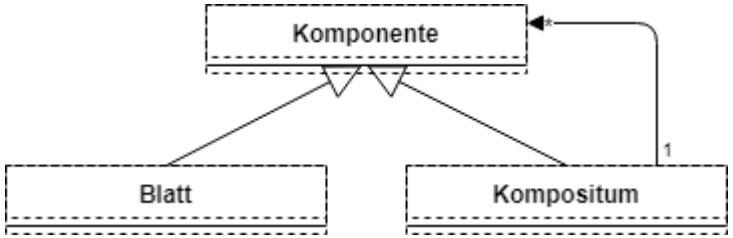
Diese Unterscheidung hat ihren Ursprung im Buch „Design Patterns – Elements of Reusable Object-Oriented Software“ der „Gang if Four“ (GOF), 1995

Neue Kategorien wurden im Laufe der Zeit hinzugefügt.

Beispiele

Strukturmuster „Kompositum“

Ziel: Einheitliche Behandlung von Objekten und Kompositionen von Objekten.



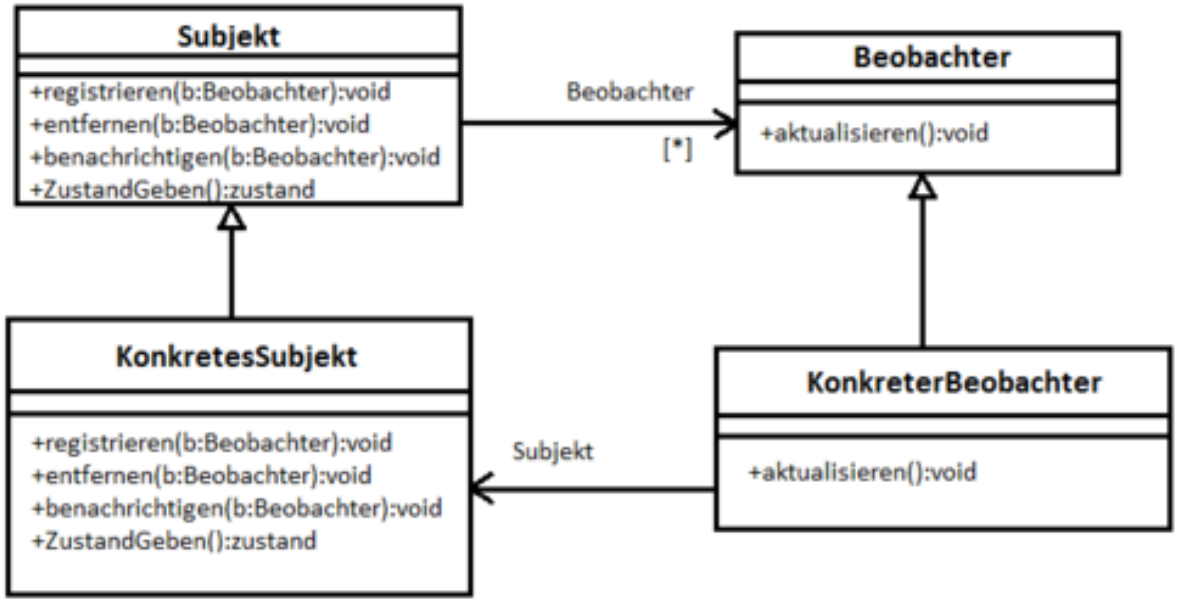
In iOS / Swift:

- Views
- …

Beispiele

Verhaltensmuster „Beobachter“ (Observer Pattern)

Ziel: Benachrichtigungen zwischen Objekten mit loser Kopplung



In iOS / Swift:

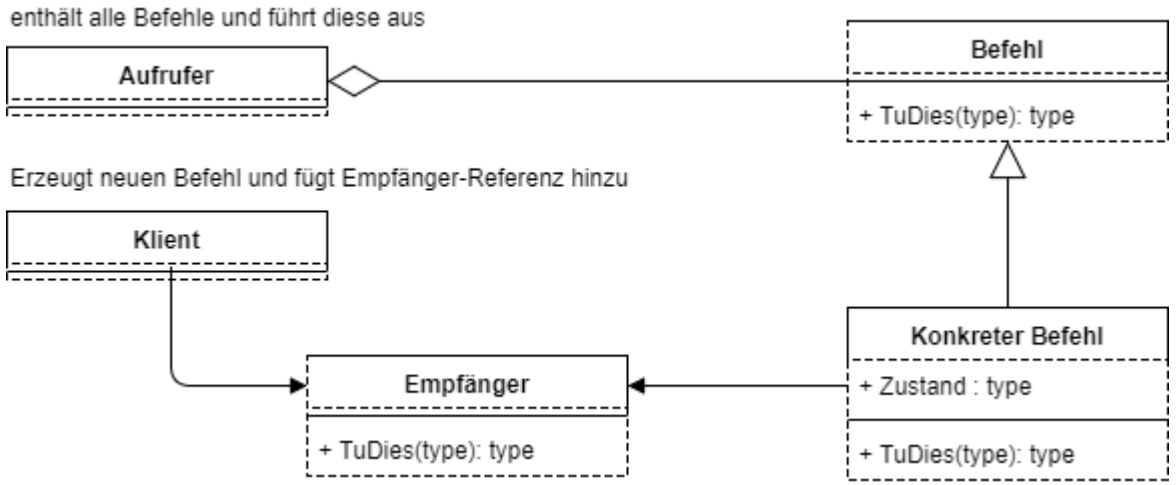
- Notification Center
- Key-Value-Observing (KVO)
- [Delegate]

Quelle: https://de.wikipedia.org/wiki/Datei:Beobachterentwurfsmuster.png

Beispiele

Verhaltensmuster „Kommando“

Ziel: Kapselung von Methodenaufrufen in Objekte => Warteschlangen



In iOS / Swift:

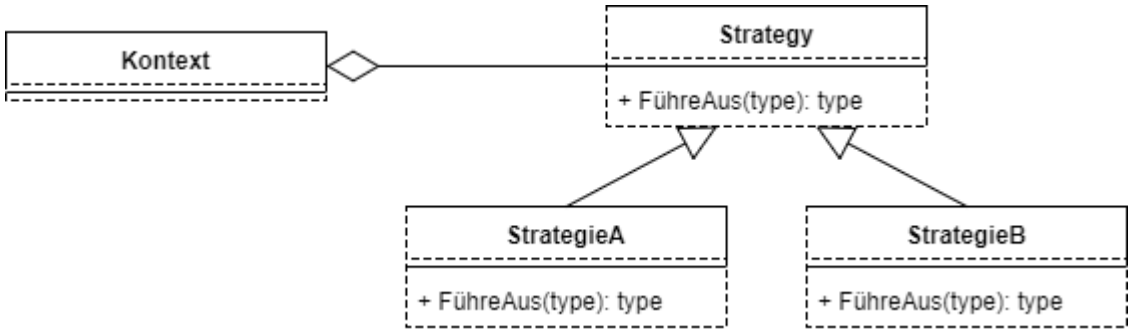
- Views: Target-Action Mechanismus

Beispiele

Verhaltensmuster „Strategie“

Ziel: Beschreibung einer Familie von Algorithmen

Interessant: Verbindung mit Fabrikmuster



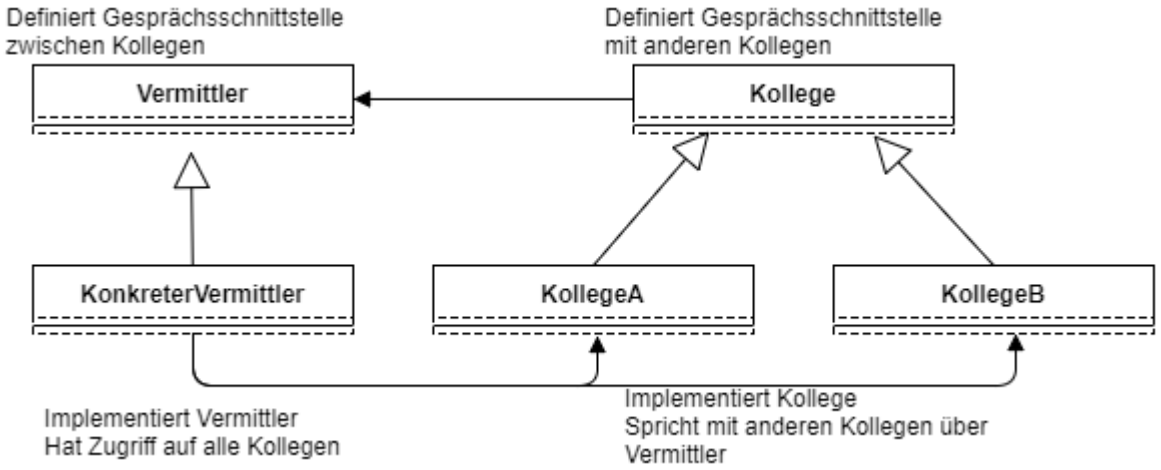
In iOS / Swift:

- View Controller

Beispiele

Verhaltensmuster „Vermittler“

Ziel: Steuerung der Kooperation verschiedener Objekte



In iOS / Swift:

- View Controller

Beobachter – Notification Center

- Mechanismus zur Verteilung von Ereignissen an registrierte Beobachter.
- Wie funktioniert das?

```
// Get application-wide NotificationCenter Singleton.
let nc = NotificationCenter.default

// Register Observer
nc.addObserver(forName:Notification.Name(rawValue:"MyNotification"),
  object:nil, queue:nil){
  notification in
// Handle notification
}

// Notify
nc.post(name:Notification.Name(rawValue:"MyNotification"),
 object: nil,
 userInfo: ["message":"Hello there!", "date":Date()])
```

Quelle: http://dev.iachieved.it/iachievedit/notifications-and-userinfo-with-swift-3-0/

Architekturmuster - Kategorien

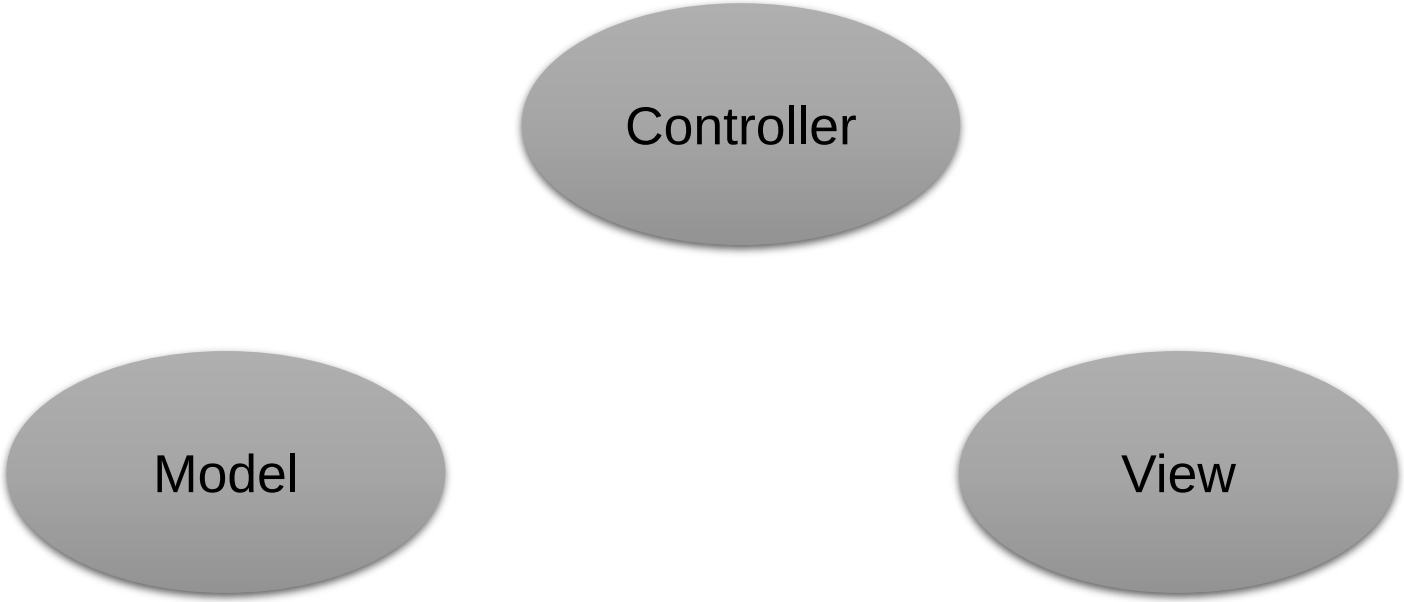
Man unterscheidet verschiedene Kategorien von Architekturmustern:

- **Adaptive Systeme:** Muster für Systeme mit Fokus auf Erweiterbarkeit
- **Datengetriebene Systeme:** Muster für datengetriebene Applikationen
- **Verteilte Systeme:** Muster zur Orchestrierung und Kommunikation in verteilten Anwendungen
- **Interaktive Systeme:** Muster für die Strukturierung von HCI Anwendungen
- **…**

Beispiele

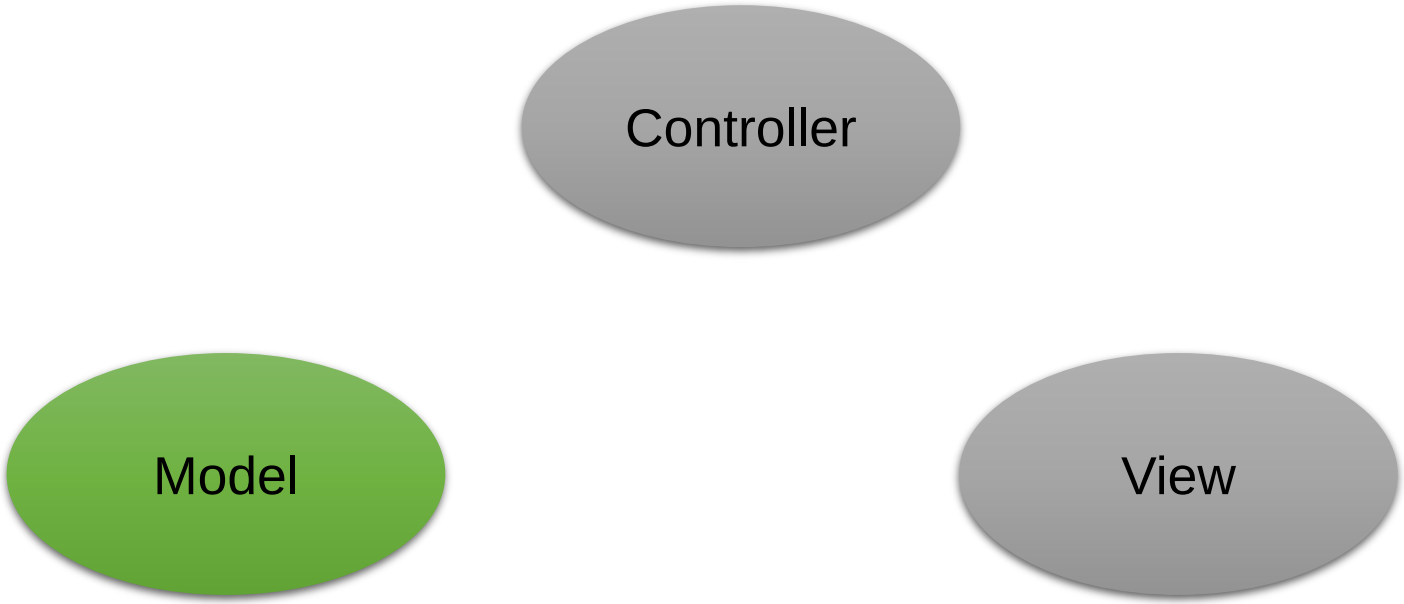
Muster interaktiver Systeme: „Model-View-Controller“ (MVC)

- Aufteilung von Objekten in drei Gruppen:



MVC - **Model**

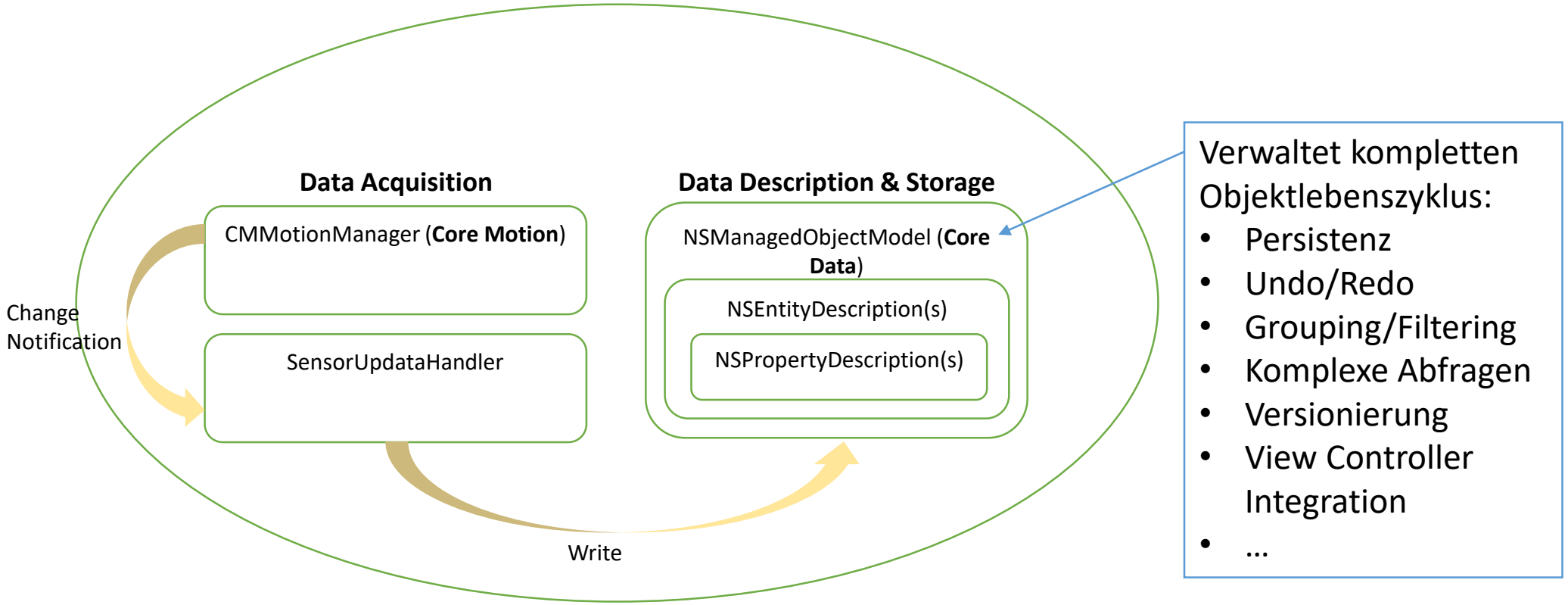
- Enthält **domänenspezifisches** Datenmodell.
- Wichtig: Datenmodell ist **unabhängig** von späterer UI-Darstellung!



MVC - **Model**

Persistenzalternativen:
- Firebase Cloud Storage
- SQLite eingebettete, lokale Datenbank

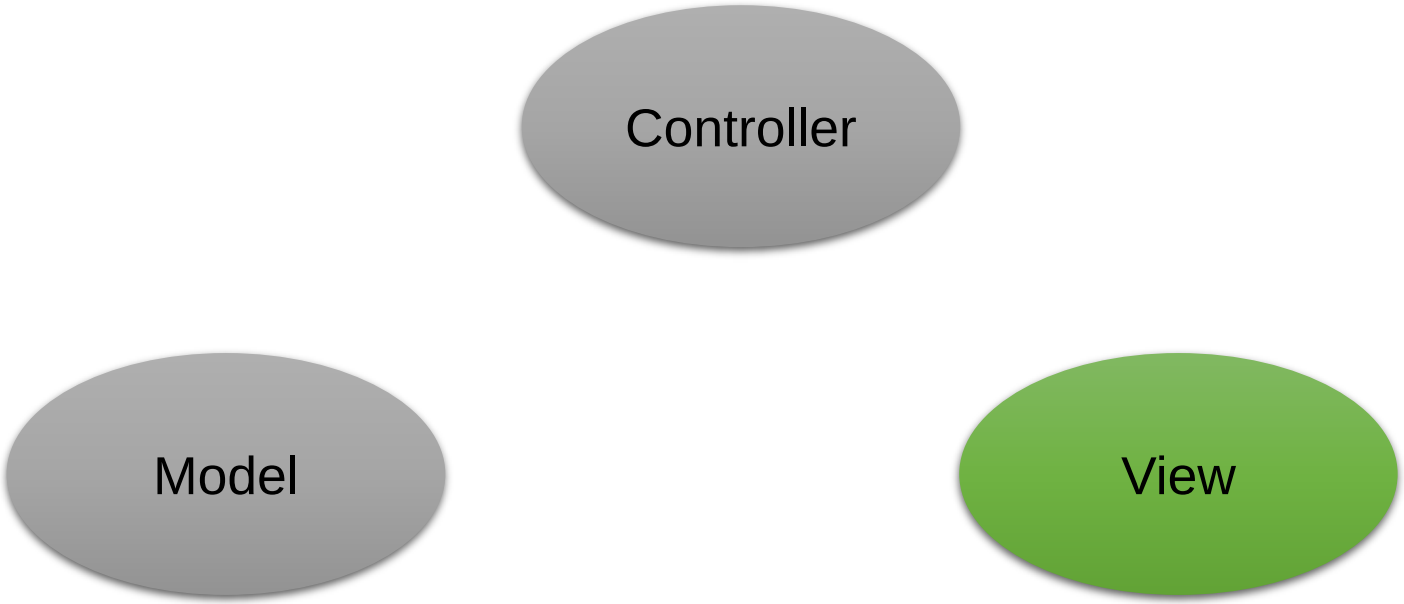
Models in iOS



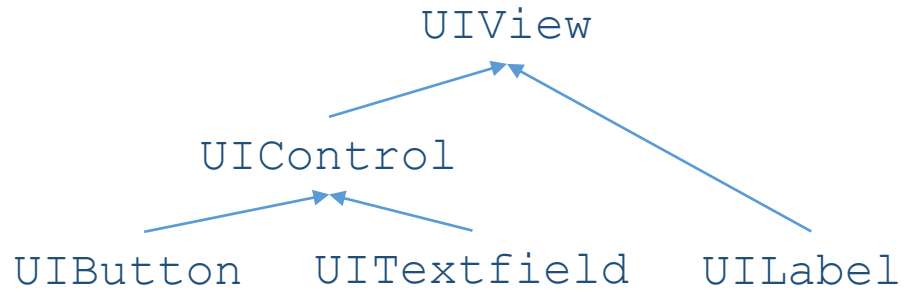
Verwaltet kompletten Objektlebenszyklus:
- Persistenz
- Undo/Redo
- Grouping/Filtering
- Komplexe Abfragen
- Versionierung
- View Controller Integration
- …

MVC - **View**

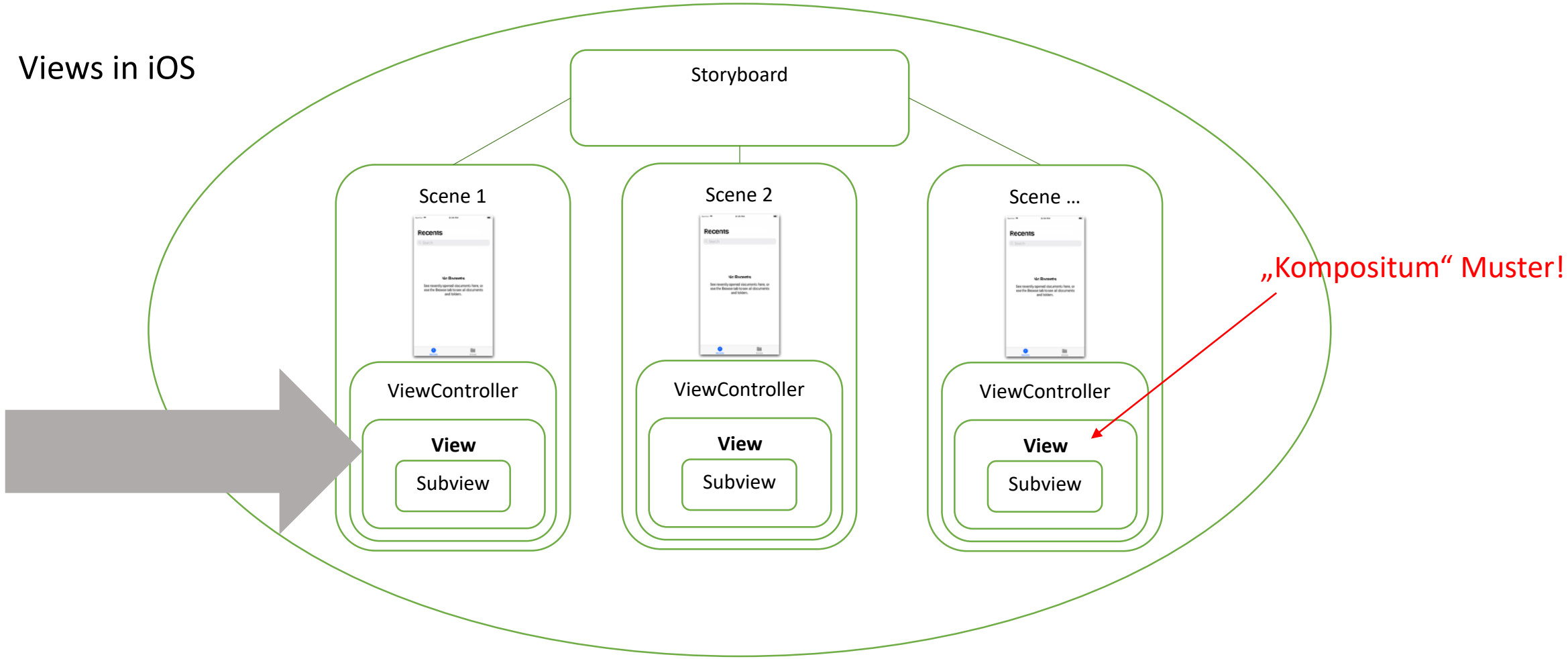
- UI-Darstellung der im Modell definierten Daten
- Benutzer interagiert mit den Daten, es findet aber **keine Verarbeitung** statt!



MVC - **View**

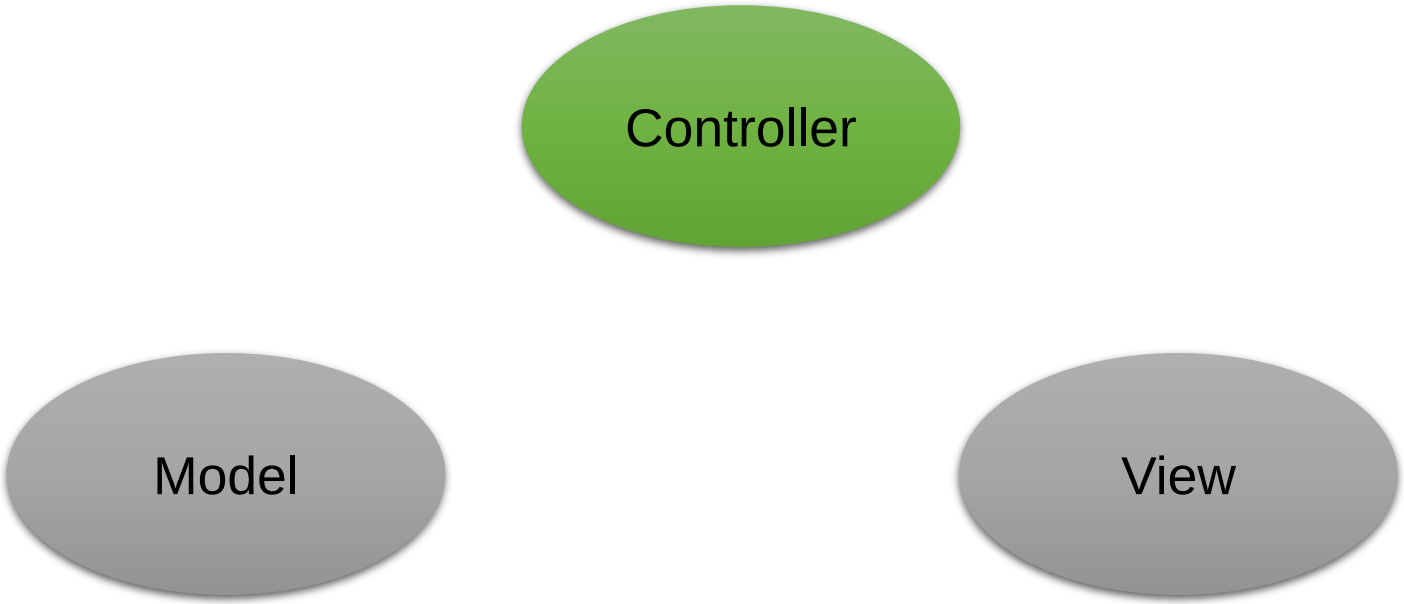


Views in iOS



MVC - **Controller**

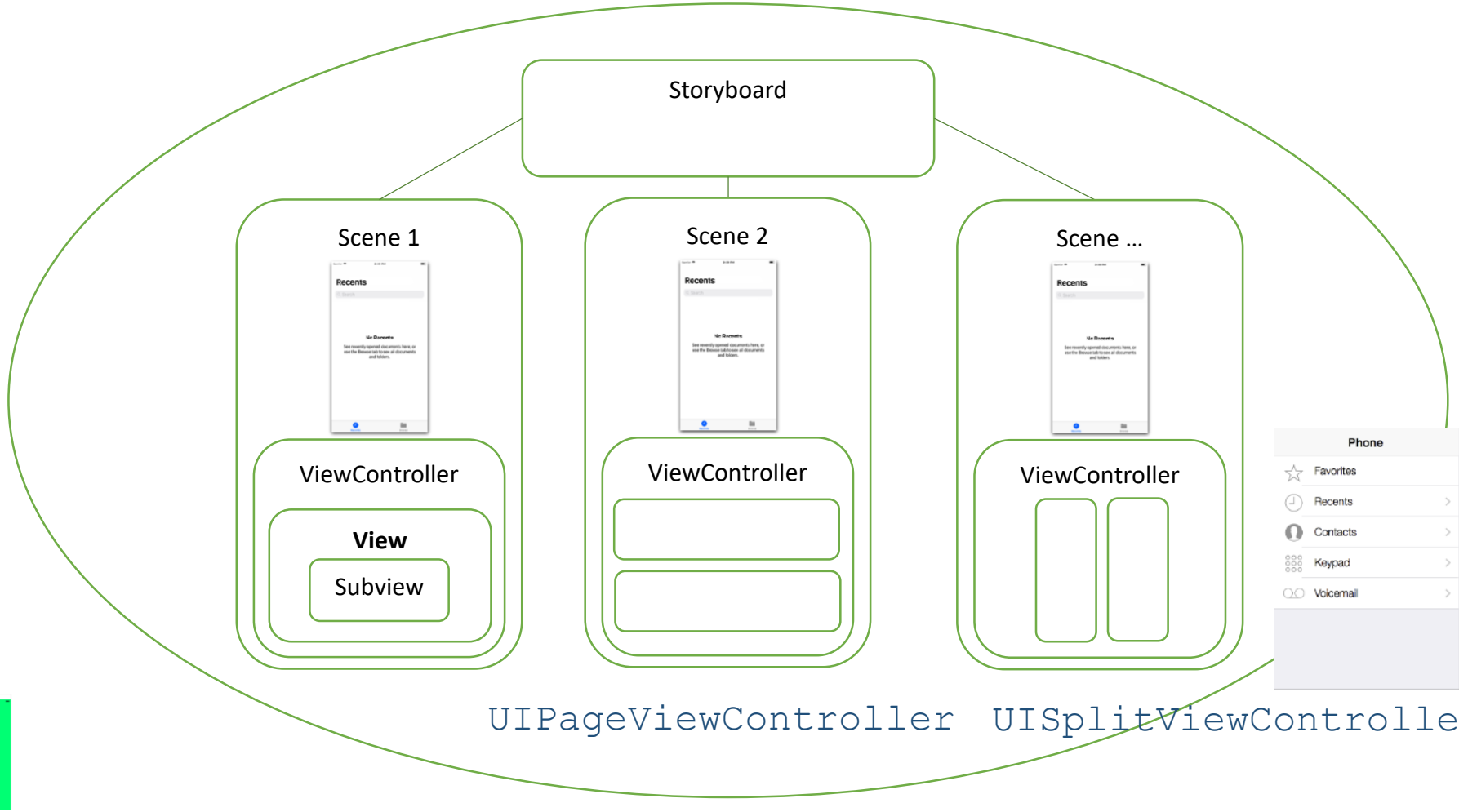
- Vermittlung zwischen Datenmodell und Darstellung (Logik der Darstellung)
  - Auswertung von Benutzerinteraktionen (View)
  - Manipulation von Daten (Model)
  - Zu jeder View existiert meistens genau ein Controller

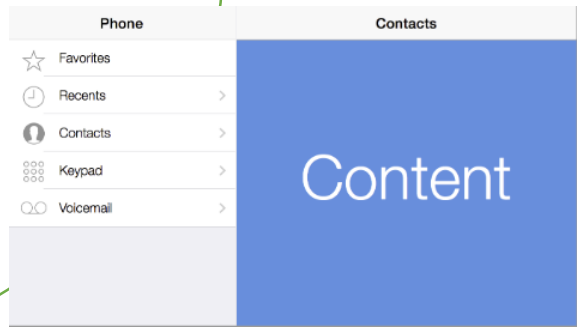


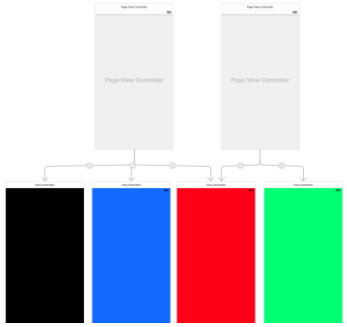
MVC - **Controller**

UIViewController

UITableViewController IOCollectionViewController

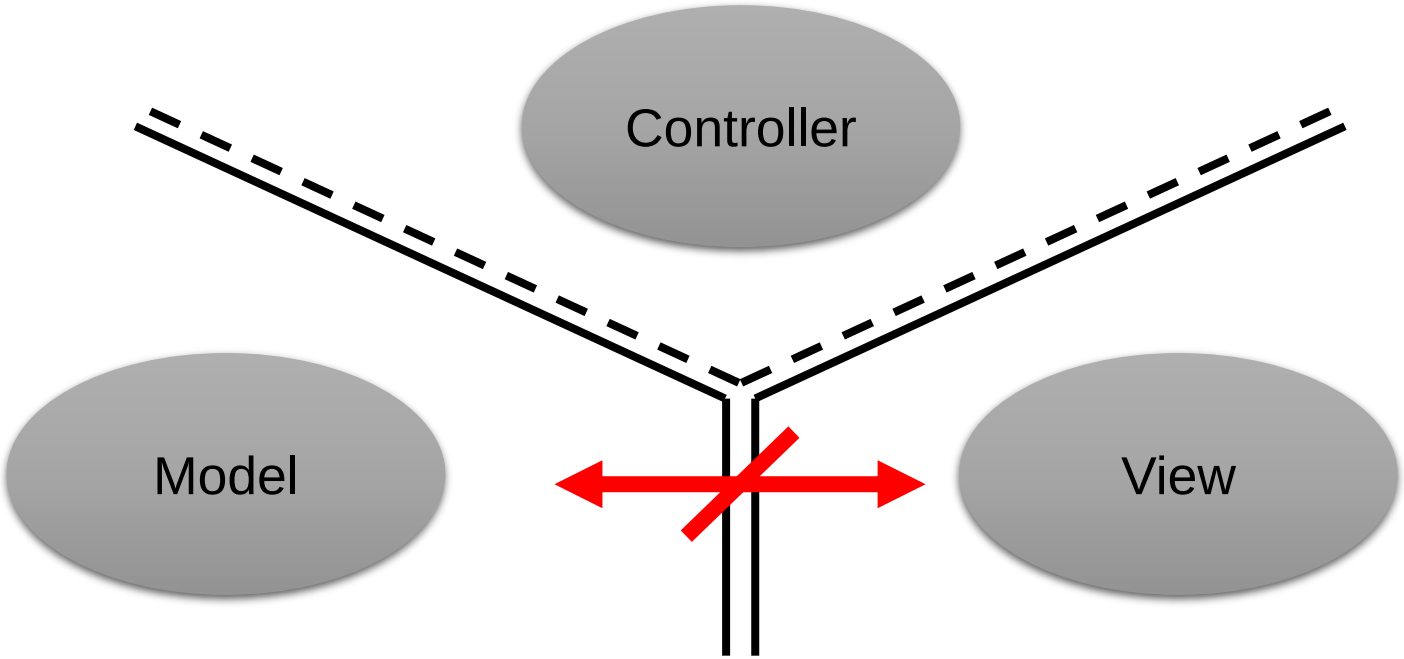






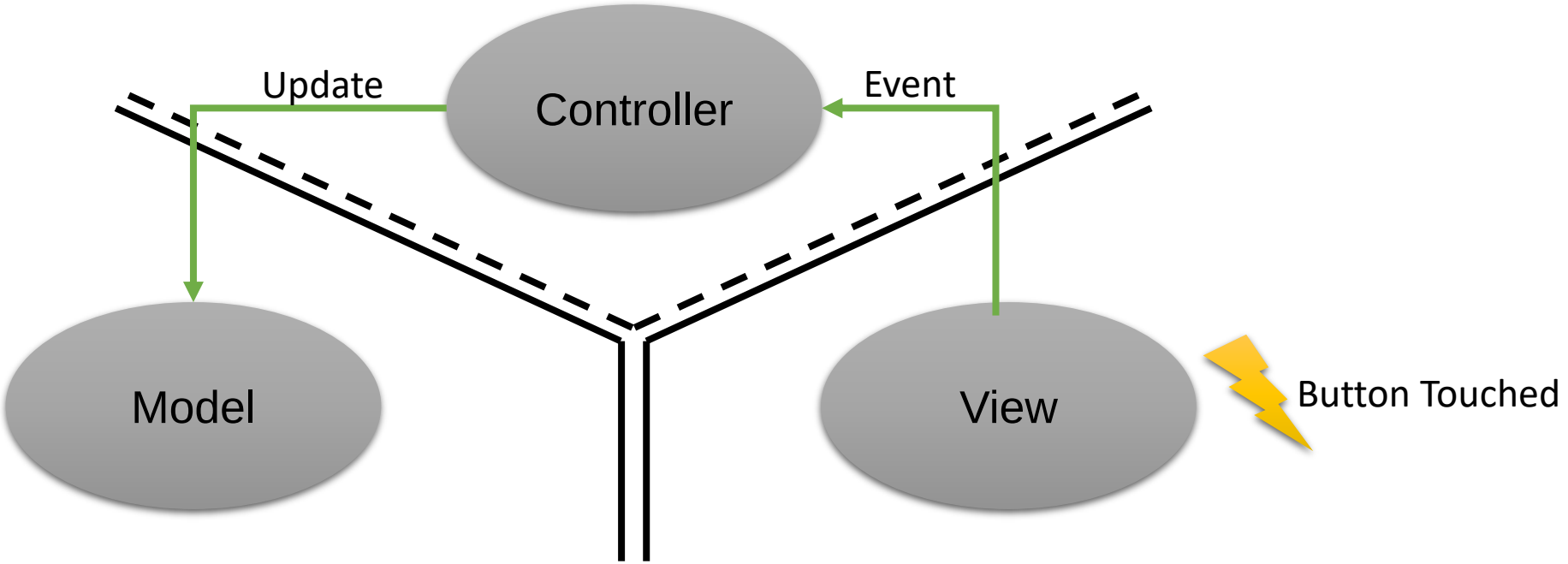
MVC - **Kommunikationswege**

- Wer kommuniziert mit wem?
- Wichtig: **Kein** Kommunikationspfad zwischen **Model** und **View**!



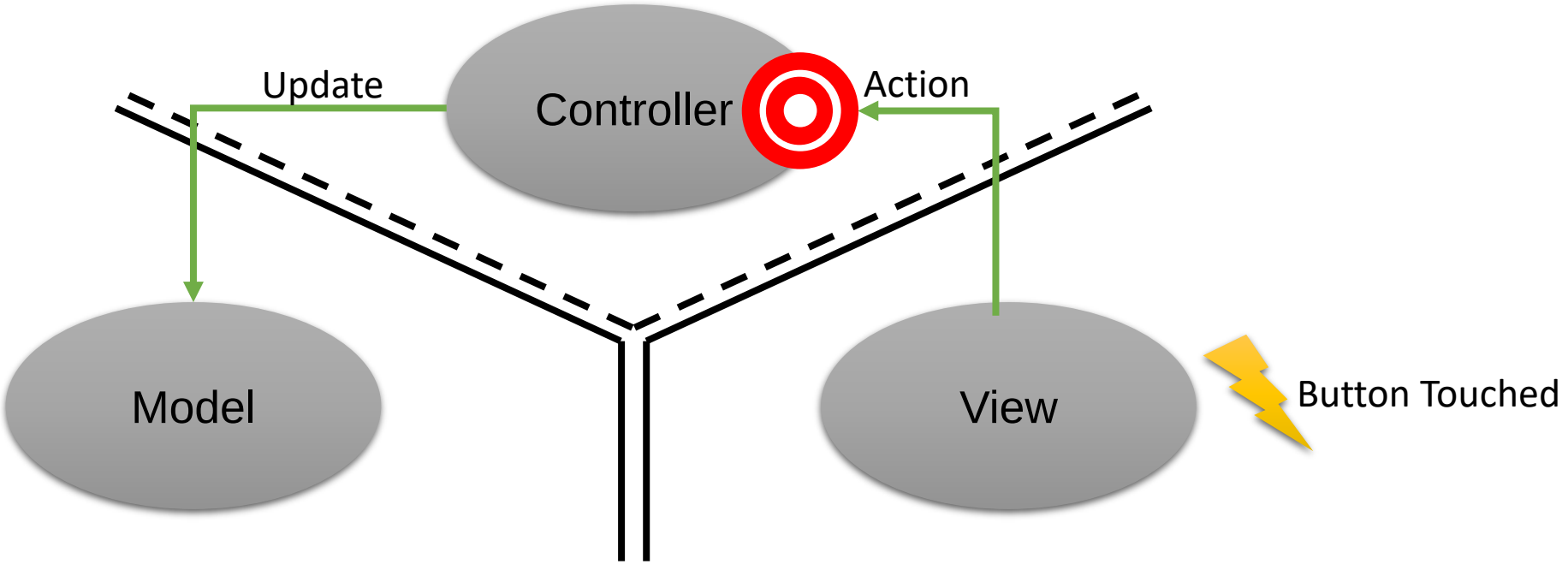
MVC – Kommunikationswege I

- **Benutzerinteraktion:**
  - Kommunikation von Ereignissen erfolgt als Aktion
  - Controller aktualisiert das Modell **nach Prüfung der Eingabe.**



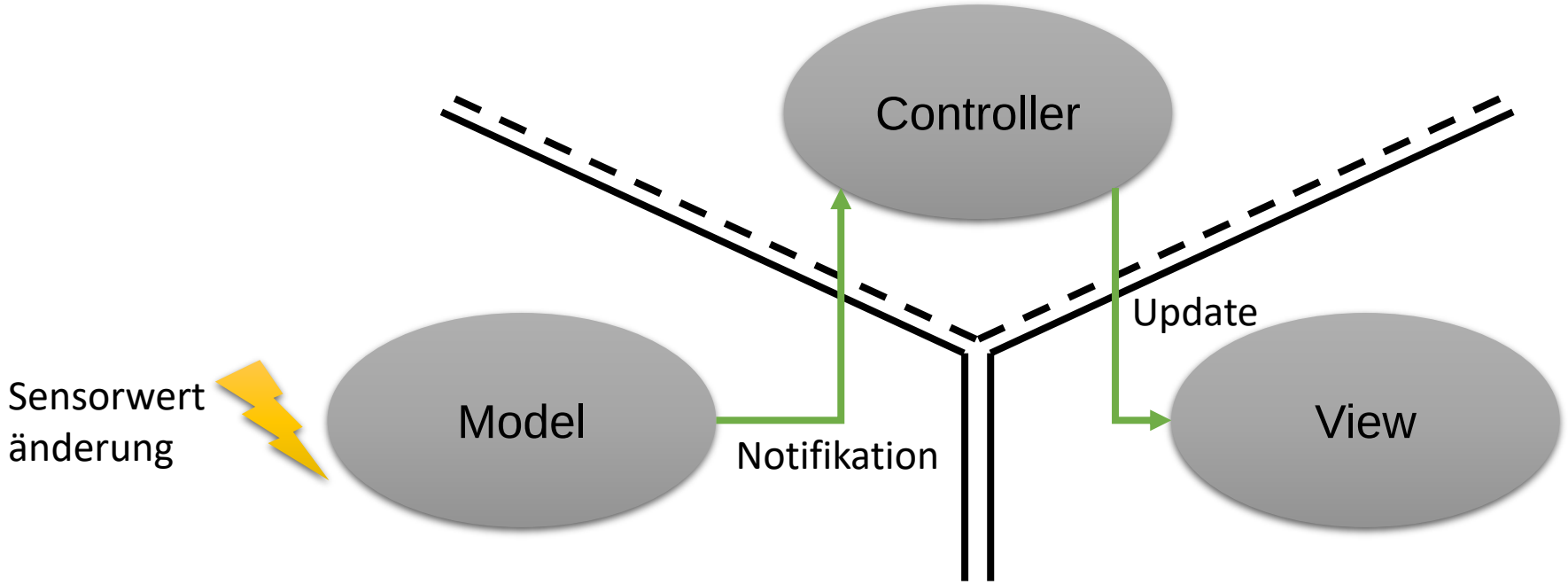
MVC – Kommunikationswege II

- **Benutzerinteraktion in iOS:**
  - Target-Action Mechanismus
  - UIControl.addTarget(targetObject, action, controlEvents)
  - targetObject ist für gewöhnlich der assoziierte ViewController



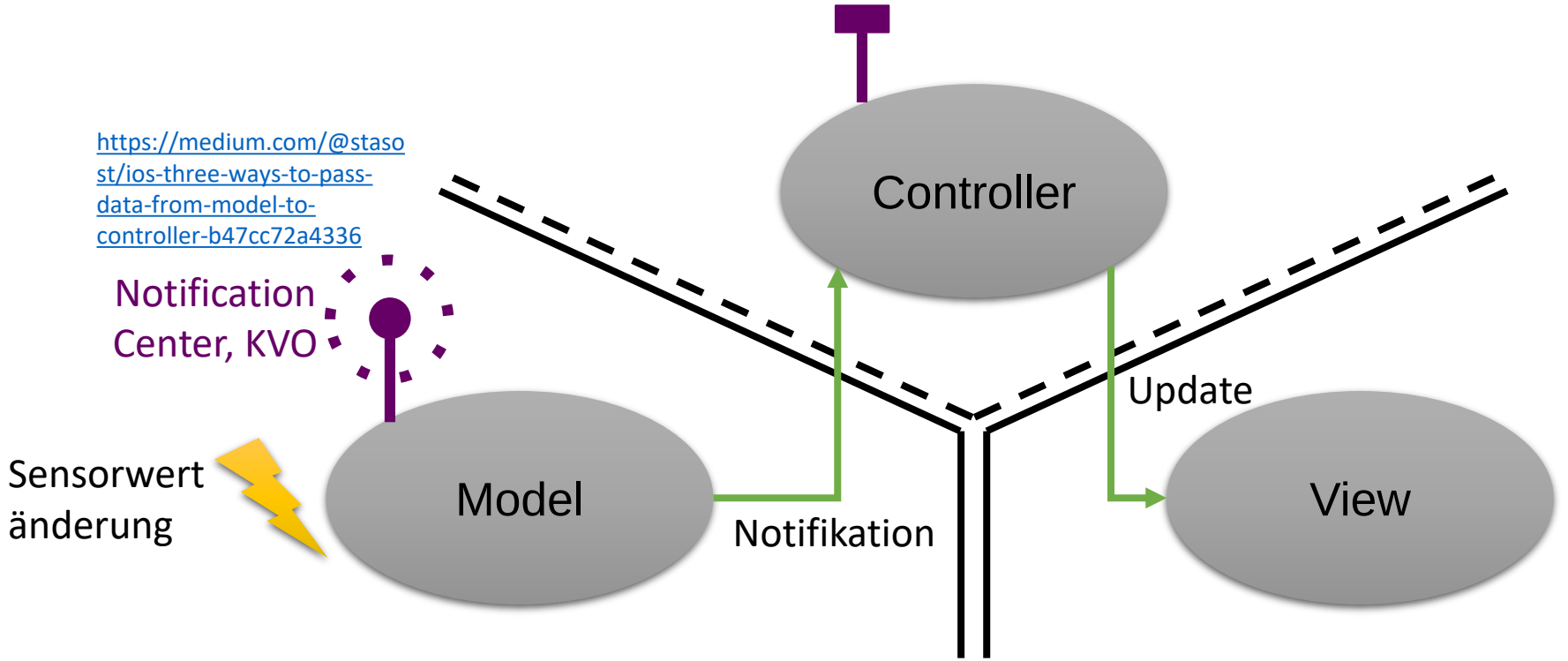
MVC – Kommunikationswege III

- **Datenaktualisierung:**
  - Model benachrichtigt nach Datenänderung den **Controller**
  - **Controller** aktualisiert **View**



MVC – Kommunikationswege IV

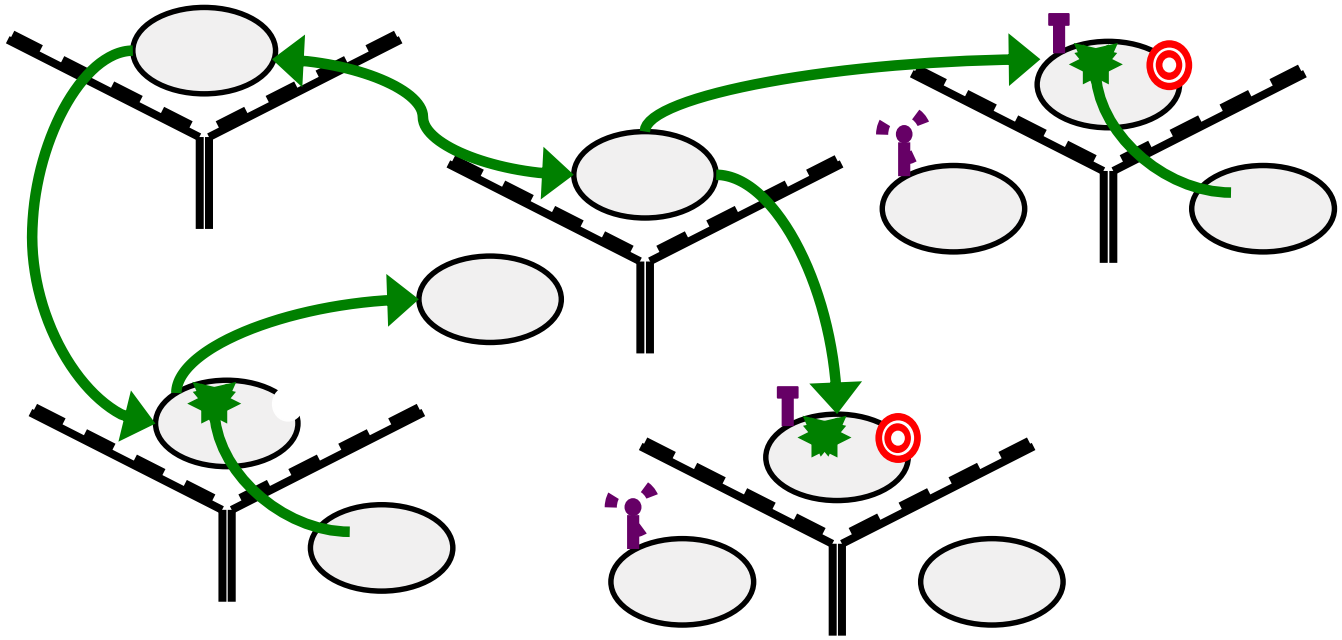
- **Datenaktualisierung in iOS:**
  - Notifikation kann durch Muster aus der Beobachterkategorie erfolgen(Delegation, Notification Center, KVO)
  - Verwendung von **Delegation** ist der am häufigsten verwendete Weg.



Delegation:
- Controller Implementiert Protokoll
- Controller übergibt sich selbst als Delegat an Model

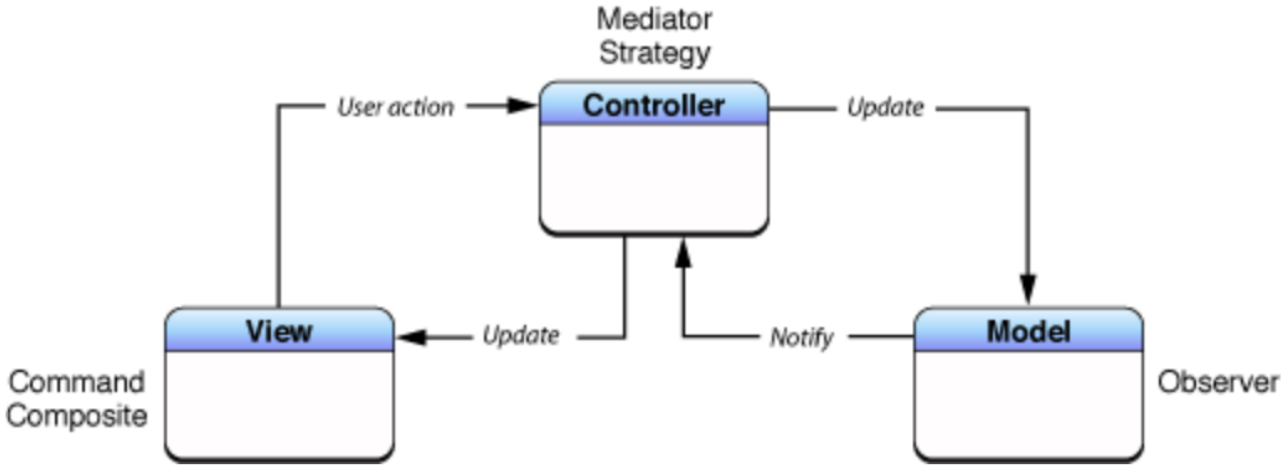
MVC

Komplexe Programme entstehen durch die Kombination mehrerer MVC-Gruppen.



MVC als Zusammensetzung von Entwurfsmustern

- Kommando:
  - Views‘ Target-Action Mechanismus
- Kompositum:
  - Views sind ineinander geschachtelt
- Vermittler:
  - View Controller vermittelt den Datenfluss zwischen Model und View
- Strategie:
  - UI Verhalten wird von View Controller implementiert (Delegate)
- Beobachter:
  - Model benachrichtigt View Controller bei Änderungen



Quelle: https://developer.apple.com/library/content/documentation/General/Conceptual/CocoaEncyclopedia/Model-View-Controller/Model-View-Controller.html

Gefahren

- Entwurfsmuster sind kein Allheilmittel!
  - Codequalität != # Muster pro 1000 Zeilen Code
  - Hilft nicht gegen falsche Wahl von Algorithmen und/oder Datenstrukturen
  - Muster passen nicht auf jedes Problem

- Antimuster!

Interessante Links

- Entwurfsmuster in Swift
  - https://github.com/ochococo/Design-Patterns-In-Swift => Repo mit Implementierungen aller klassischen Muster
- MVC
  - https://medium.com/ios-os-x-development/ios-architecture-patterns-ecba4c38de52 => Von MVC über MVP zu MVVM
- Object Modeling mit dem Core Data framework
  - https://developer.apple.com/library/content/documentation/General/Conceptual/CocoaEncyclopedia/ObjectModeling/ObjectModeling.html#//apple_ref/doc/uid/TP40010810-CH15-SW1 => Einführung