




# Лекция 3.0

## Architecture

Как правильно готовить брюкву



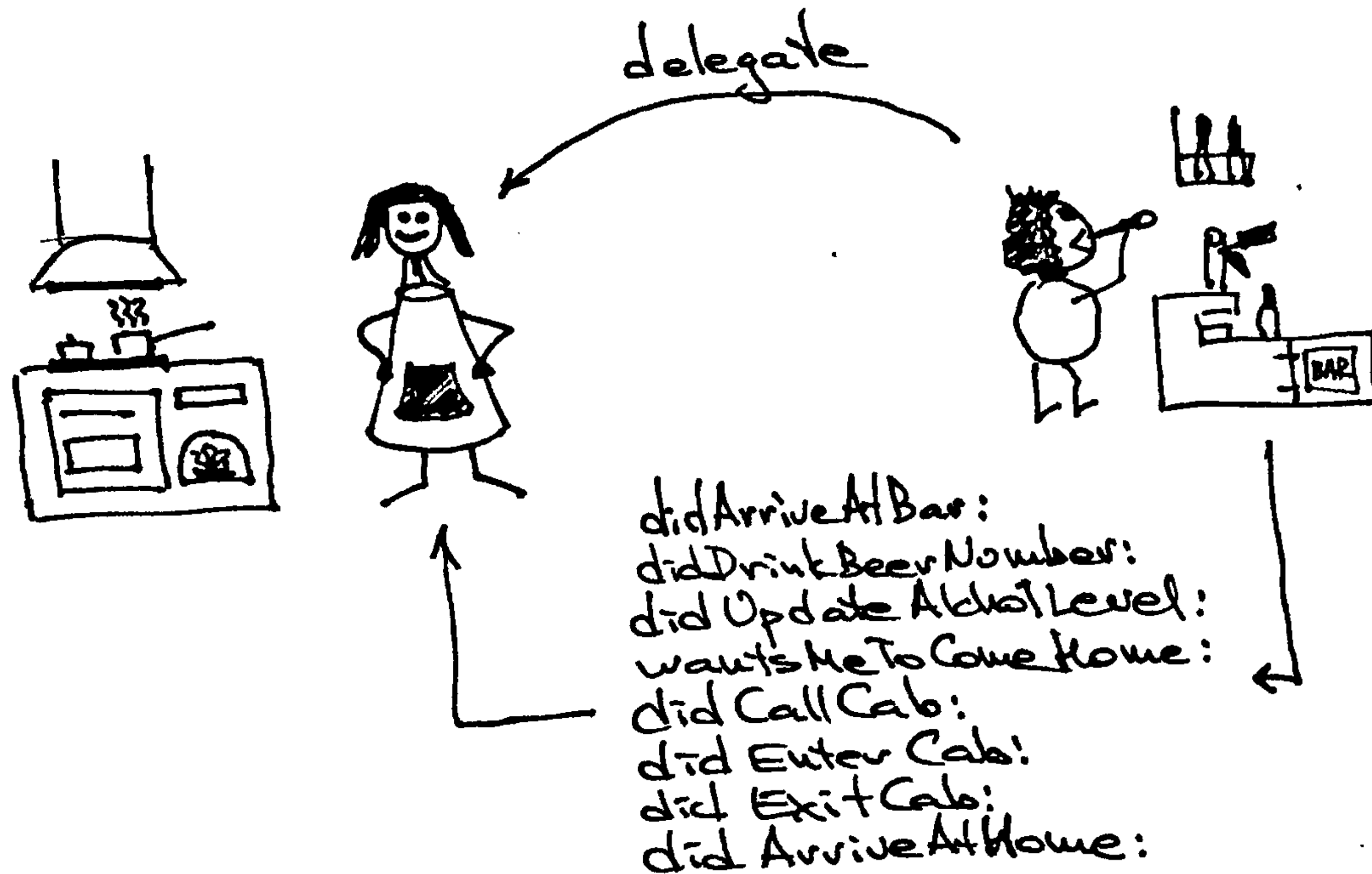


# Архитектура - очень спорная тема

Я вам расскажу одно из видений, как это должно быть

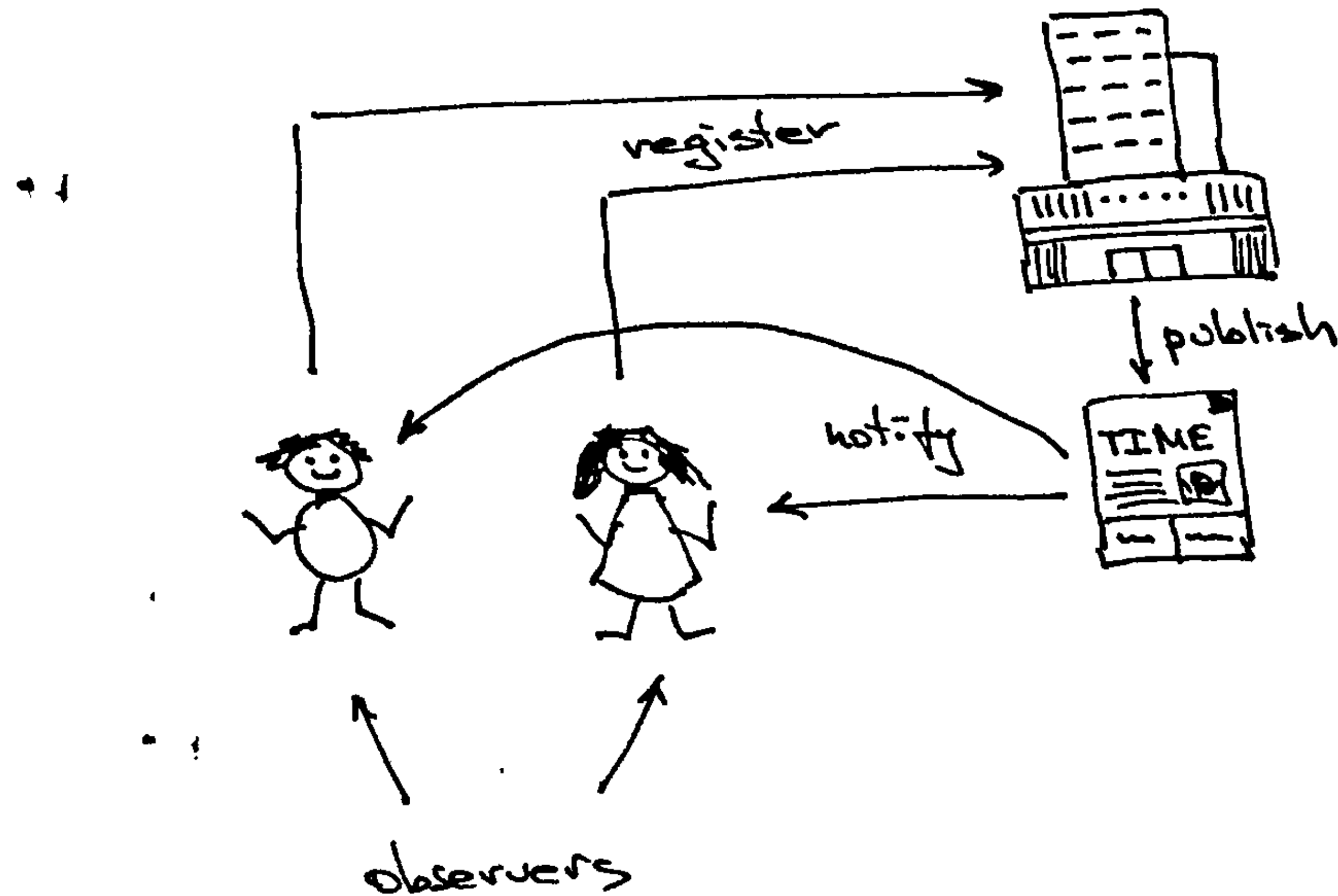
# Паттерн - Делегат

# Паттерн - Делегат



# Паттерн - Наблюдатель

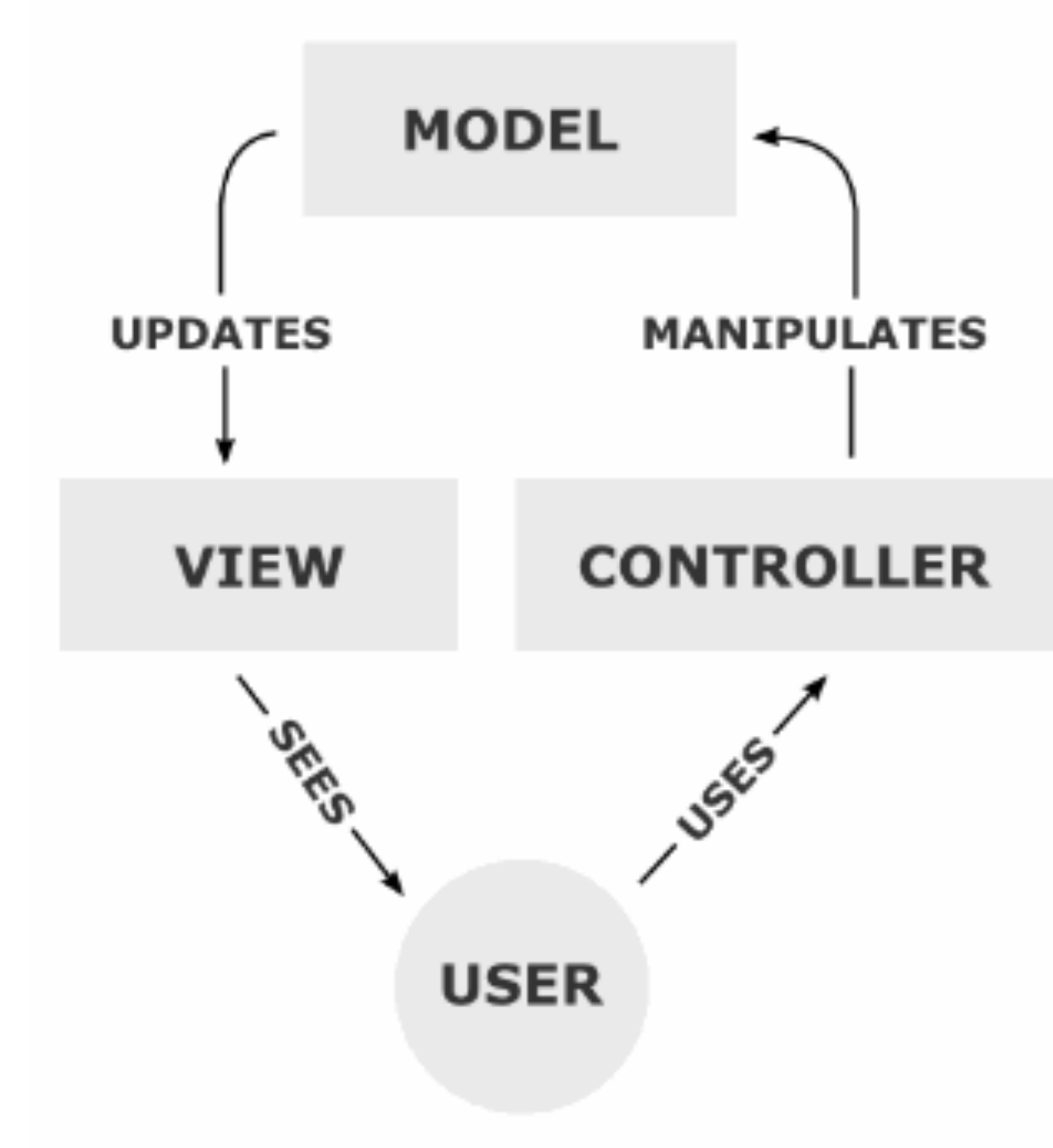
# Паттерн - Наблюдатель



# Паттерн - MVC

Model View Controller - Модель Представление Контроллер

# Паттерн - MVC



В Wikipedia можно увидеть такую картину, но это не совсем так



# MVC - Хорошие статьи почитать

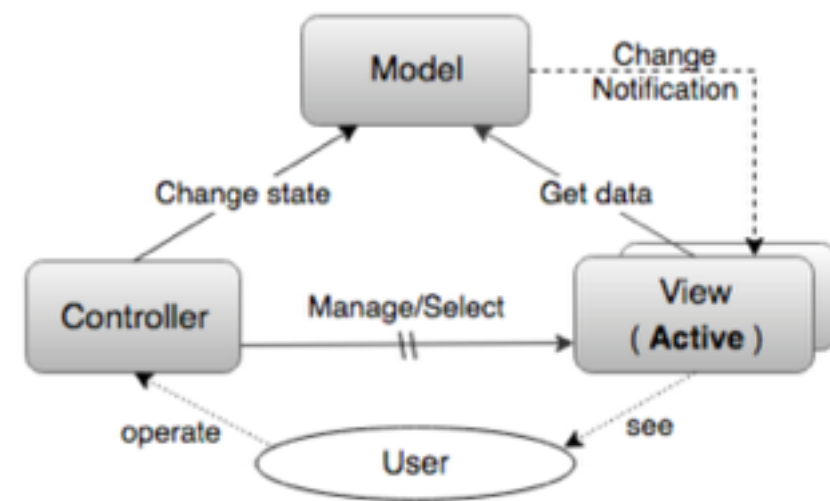


Про разновидности MVC

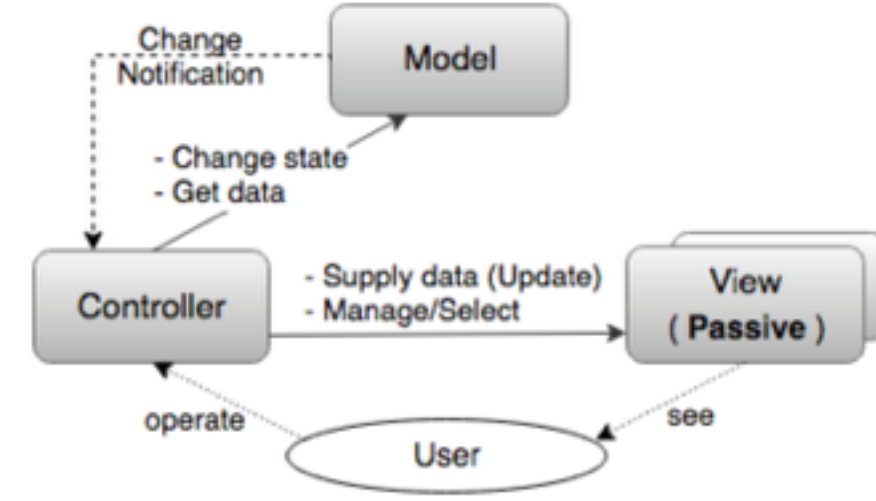


MVC, MVP, MVVM, VIPER ...

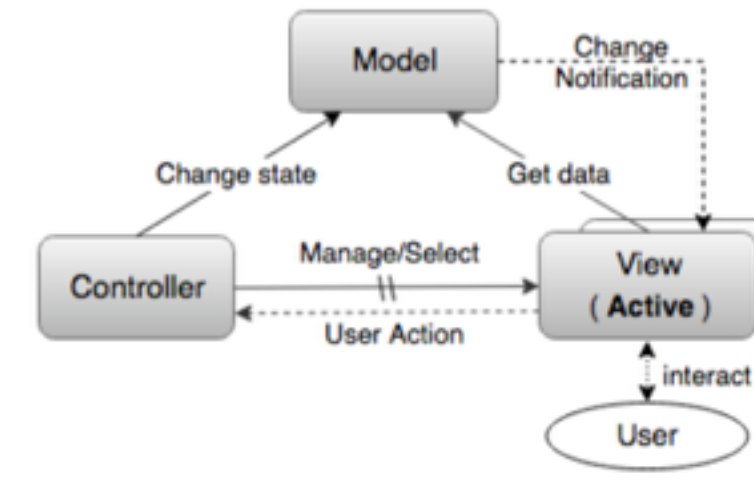
# MVC - типы



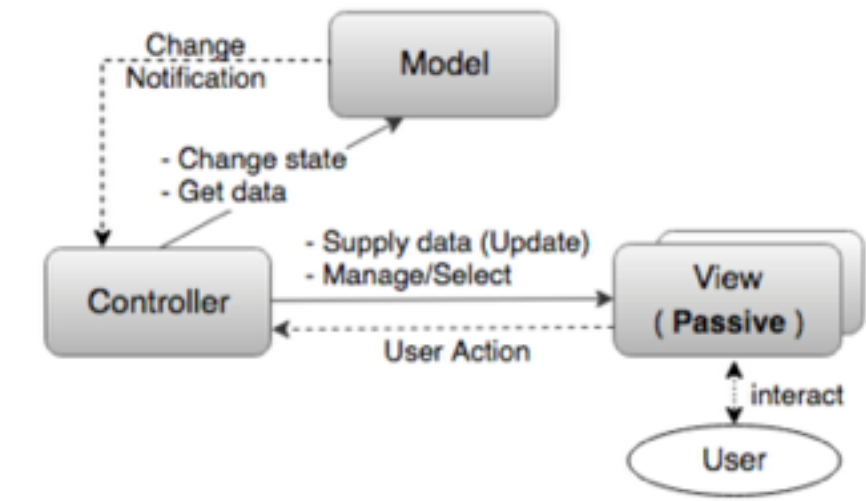
MVC 1



MVC 2

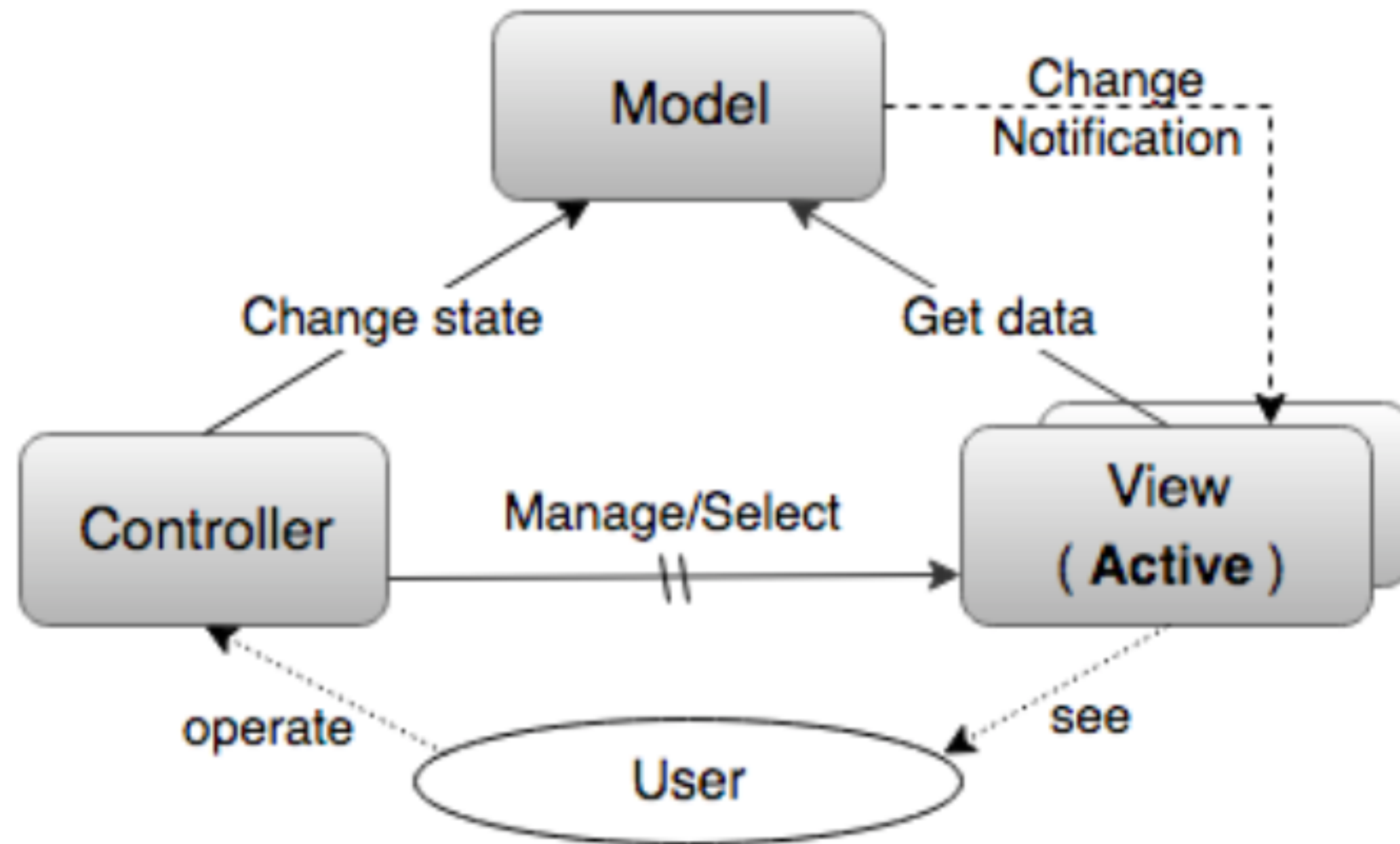


MVC 3

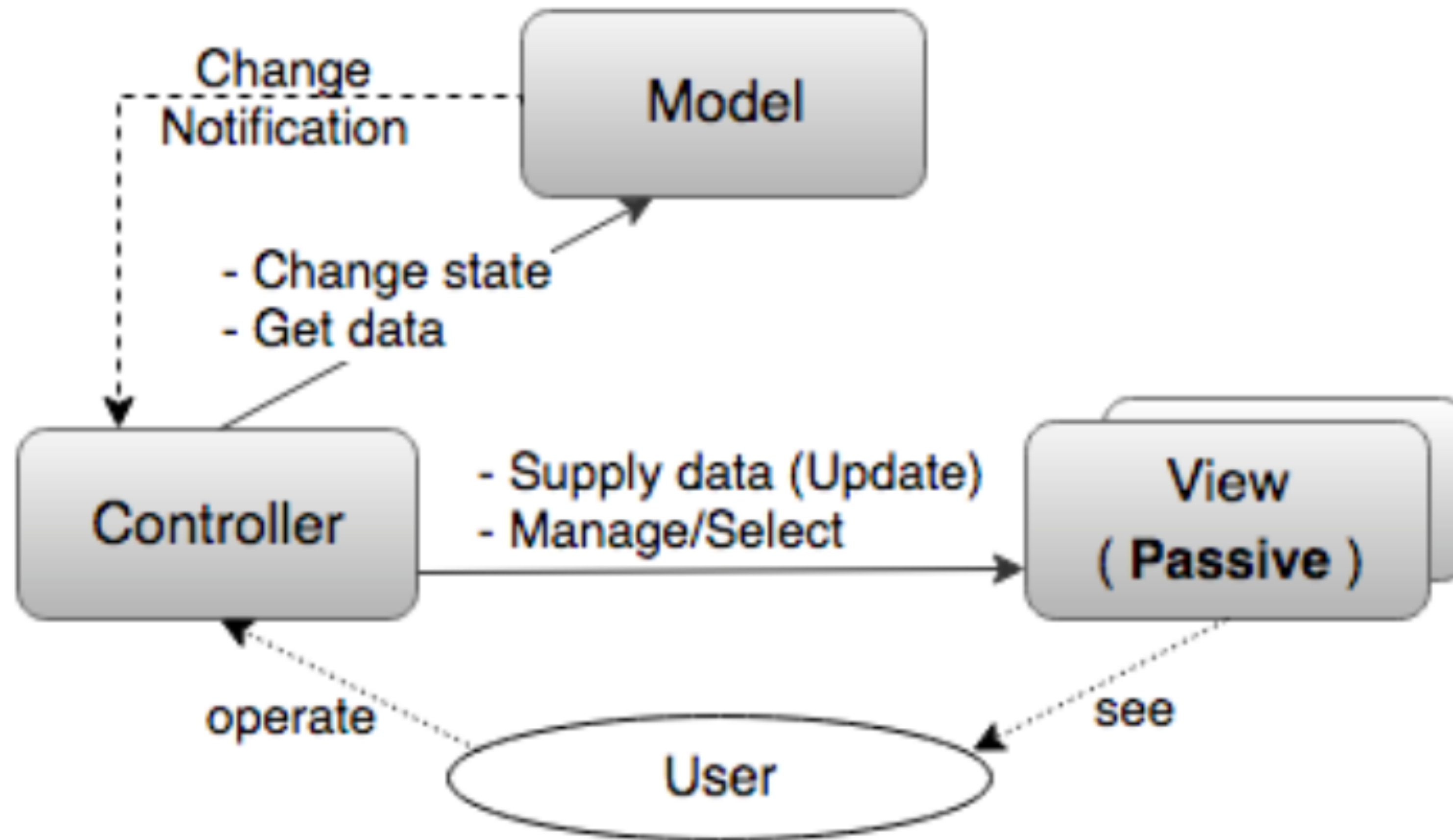


MVP

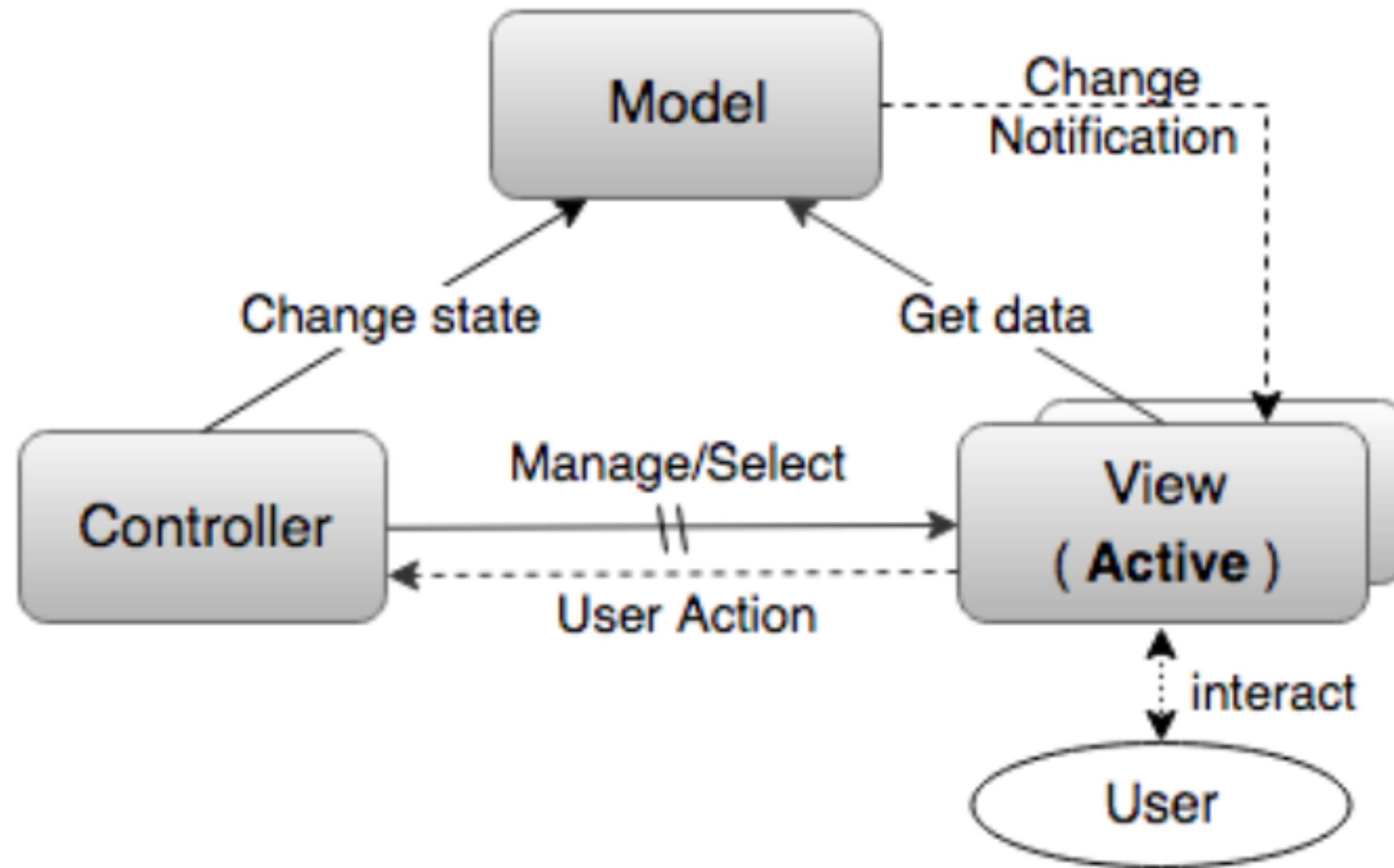
# MVC1



# MVC2

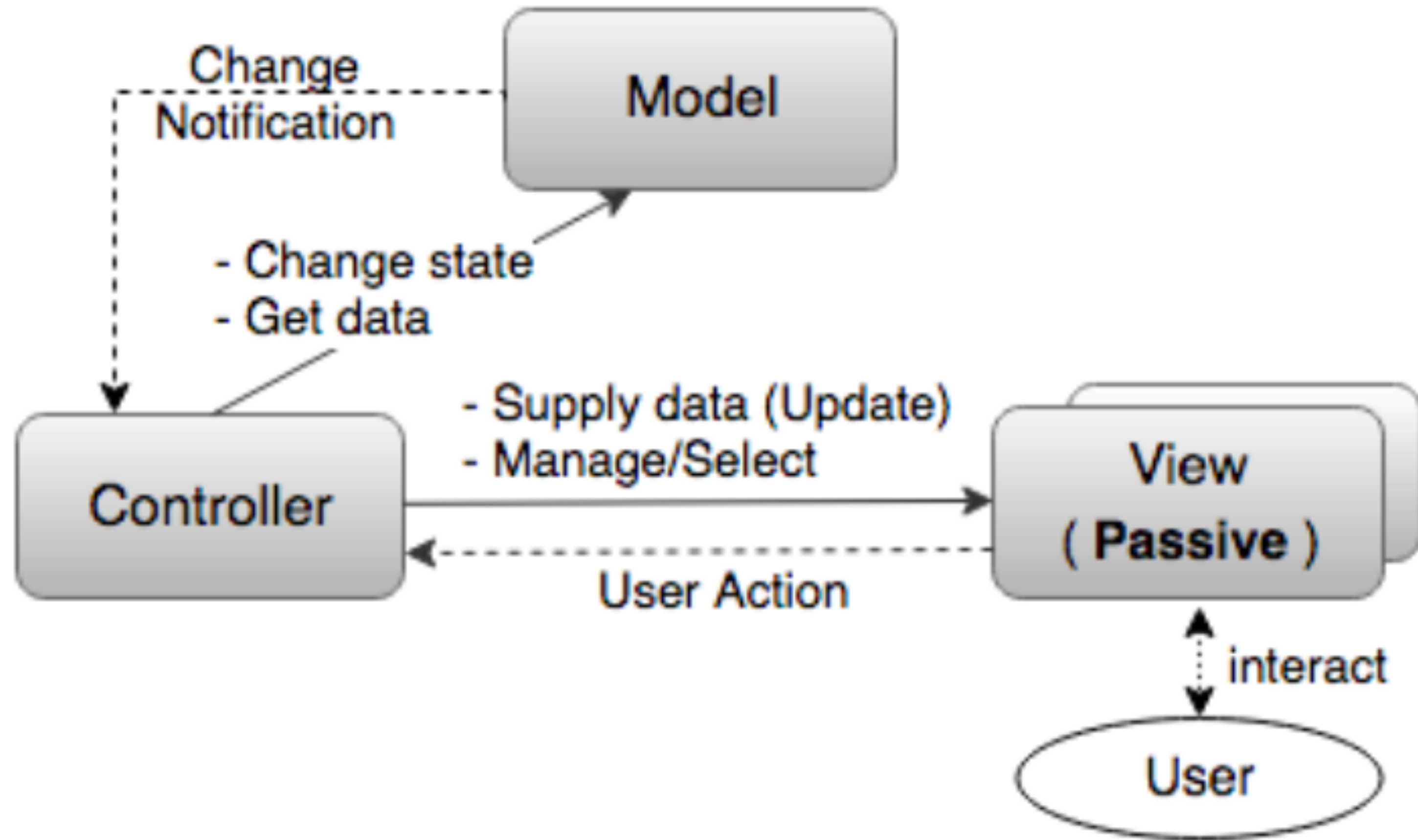


# MVC3



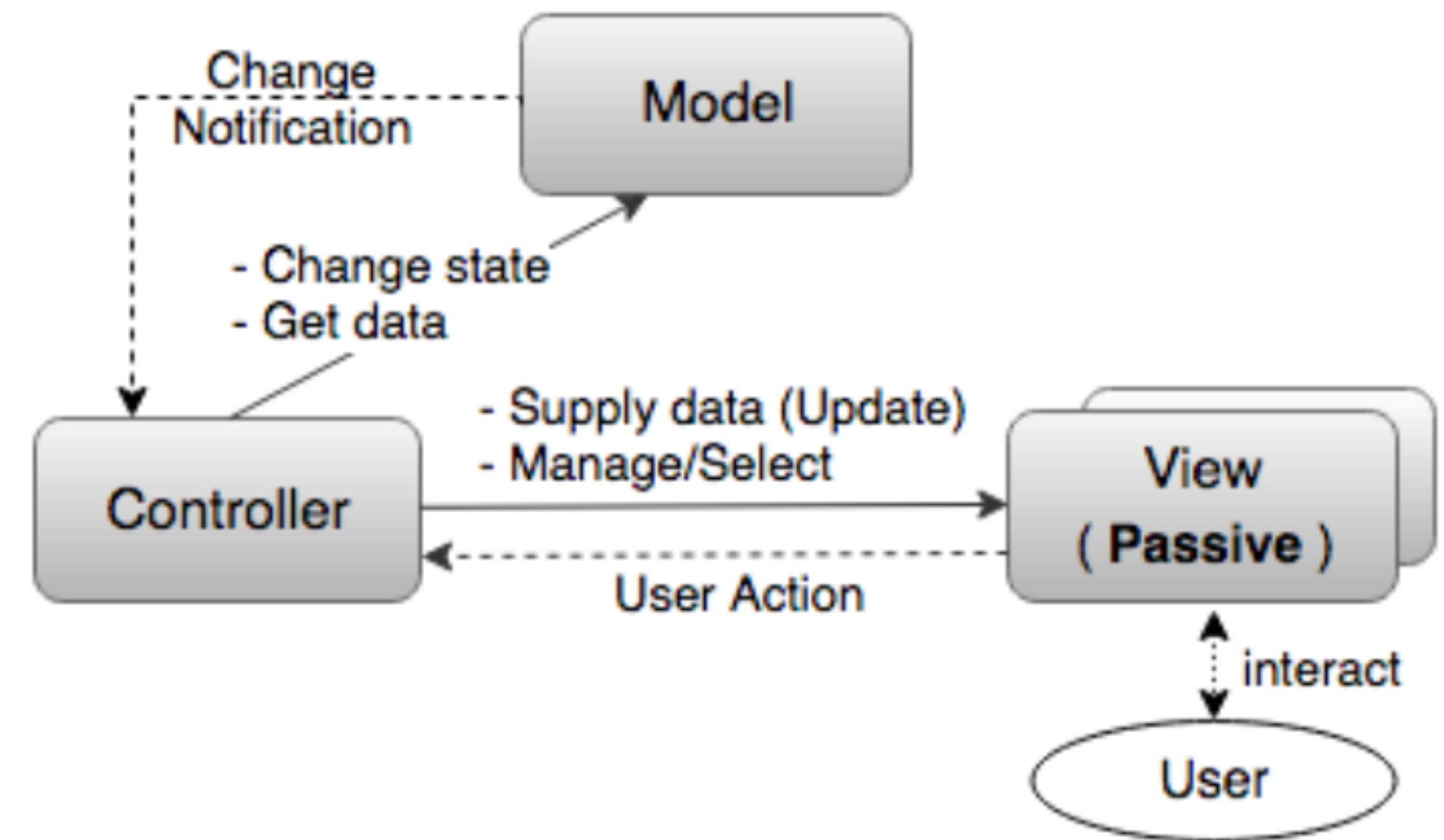


# Apple MVC



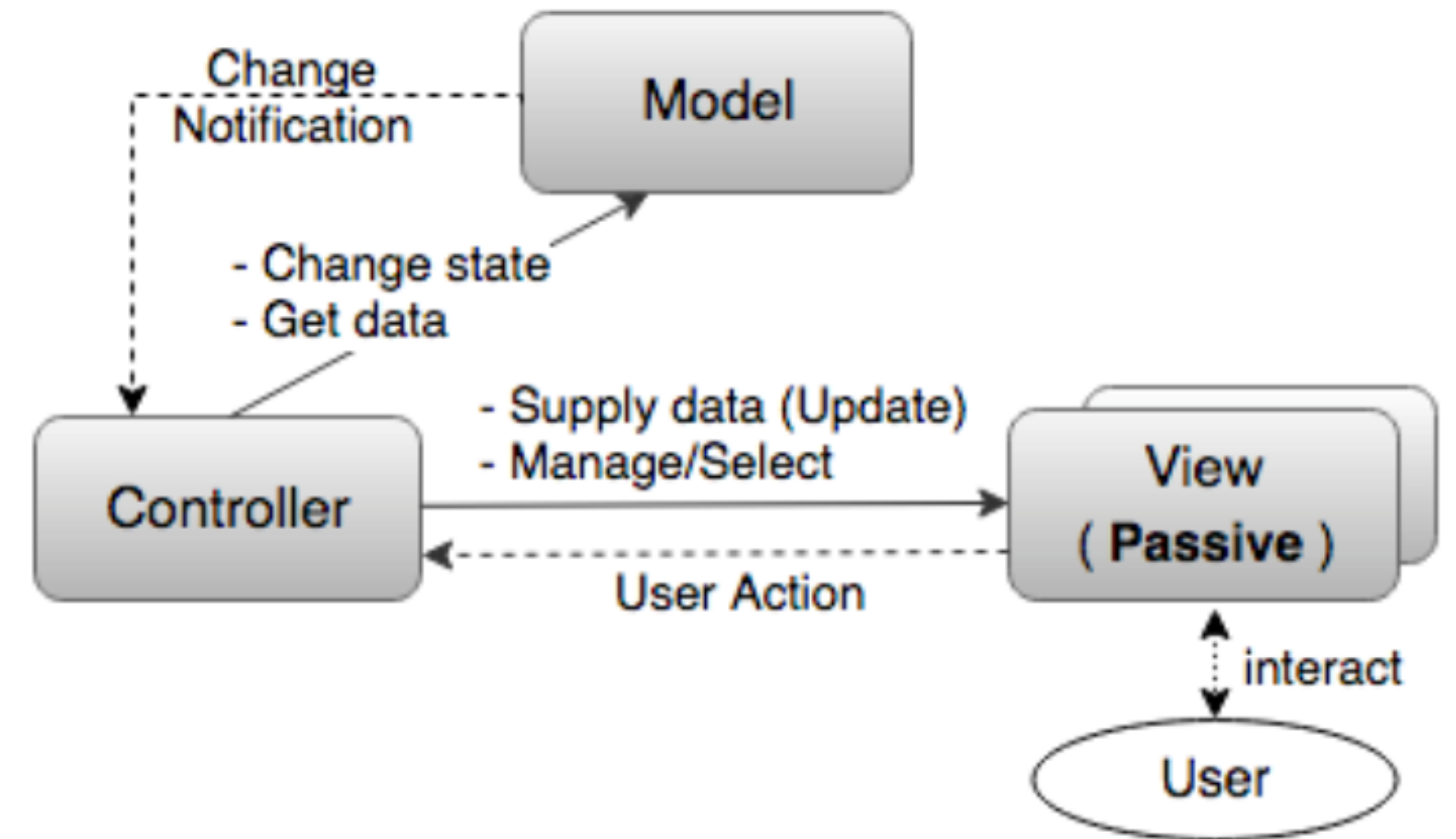
# Важно!

- › Все единицы - это собирательные названия классов, представляющих (например) Model может быть много
- › Model и View напрямую не взаимодействуют
- › MVC может быть вложенным



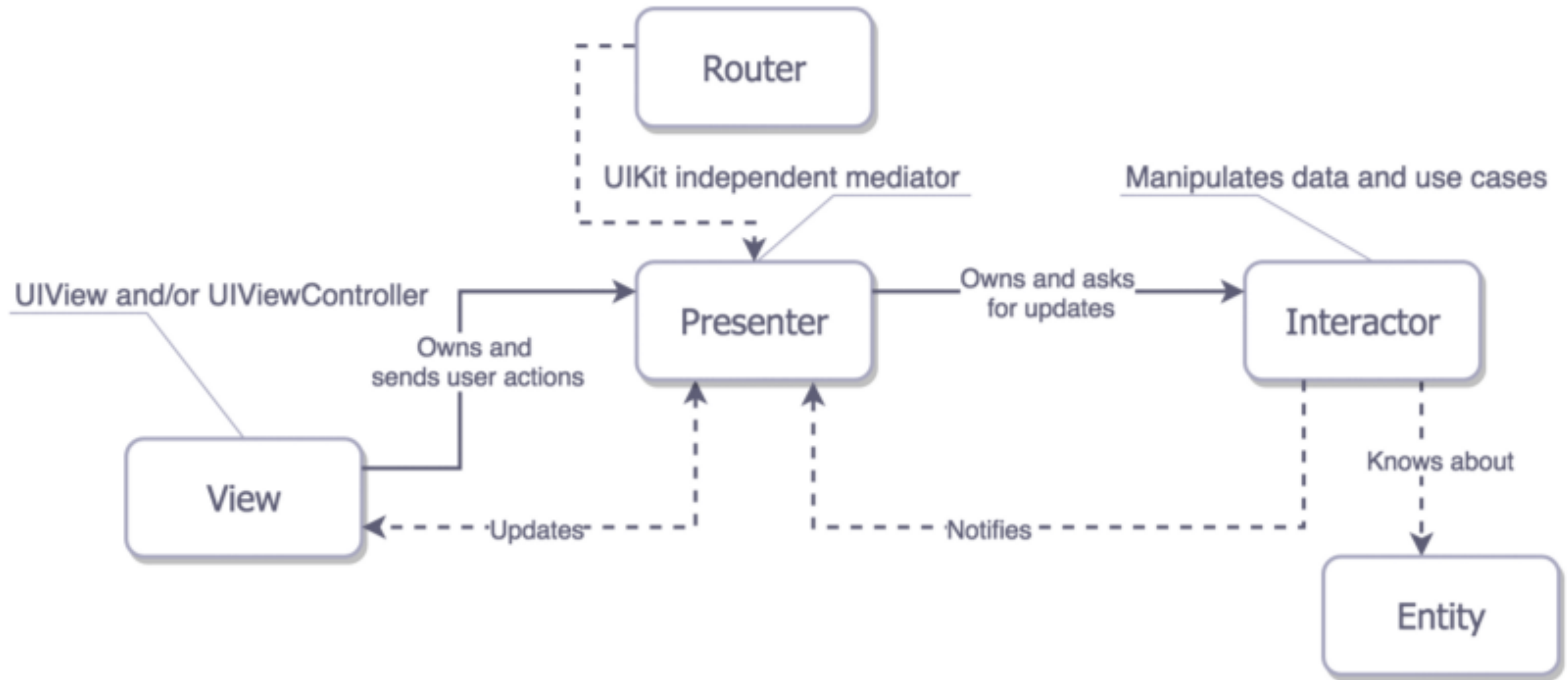
# Проблемы

- › Massive View Controller
- › Рост сложности

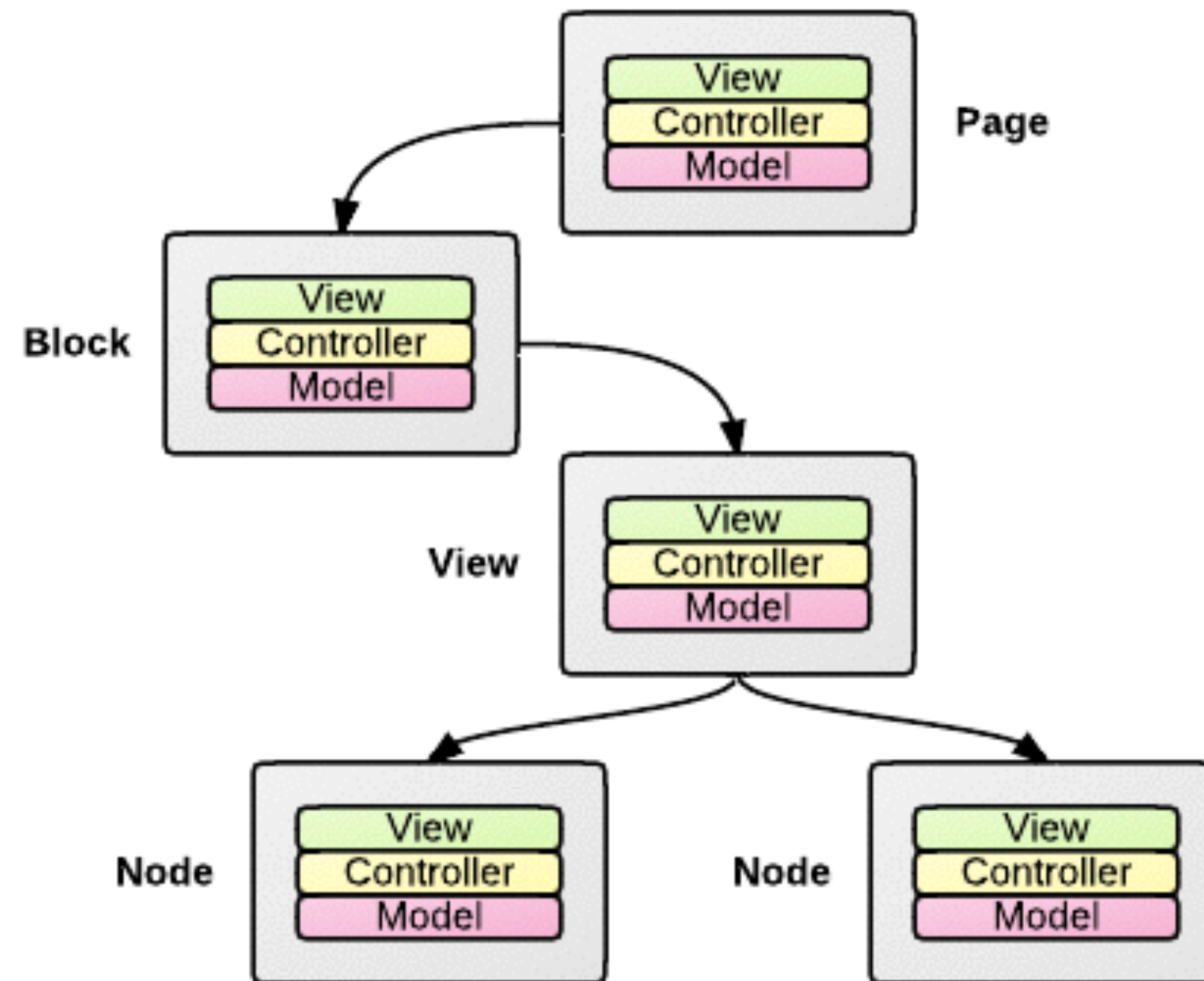




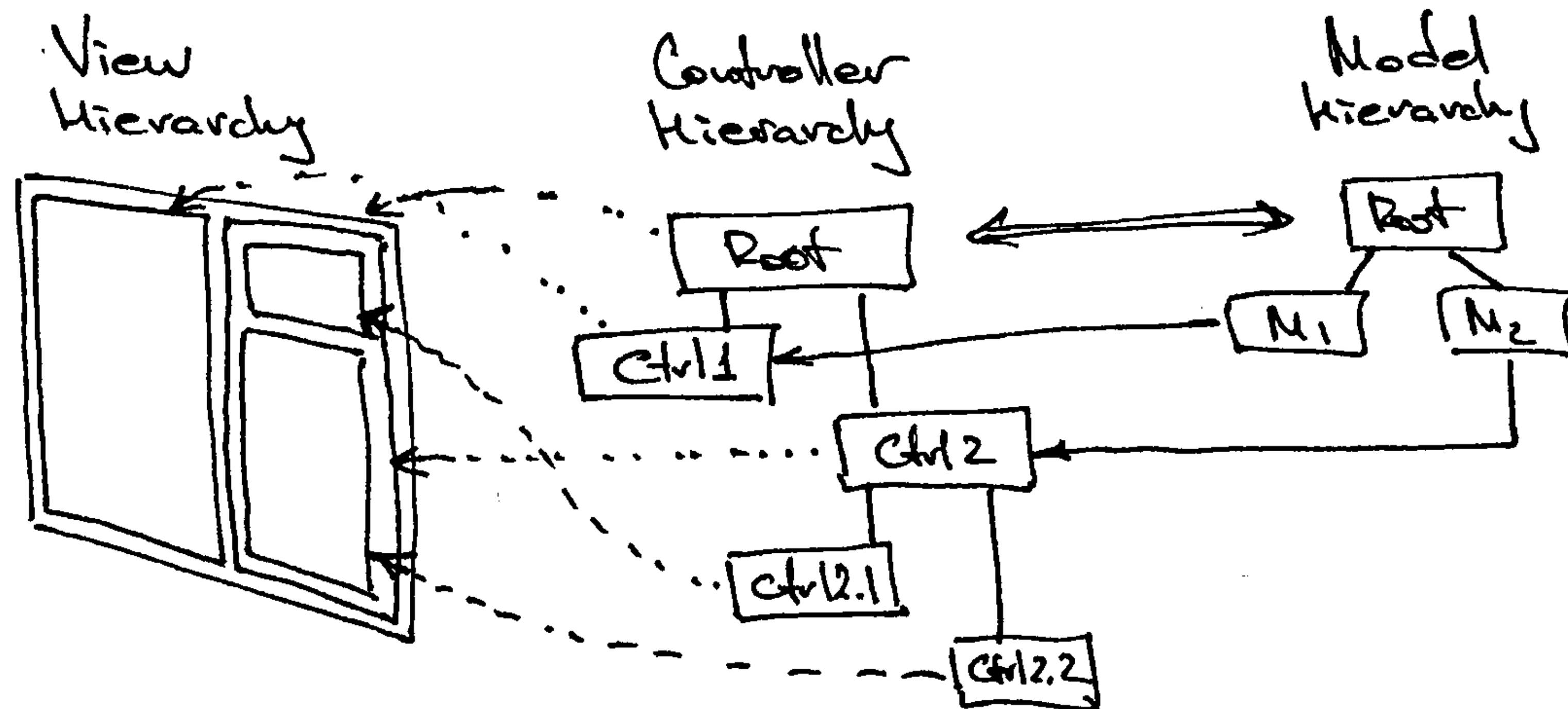
# VIPER (RRRRRR)



# HMVC (Как показывают)



# HMVC (Как на самом деле)

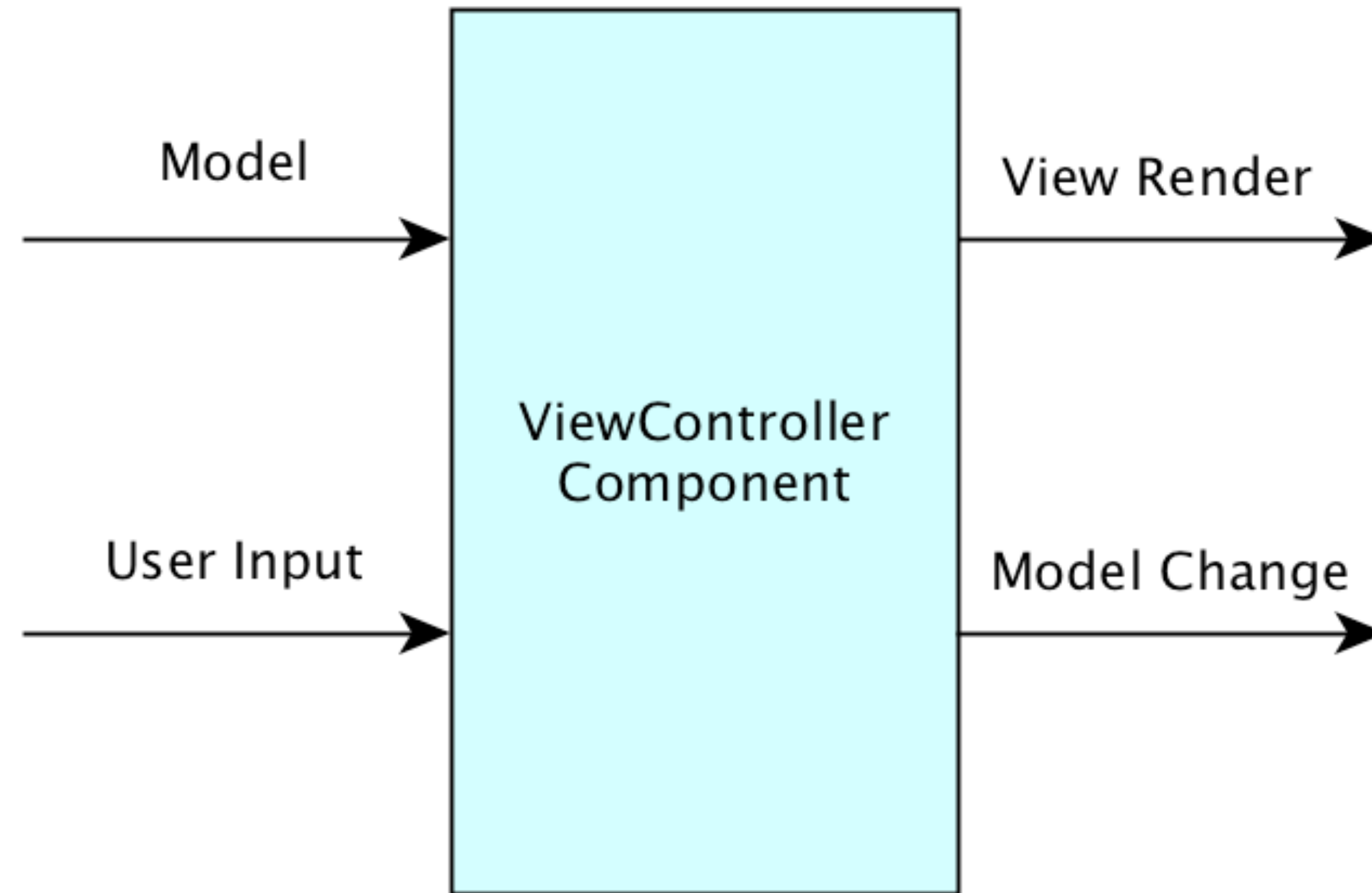


Части MVC выстраиваются в собственные иерархии, а не в иерархии модулей

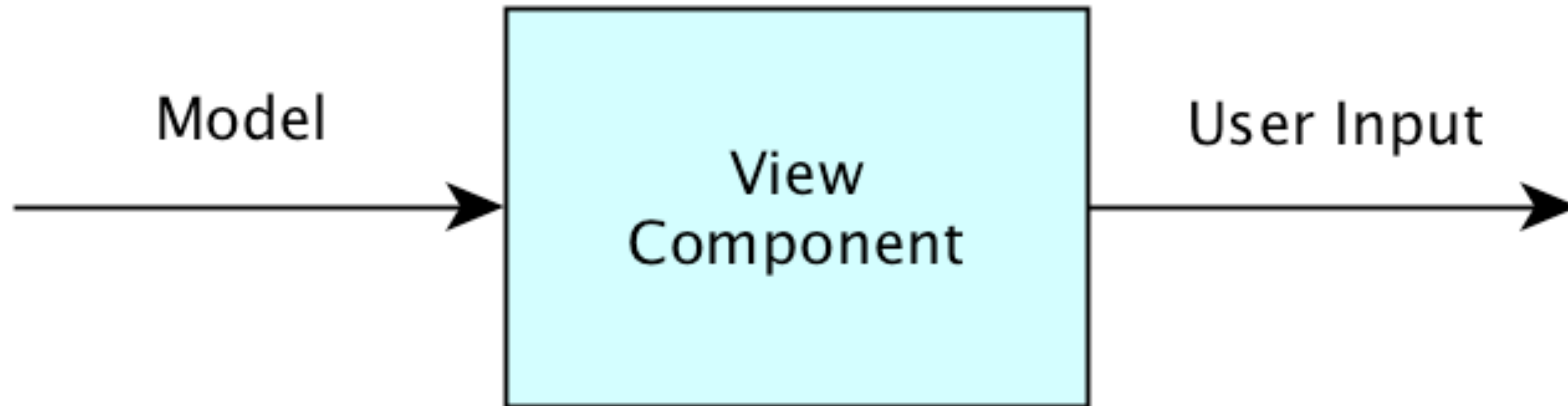
# Component



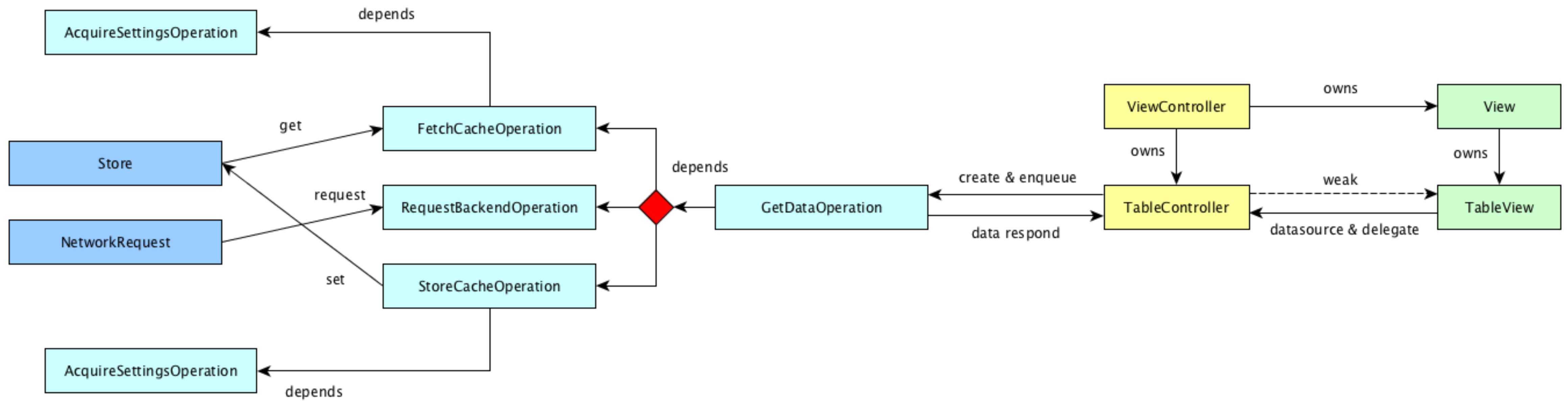
# View Controller Component



# View Component

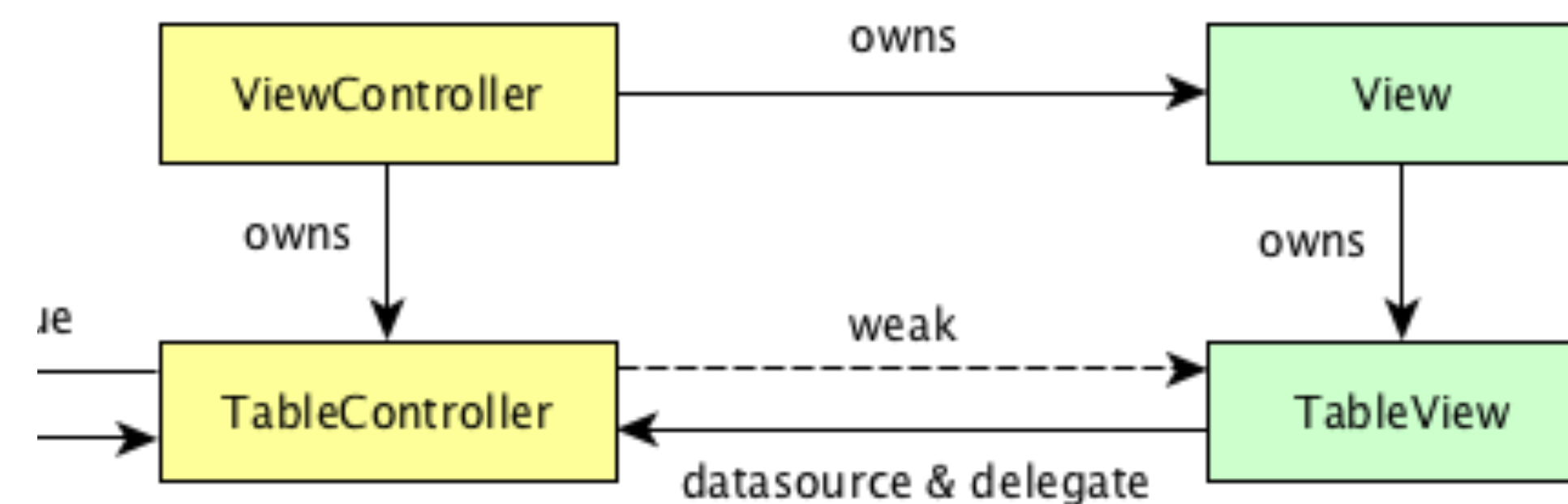


# Архитектура



# Представление & Контроллер

- › Тема 4 Лекции
- › О верстке и отрисовке
- › Обработка пользовательского ввода

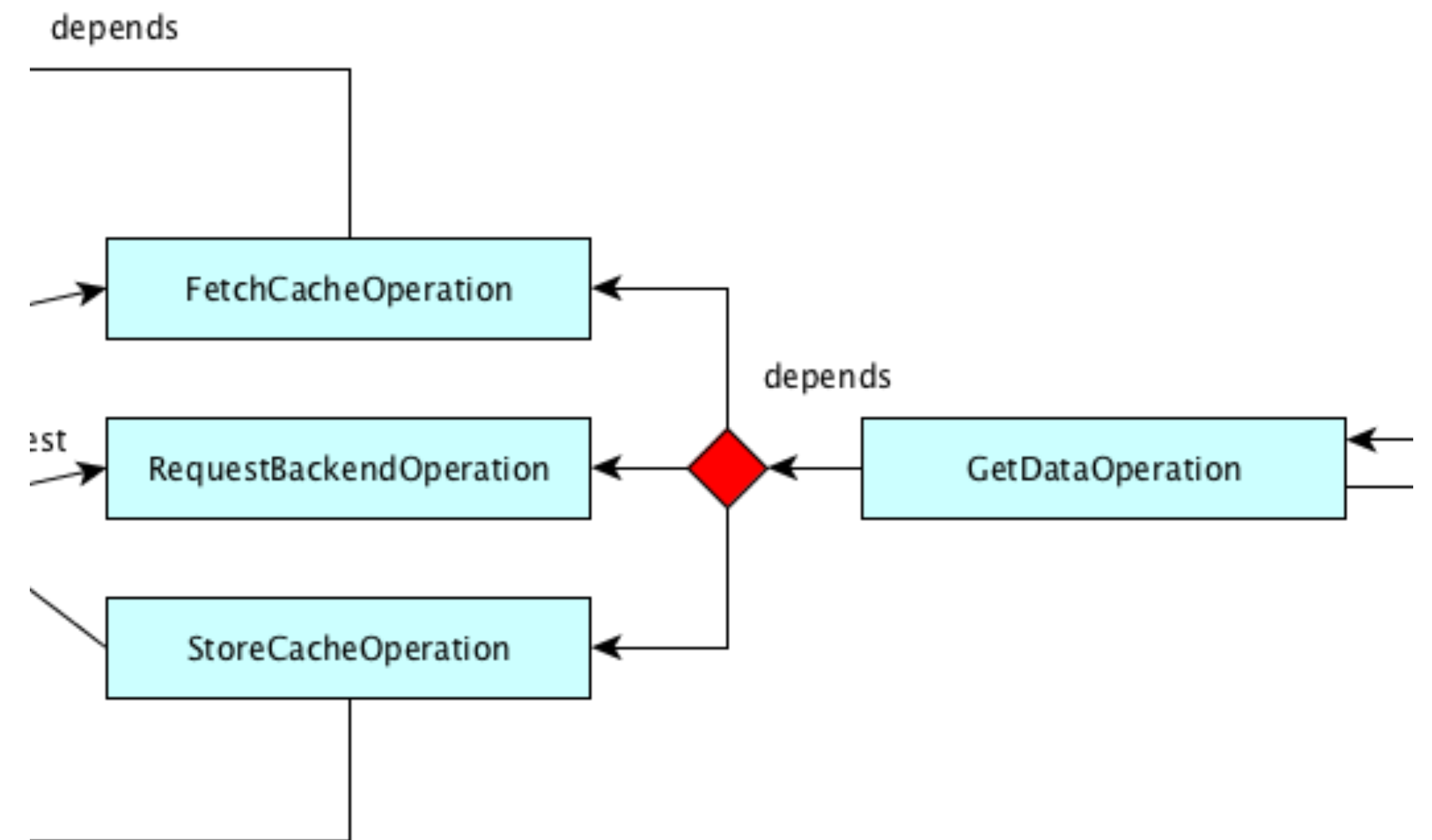




# Операции

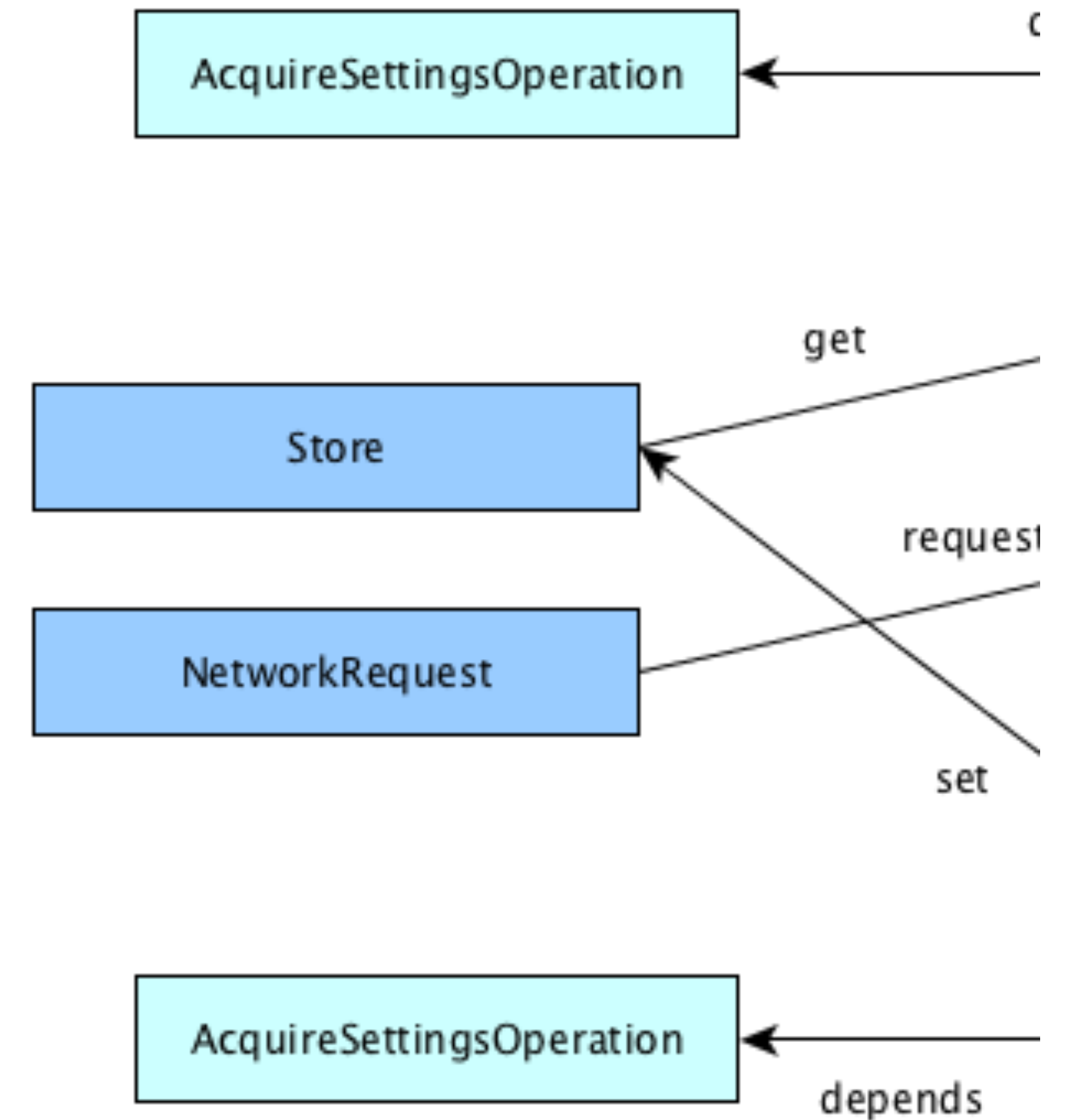
› Тема 5 Лекции

› Вопросы Concurrency



# Бизнес логика

- › Работа с сетью - Тема 6 лекции
- › Хранение данных - Тема 7 лекции



| Немного про DI и IoC

# IoC и DI

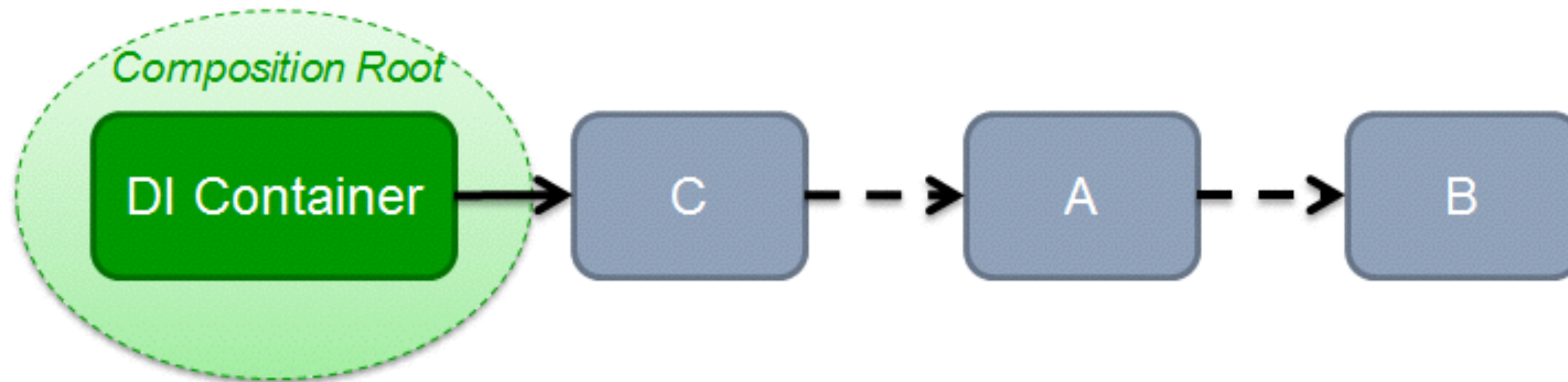
Dependencies



Service Location / **Active** Calling



Dependency Injection / Auto-Wiring / **Passive** Calling



Вопросы ?



# Задача



# Общие требования

- › \* В рамках данной задачи вам необходимо спроектировать архитектуру вашего приложения
- › \*\* Необходимо написать Unit-тесты для ваших классов

# Описание приложения

- › Два экрана: список заметок и окно редактирования одной заметки
- › Локальный кэш заметок
- › Синхронизация заметок с сервером



# Требования к архитектуре

- › Какие у вас будут экраны?
- › Какие будут элементы на них и, соответственно, компоненты?
- › Какие будут операции?
- › Какие будут сервисы бизнес-логики?

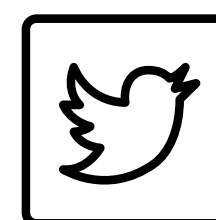
# Контакты

Малых Денис

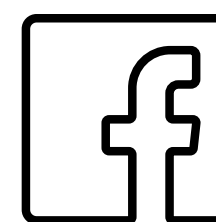
iOS Developer



[mrdekk@yandex.ru](mailto:mrdekk@yandex.ru)



@mrdekk



mrdekk

Спасибо за внимание

