Как стать архитектором?

Белорусское законодательство

- ☐ ОКРБ 006 - 2009 “Профессии рабочих и должности служащих”
- ☐ 24731 “Системный аналитик”
- ☐ 24732 “Системный архитектор”

Области знаний

- ☐ Программная архитектура
- ☐ Принципы проектирования
- ☐ Паттерны проектирования
- ☐ Паттерны конструирования

Программная архитектура

- ☐ Веб-сервер - конвейерная архитектура
- ☐ SOA - компонентная архитектура
- ☐ Клиент/сервер - многослойная архитектура
- ☐ Одноуровневое ПО - монолитная архитектура
- ☐ Облако - многоуровневая сетевая архитектура

Принципы проектирования

- ☐ Абстракция
- ☐ Инкапсуляция
- ☐ Связность
- ☐ Связанность
- ☐ Сложность

Паттерны проектирования

- ☐ Структурные / Поведенческие / Порождающие – GoF
- ☐ Управляемые событиями – Observer
- ☐ Plug-in’ы – Abstract Factory
- ☐ Domain / Active Record / Table Module – Layer (PoEAA)
- ☐ Алгоритмы – Strategy

Паттерны конструирования

- ☐ Проектирование наследования
- ☐ Компонентное проектирование
- ☐ Проектирование слоёв (layer design)
- ☐ Проектирование уровней (tier design)
- ☐ Методология поставки
- ☐ Ухудшение архитектуры

Виды архитекторов

- [ ] Application Architect
- [ ] Solution Architect
- [ ] Enterprise Architect
- [ ] Cloud Architect
- [ ] Infrastructure Architect
- [ ] System Architect

Многие “эксперты” не знают

- ☐ Фундаментальные основы паттернов GoF
- ☐ Понятие “Язык паттернов” (“Pattern Language”)
- ☐ Ссылки на первоисточники

Особенности ответов на вопросы интервью

- ☐ Правильные ответы на вопросы
- ☐ Использовать стандартную терминологию
- ☐ Стандартный ответ vs. ответ меньшинства
- ☐ Решение vs. экзотическое решение

Типовые вопросы интервью

- [ ] Программная архитектура
- [ ] Принципы проектирования
- [ ] Паттерны проектирования
- [ ] Принципы конструирования

Другие моменты

- ☐ Стеки технологий, производители
- ☐ ООП / функциональное / императивное
- ☐ MVC / MVP / MVVM
- ☐ Big data / Cloud computing / OO Database
- ☐ Стандарты

Тема: Стили и паттерны проектирования Архитектуры ПО

1. Архитектурные стили
2. Области применения архитектурных паттернов
3. Системы архитектурных паттернов
4. MVC и его наследники

Архитектурные стили

- семейство систем с точки зрения схемы организации структуры
- определяет набор компонентов и соединений
- описывает ряд ограничений (например, топологических)

Архитектура клиент-сервер

Система разделяется на два приложения, где клиент выполняет запросы к серверу.

Во многих случаях в роли сервера выступает база данных, а логика приложения представлена процедурами хранения.

- ☐ безопасность
- ☐ централизованный доступ к данным
- ☐ простота обслуживания

Компонентная архитектура

Дизайн приложения разлагается на функциональные или логические компоненты с возможностью повторного использования, предоставляющие тщательно проработанные интерфейсы связи.

- ☐ повторное использование
- ☐ замещаемость
- ☐ независимость от контекста
- ☐ расширяемость
- ☐ инкапсуляция

Domain Driven Design

Объектно-ориентированный архитектурный стиль, ориентированный на моделирование сферы деловой активности и определяющий бизнес-объекты на основании сущностей этой сферы.

- ☐ обмен информацией
- ☐ расширяемость
- ☐ удобство тестирования

Многослойная архитектура

Функциональные области приложения разделяются на многослойные группы (уровни).

- ☐ абстракция
- ☐ инкапсуляция
- ☐ функциональные слои
- ☐ высокая связность
- ☐ повторное использование
- ☐ слабое связывание

Шина сообщений

Архитектурный стиль, предписывающий использование программной системы, которая может принимать и отправлять сообщения по одному или более каналам связи, так что приложения получают возможность взаимодействовать, не располагая конкретными сведениями друг о друге.

- ☐ взаимодействие, основанное на сообщениях
- ☐ сложная логика обработки
- ☐ изменение логики обработки
- ☐ интеграция с разными инфраструктурами

Сервис-ориентированная архитектура

Описывает приложения, предоставляющие и потребляющие функциональность в виде сервисов с помощью контрактов и сообщений.

- ☐ сервисы автономны
- ☐ сервисы могут быть распределены
- ☐ сервисы слабо связаны
- ☐ сервисы совместно используют схему или контракт, но не класс
- ☐ совместимость основана на политике

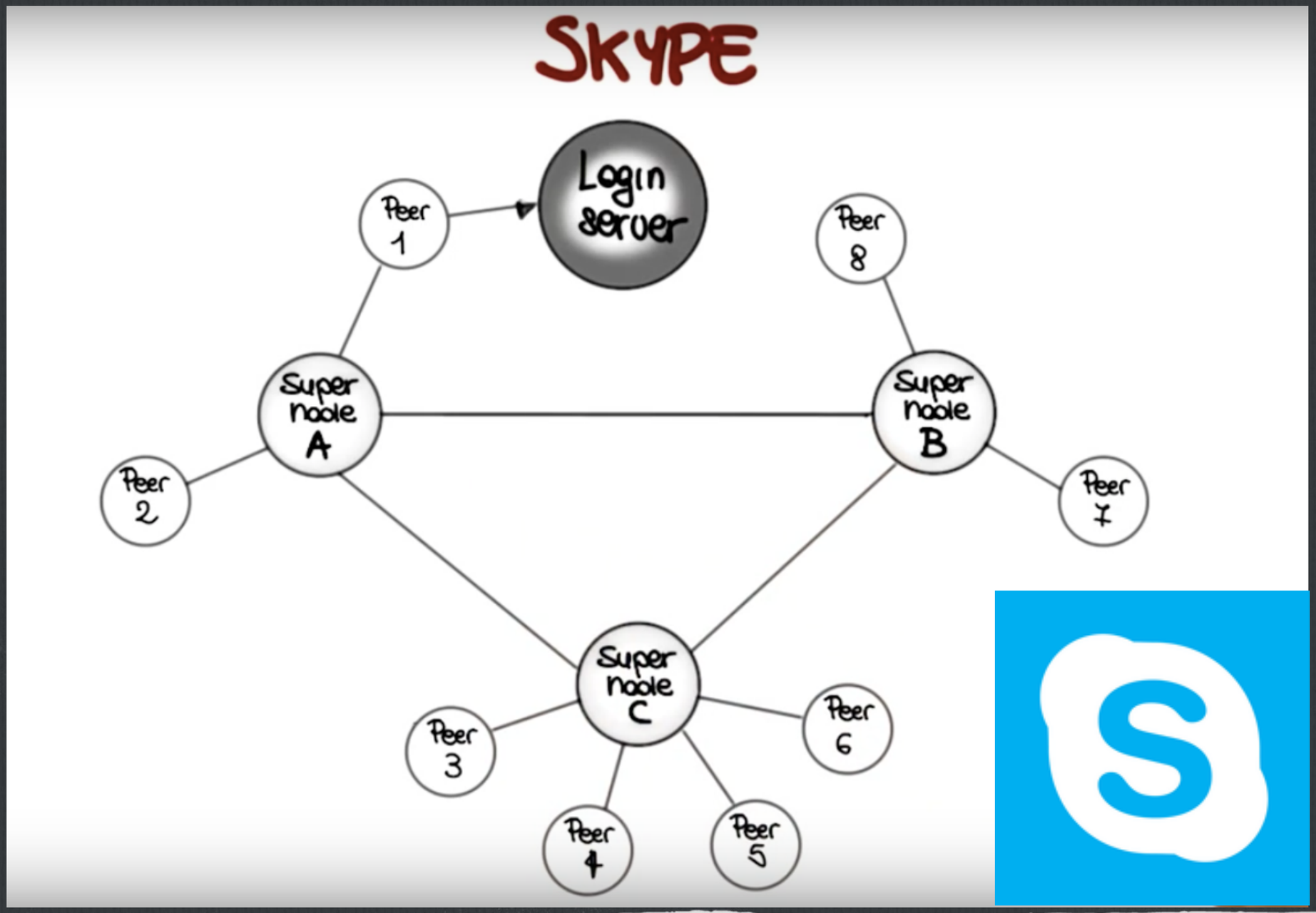
Ещё одна классификация

- ☐ pipes & filters
- ☐ event driven architecture
- ☐ client-server
- ☐ publisher-subscriber
- ☐ p2p
- ☐ REST

Выбор стиля

- ☐ требования бизнеса
- ☐ требования качества
- ☐ особенности проекта
- ☐ сочетания стилей

Пример


Skype
Login server
Peer 1
Peer 8
Super node A
Super node B
Peer 2
Peer 7
Super node C
Peer 3
Peer 6
Peer 4
Peer 5

Язык паттернов

- ☐ Обобщённая задача / Типовое решение
- ☐ Название / Проблема / Решение / Последствия
- ☐ Архитектура / Проектирование / Идиомы

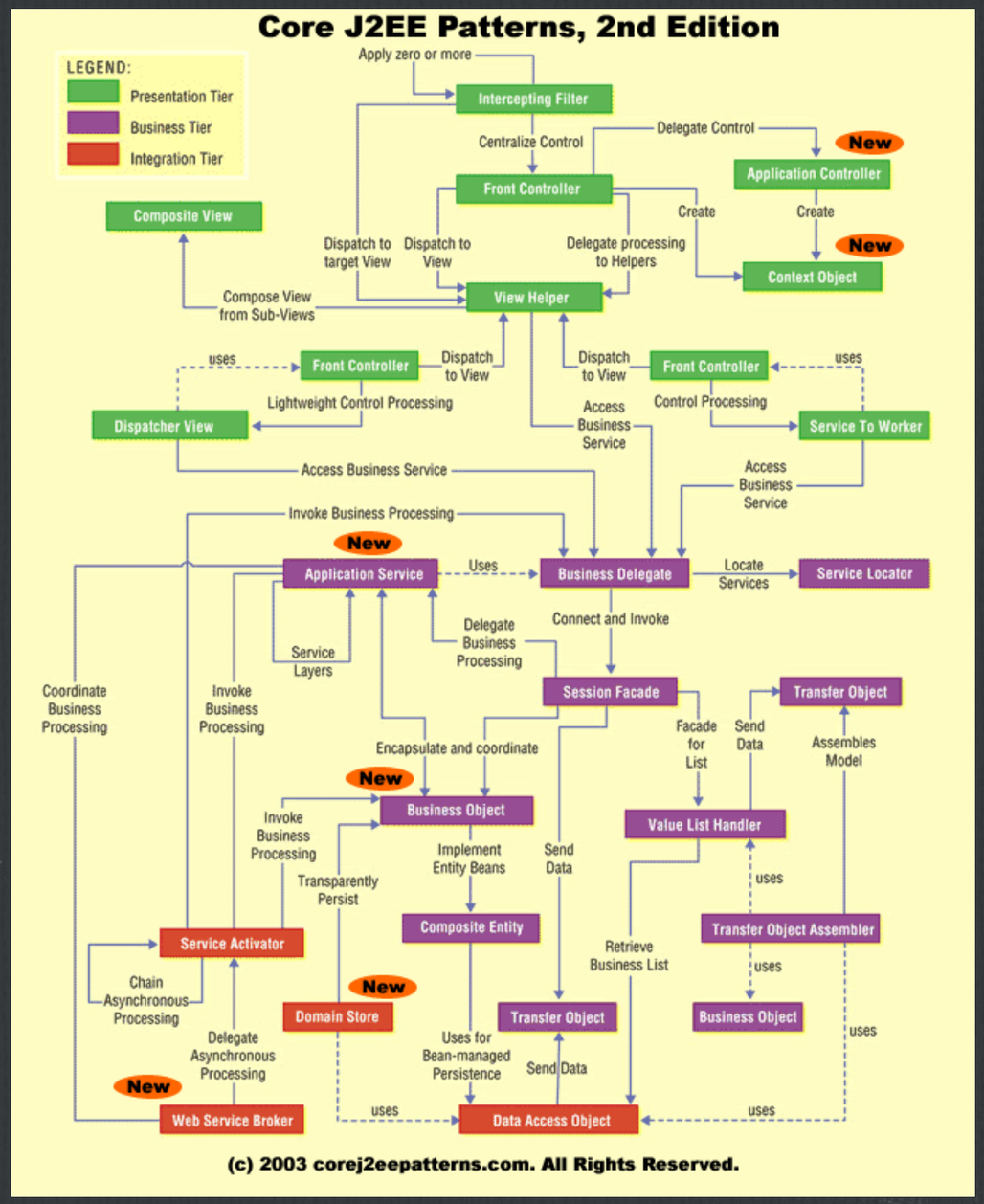
Архитектурные паттерны

- ☐ Система - подсистема
- ☐ Компоненты и их связи
- ☐ Предметная область
- ☐ Уровни и слои (Tiers and Layers)

Стили и шаблоны Microsoft

- ☐ Типы приложений
- ☐ Версии платформы и средств разработки
- ☐ Атрибуты качества

Система J2EE

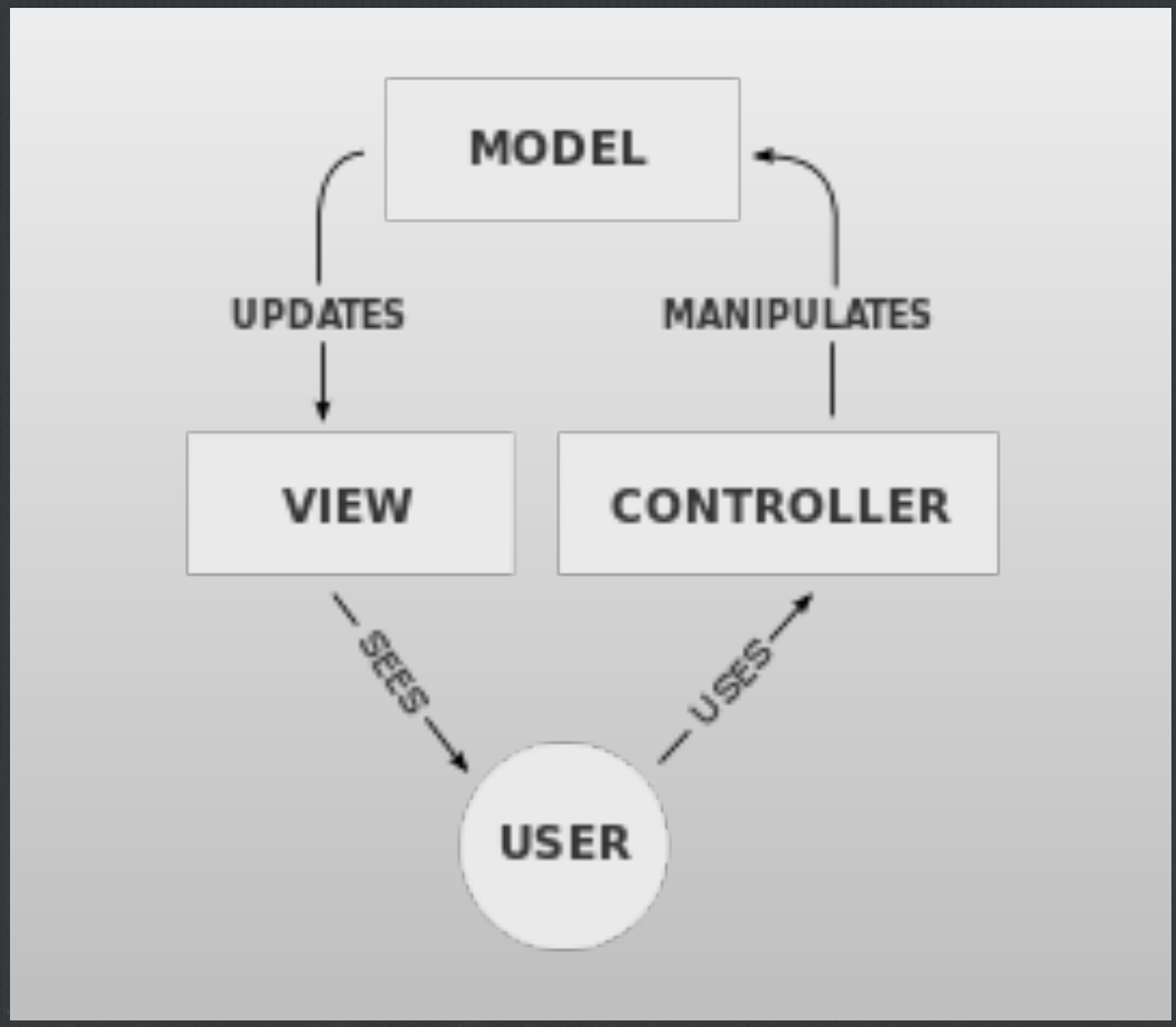


- ☐ presentation tier
- ☐ business tier
- ☐ integration tier

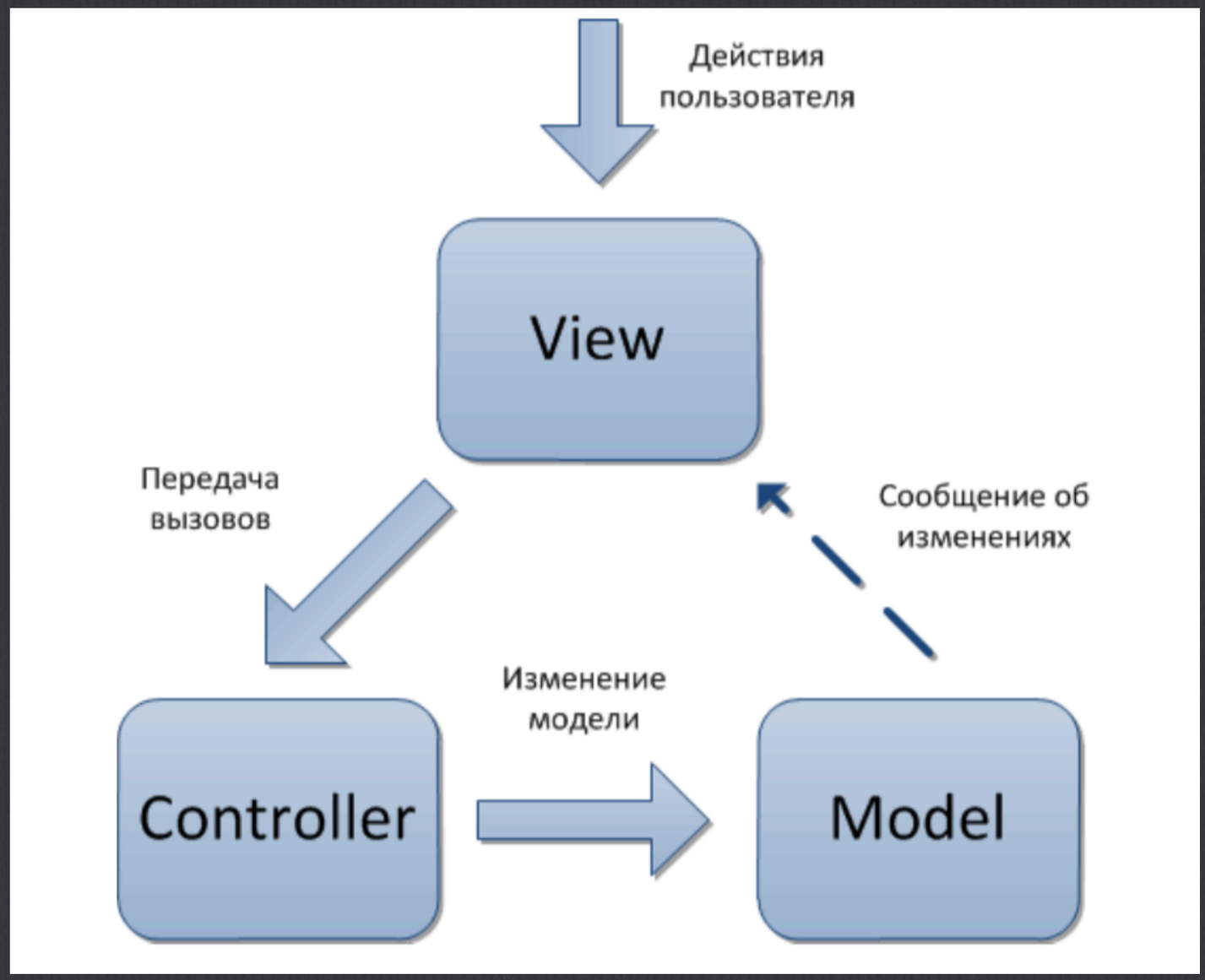
PoEAA

- ☐ Domain Logic
- ☐ Data Source
- ☐ Object-Relational Behavioral / Structural
- ☐ Web Presentation
- ☐ Distribution
- ☐ Offline Concurrency
- ☐ Session State

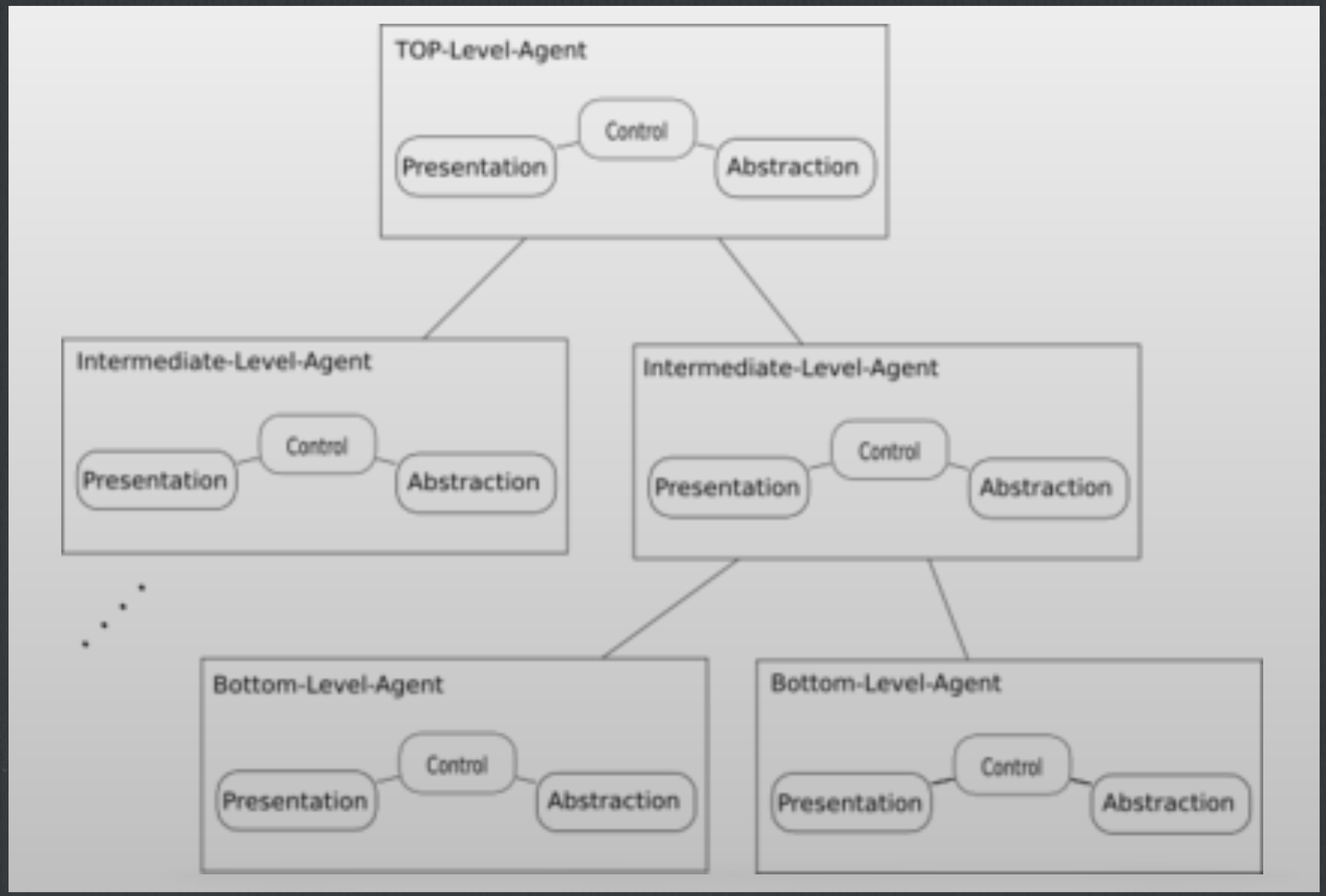
MVC (не вставлять в диплом!)



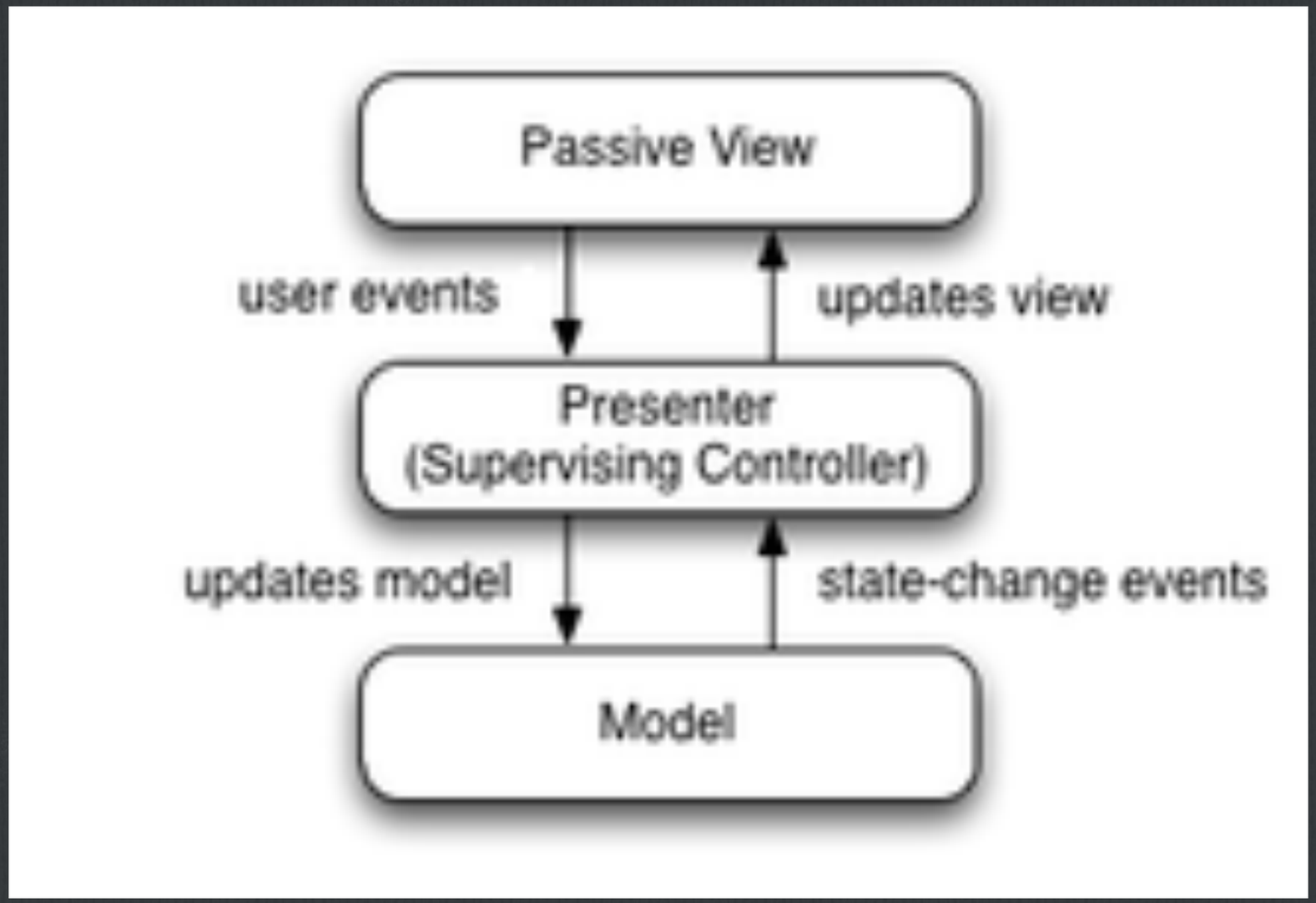
MVC


Действия
пользователя
View
Передача
вызовов
Сообщение об
изменениях
Изменение
модели
Controller
Model

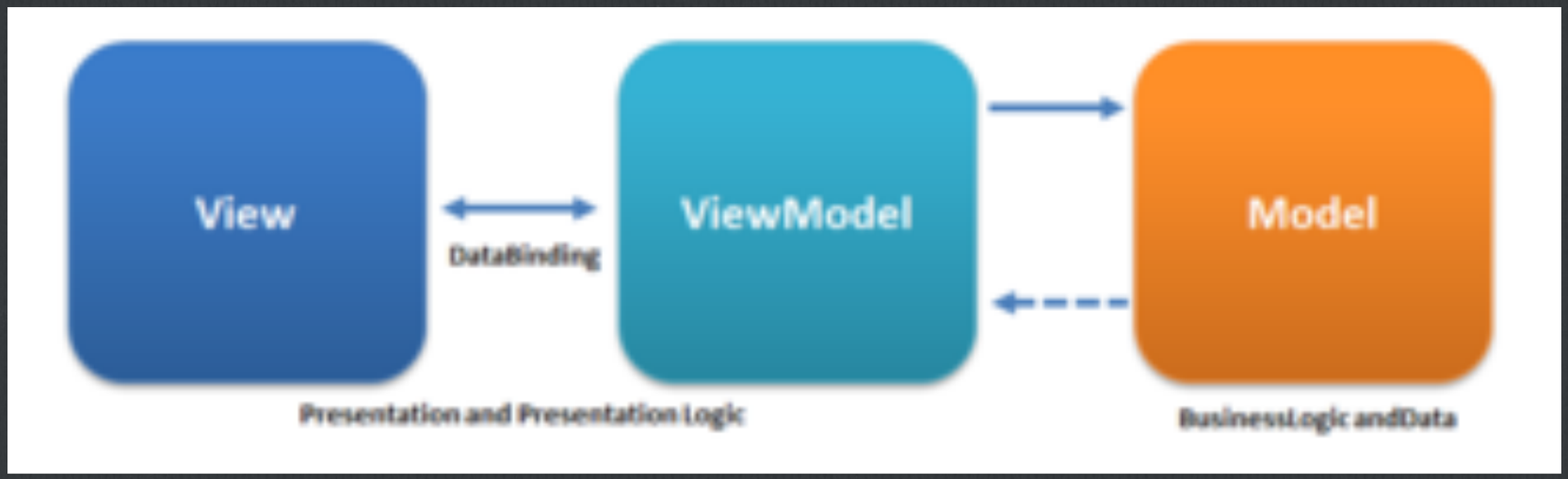
HMVC (HPAC)


TOP-Level-Agent
Control
Presentation
Abstraction
Intermediate-Level-Agent
Control
Presentation
Abstraction
Intermediate-Level-Agent
Control
Presentation
Abstraction
Bottom-Level-Agent
Control
Presentation
Abstraction
Bottom-Level-Agent
Control
Presentation
Abstraction

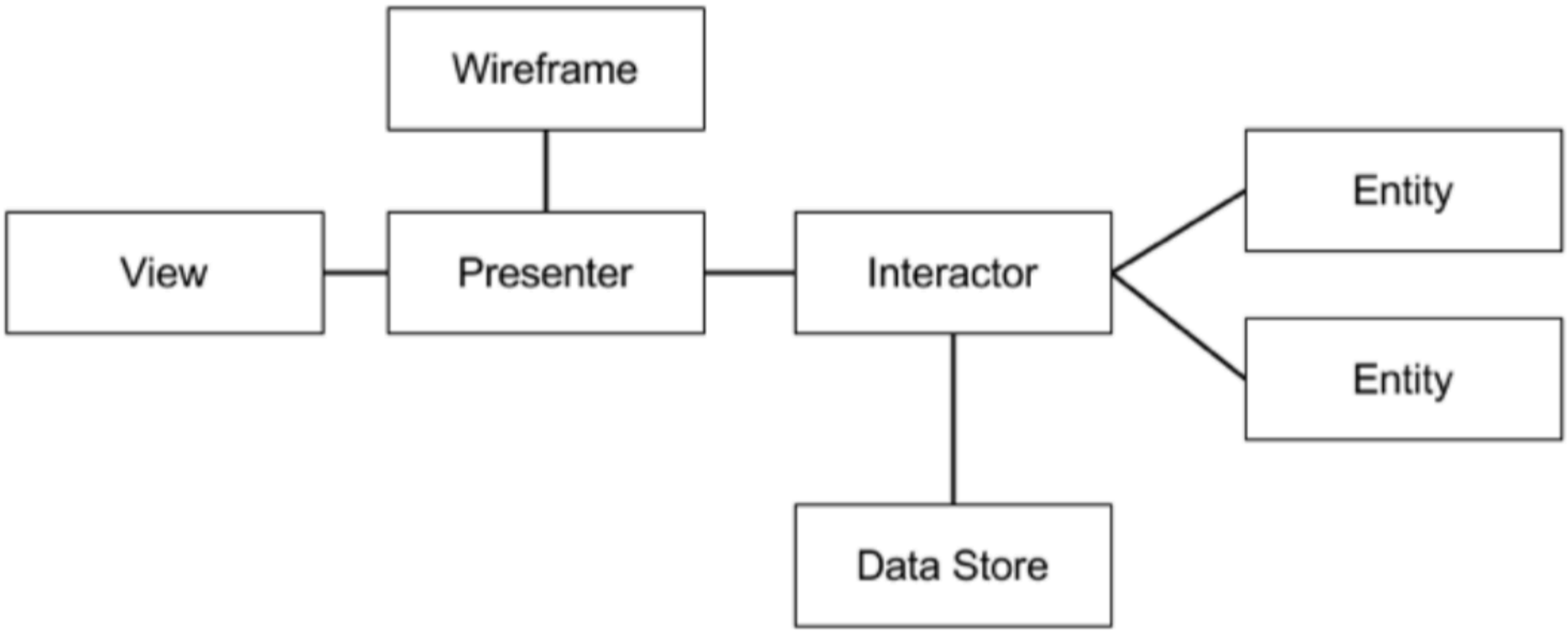
MVP


Passive View
user events
updates view
Presenter
(Supervising Controller)
updates model
state-change events
Model

MVVM


View
DataBinding
ViewModel
Model
Presentation and Presentation Logic
BusinessLogic andData

VIPER


Wireframe
View
Presenter
Interactor
Entity
Entity
Data Store

MVC и неправильная трактовка

- [ ] MVC — Massive-View-Controller!
- [ ] Бессмысленное повторение«чисто книжного» MVC != нормальный MVC
- [ ] У всех свой MVC (iOS, Java, .NET, Rails)