Skillbox

Архитектурные стили

Монолит

Павел Елисеев

Software Architect
Сбер

Skillbox

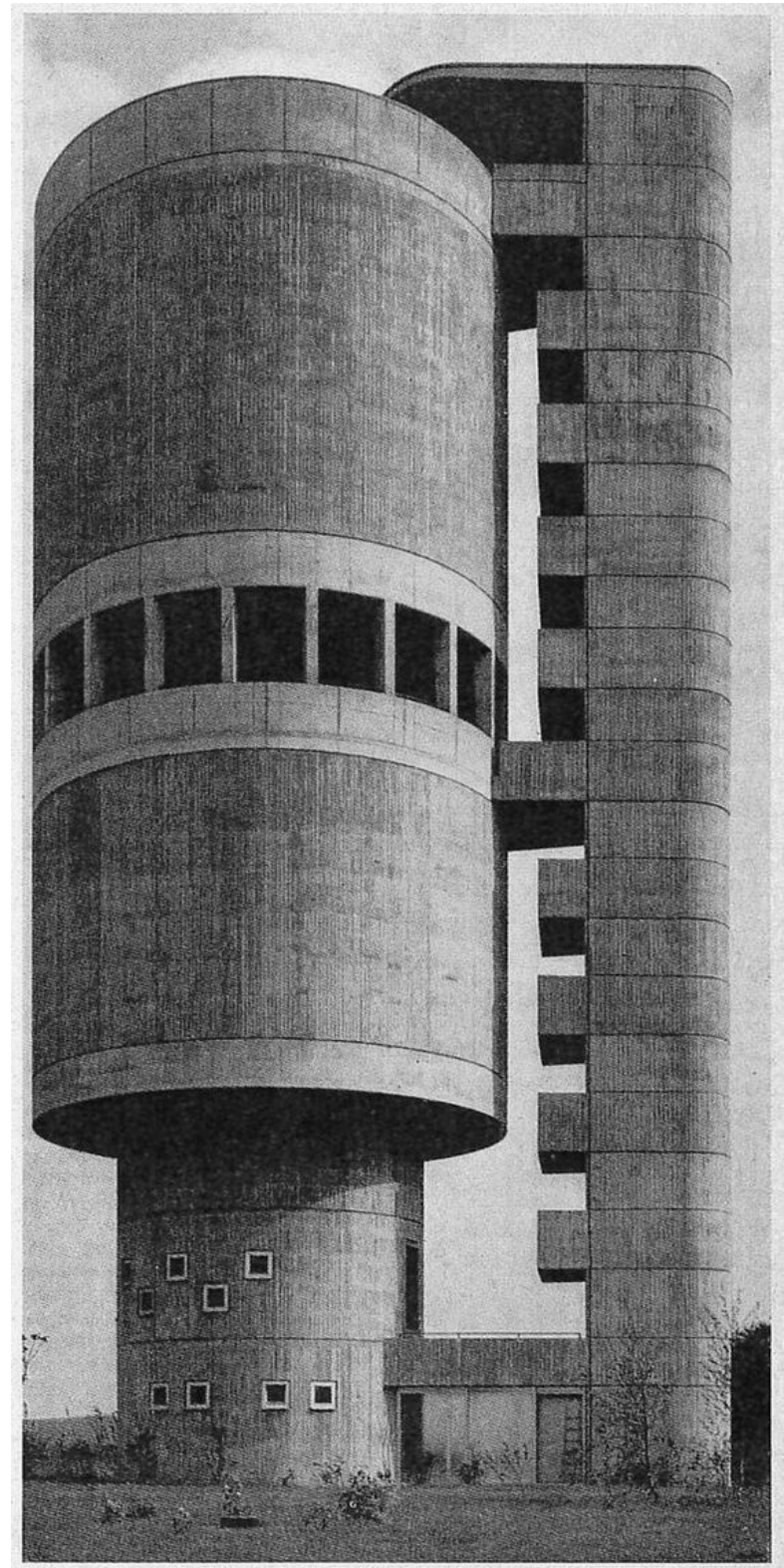
На прошлом занятии

- Рассмотрели, как развивалась архитектура ПО.
- Познакомились с фундаментальными архитектурными стилями.

На этом занятии

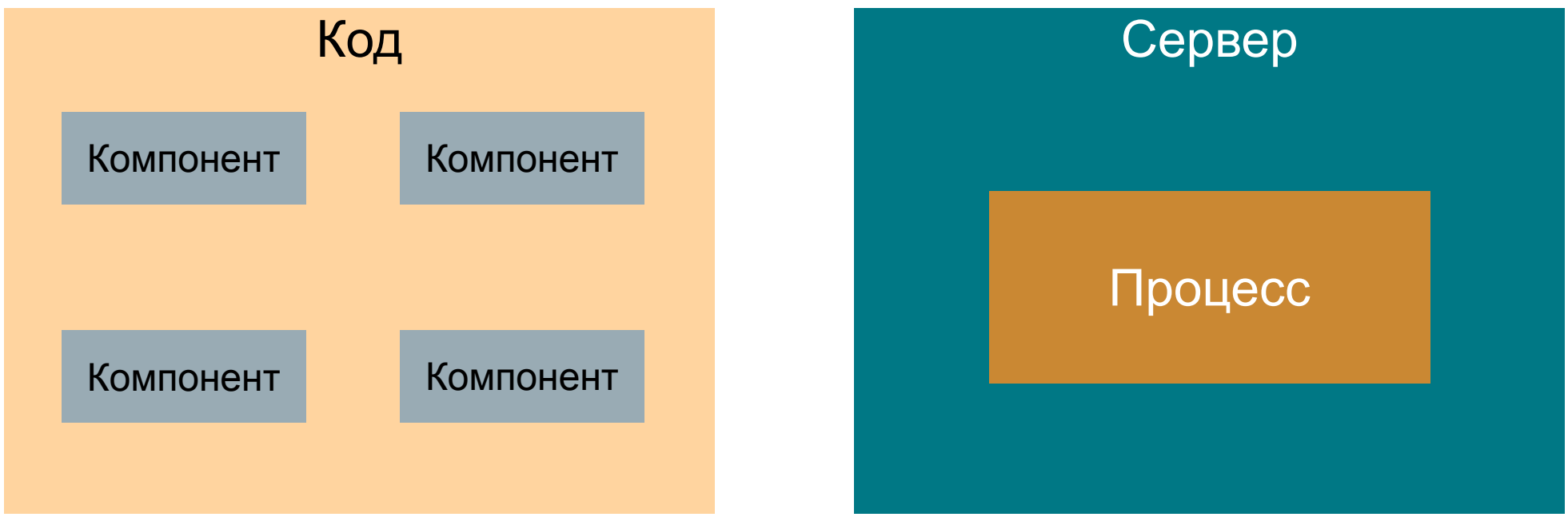
- Рассмотрим архитектурные стили построения монолитных приложений.
- Обсудим, что монолит — это плохо.

Монолит



- **Устанавливается** и исполняется как один **процесс** на одном **узле**.

Монолит



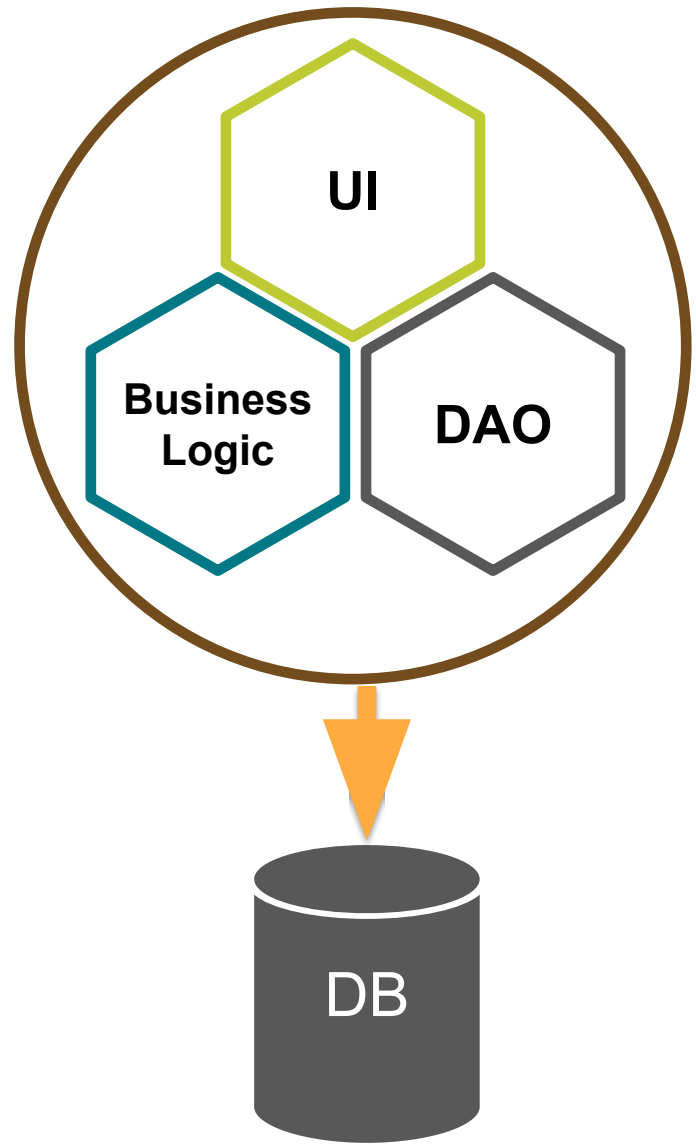
Представление реализации

Процессное и физическое представление

Skillbox

Монолит. Стили

- Microkernel
- Layered
- Modular

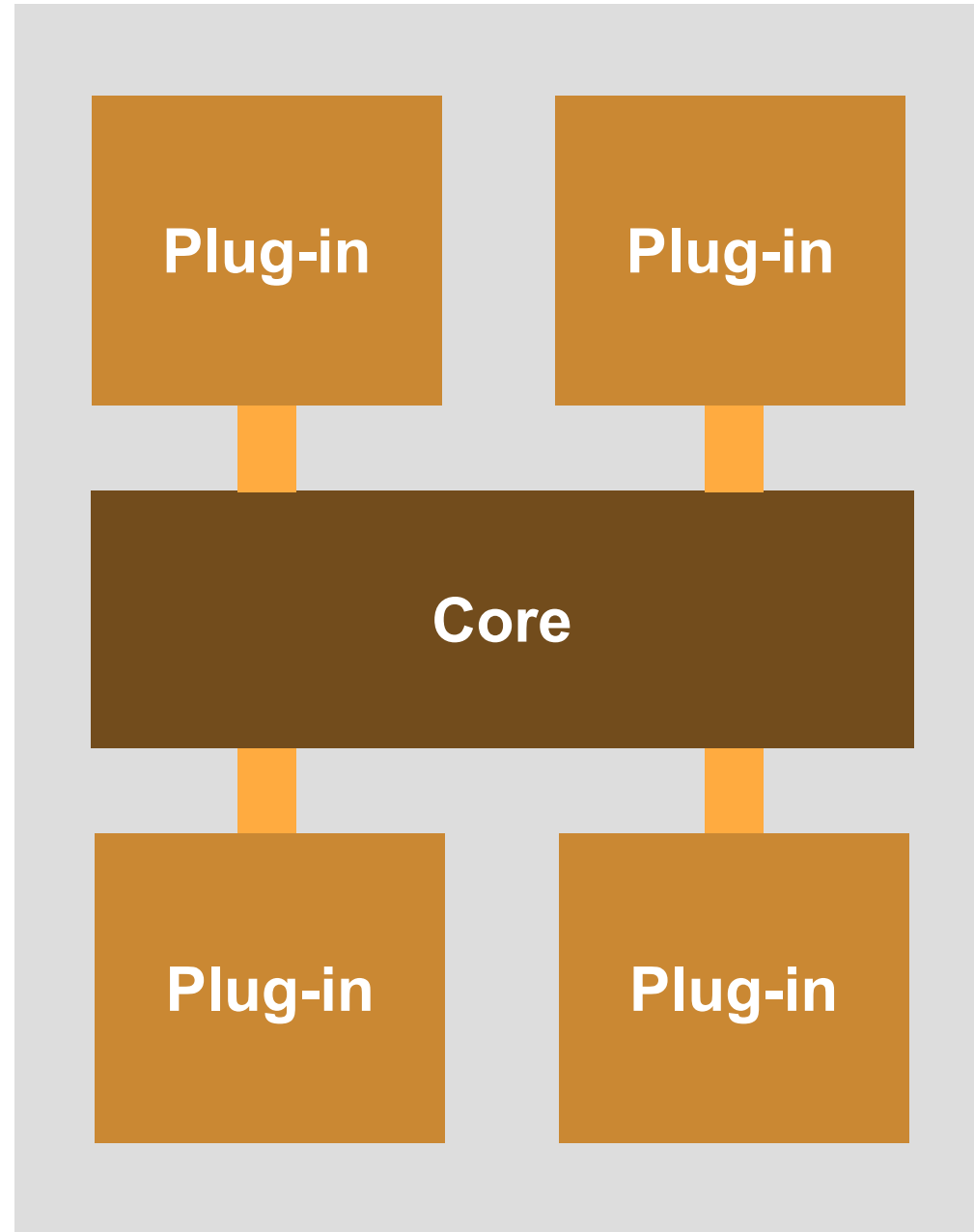


Skillbox

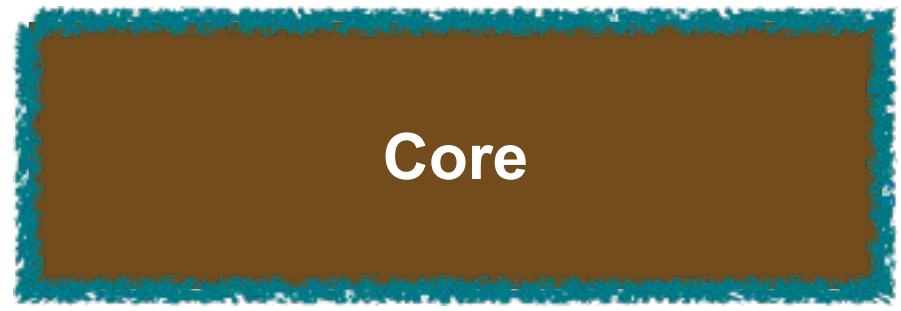
Microkernel

Ключевые компоненты

- Ядро
- Плагин
- Реестр

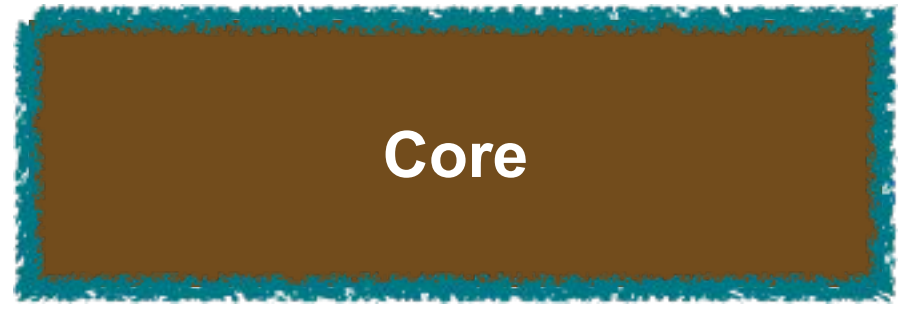


Microkernel. Ядро




```
public void startCourse(String courseName) {
    switch (courseName) {
        case "SoftwareArchitect":
            startSoftwareArchitectCourse();
            break;
        case "Spring":
            startSpringCourse();
            break;
        case "Python":
            startPythonCourse();
            break;
        case "SMM":
            startSMMCourse();
            break;
    }
}
```

Microkernel. Ядро




```
public void startCourse(String courseName) {
    String course = courseRegistry.get(courseName);
    try {
        Class<?> clazz = Class.forName(course);
        Constructor<?> constructor = clazz.getConstructor();
        CoursePlugin coursePlugin =
                (CoursePlugin)constructor.newInstance();
        coursePlugin.start();
    } catch (Exception e) {
        handleException(e);
    }
}
```

Skillbox

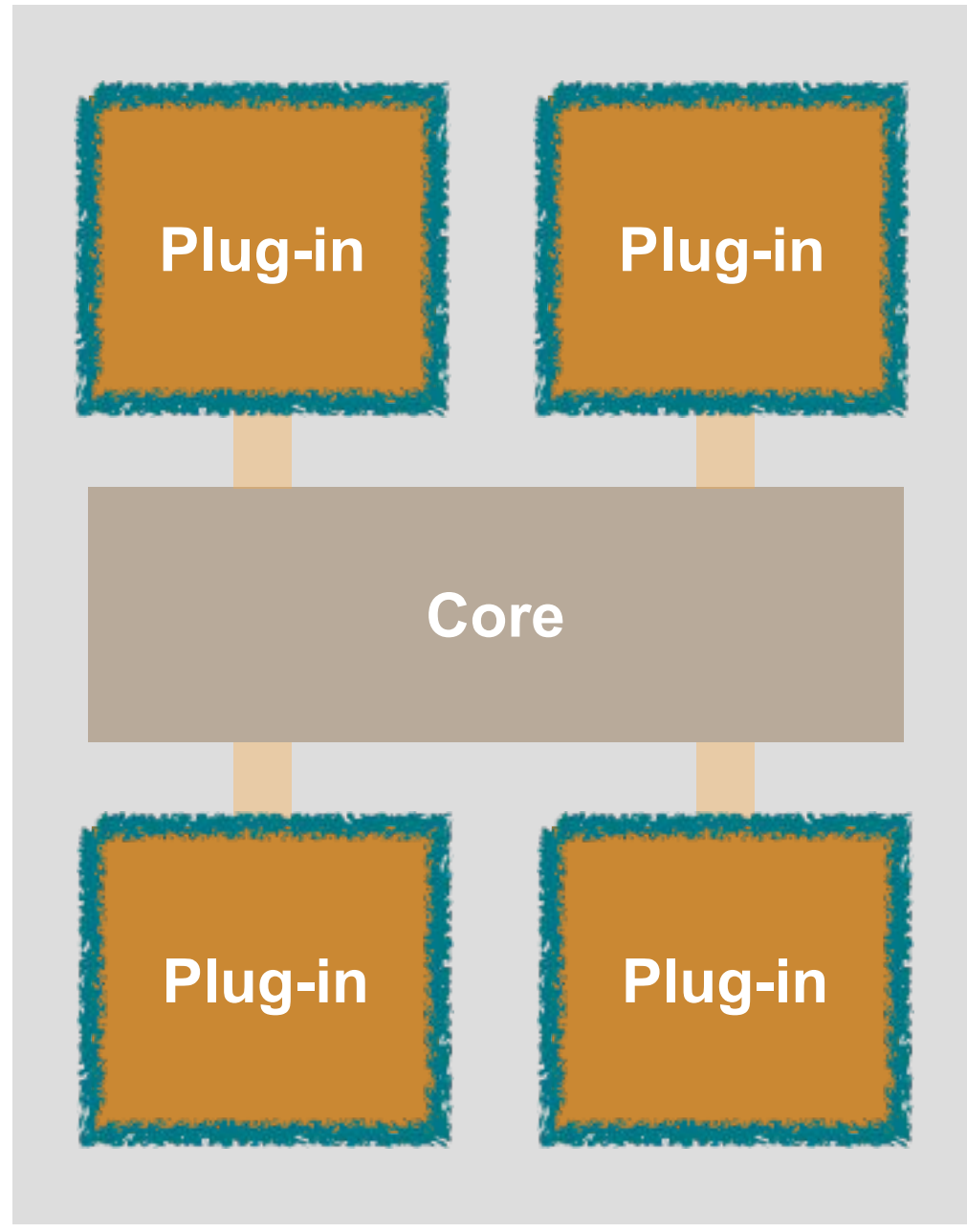
Microkernel. Плагин

Принцип

- Автономность

Управление

- На логическом представлении (OSGi)
- На представлении реализации



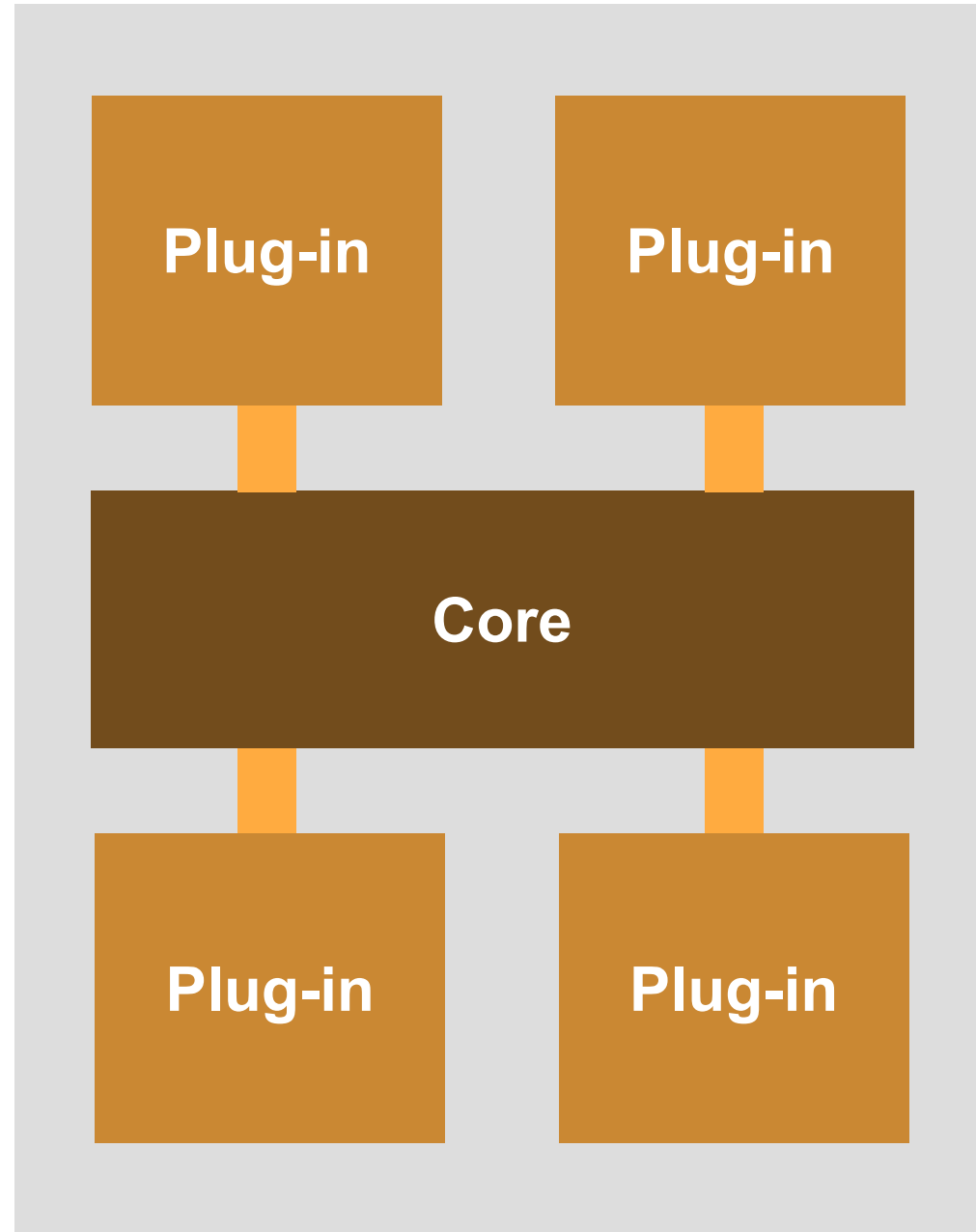
Skillbox

Microkernel. Регистр

```
static Map<String, String> courseRegistry = new HashMap<>();
static {
    courseRegistry.put("SoftwareArchitect", "SoftwareArchitectPlugin");
    courseRegistry.put("Spring", "SpringPlugin");
    courseRegistry.put("Python", "PythonPlugin");
    courseRegistry.put("SMM", "SMMPlugin");
}
```

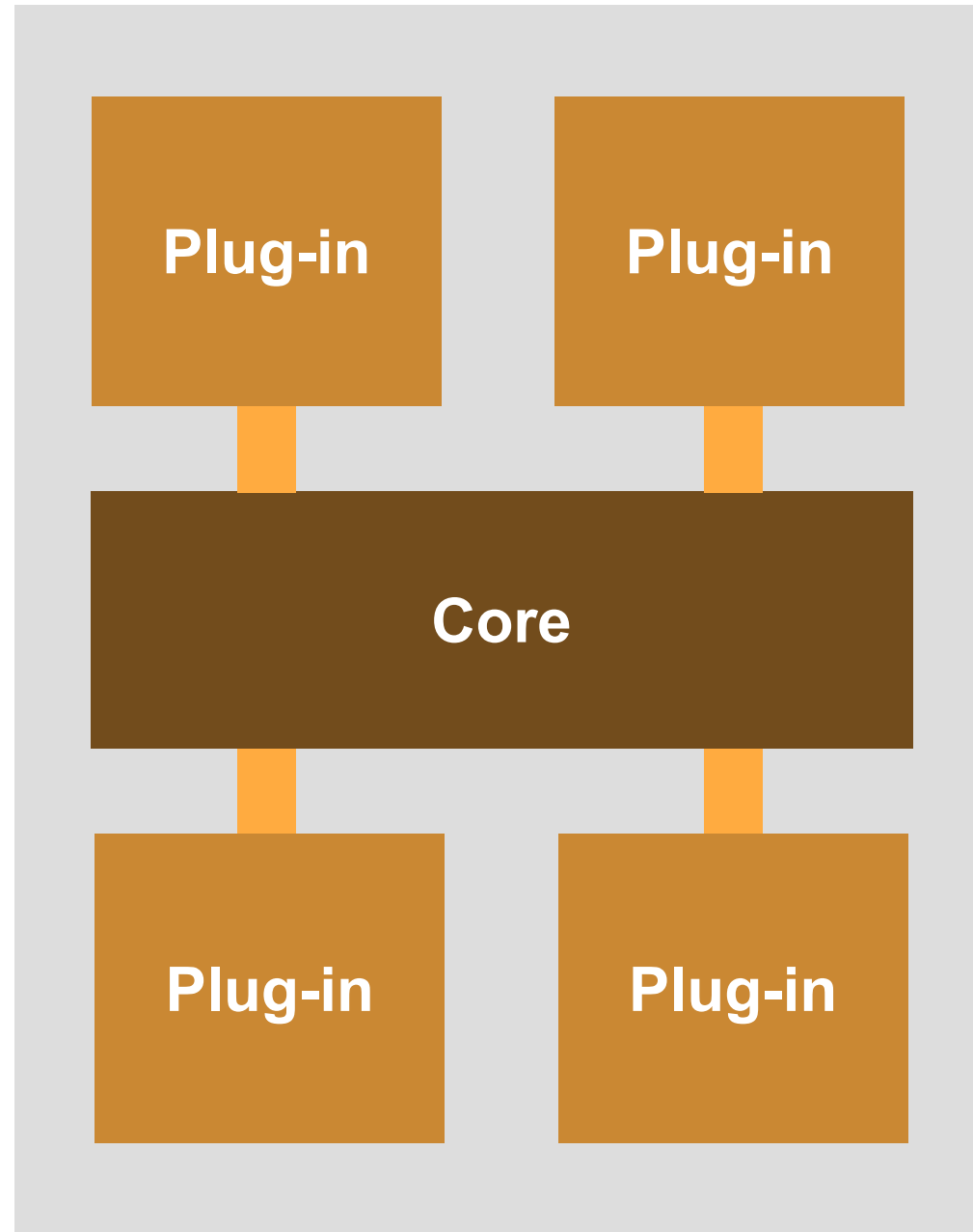
Microkernel. Примеры

- IntelliJ IDEA
- Jenkins
- Google Chrome



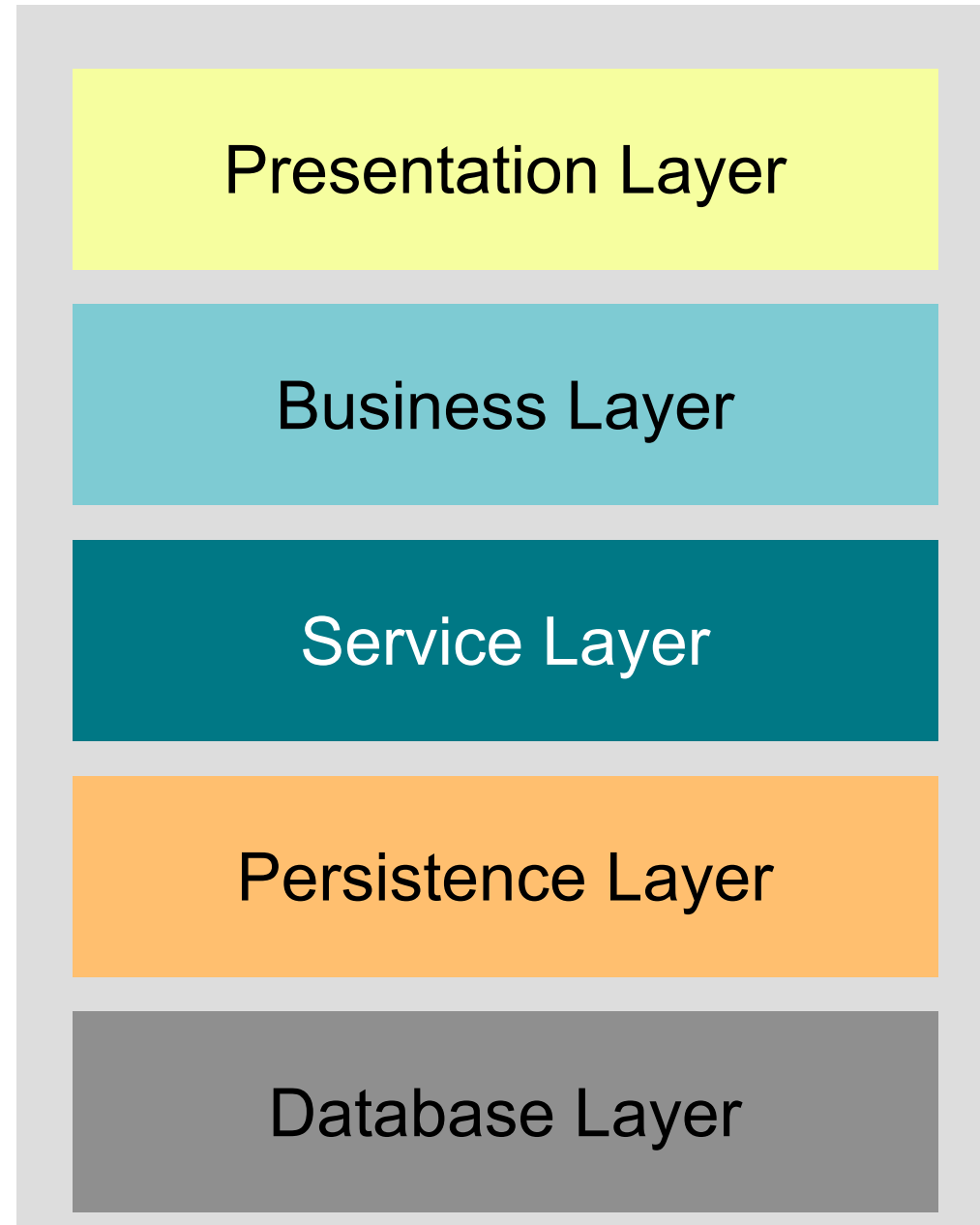
Microkernel. Когда использовать

- Вы являетесь вендором, а приложение — конечный продукт
- Приложение развёртывается в разных условиях
- Кажется, что нужны разные дистрибутивы, но ядро — общее



Layered (N-tier)

- Стандарт де-факто
  для большинства приложений
- Прост в разработке
- Хорошо ложится на оргструктуру
  большинства организаций
  (закон Конвея)

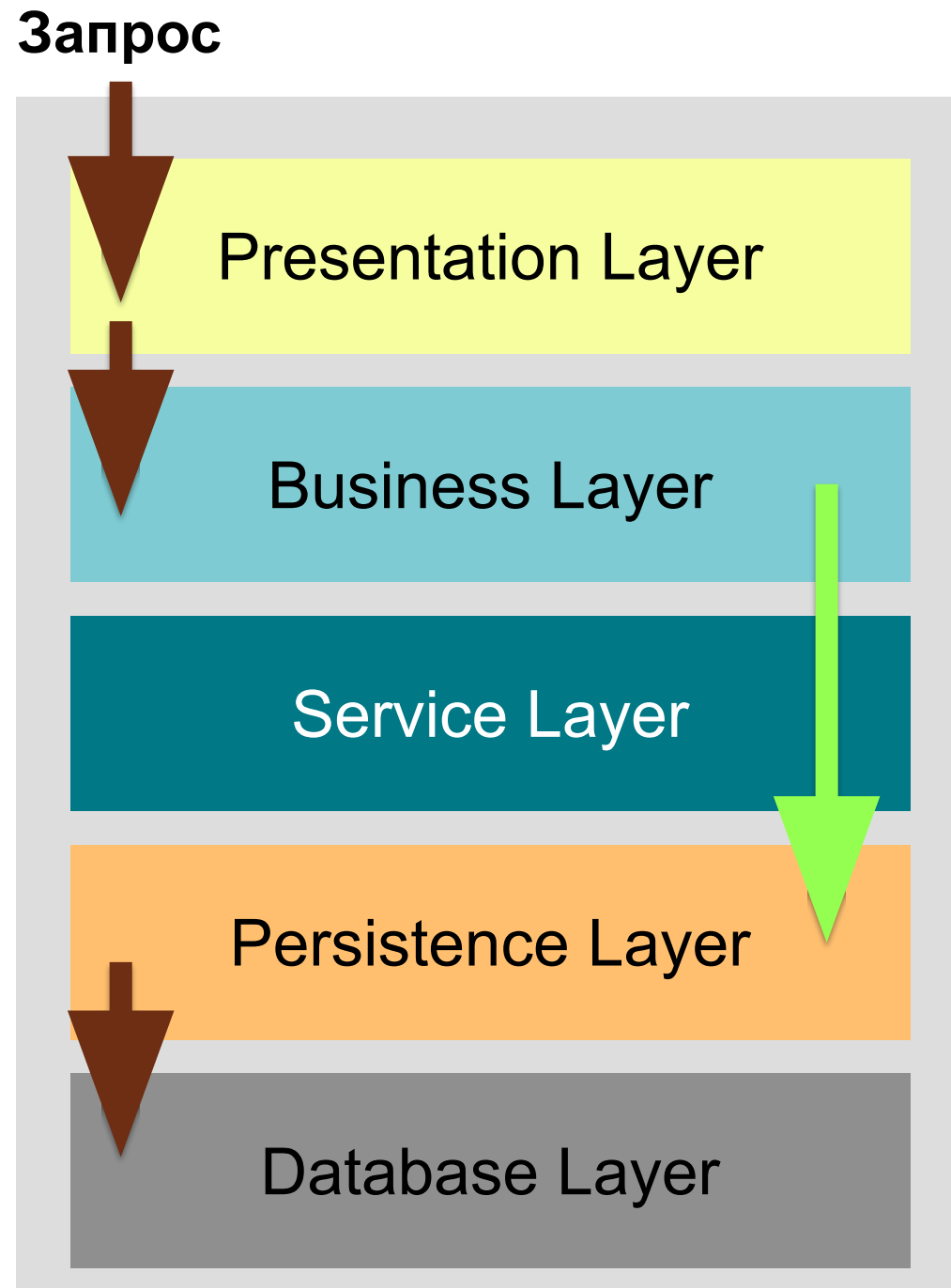


Layered. Изоляция слоёв

- Каждый слой может быть либо закрытым, либо открытым

Закрытый слой

Открытый слой

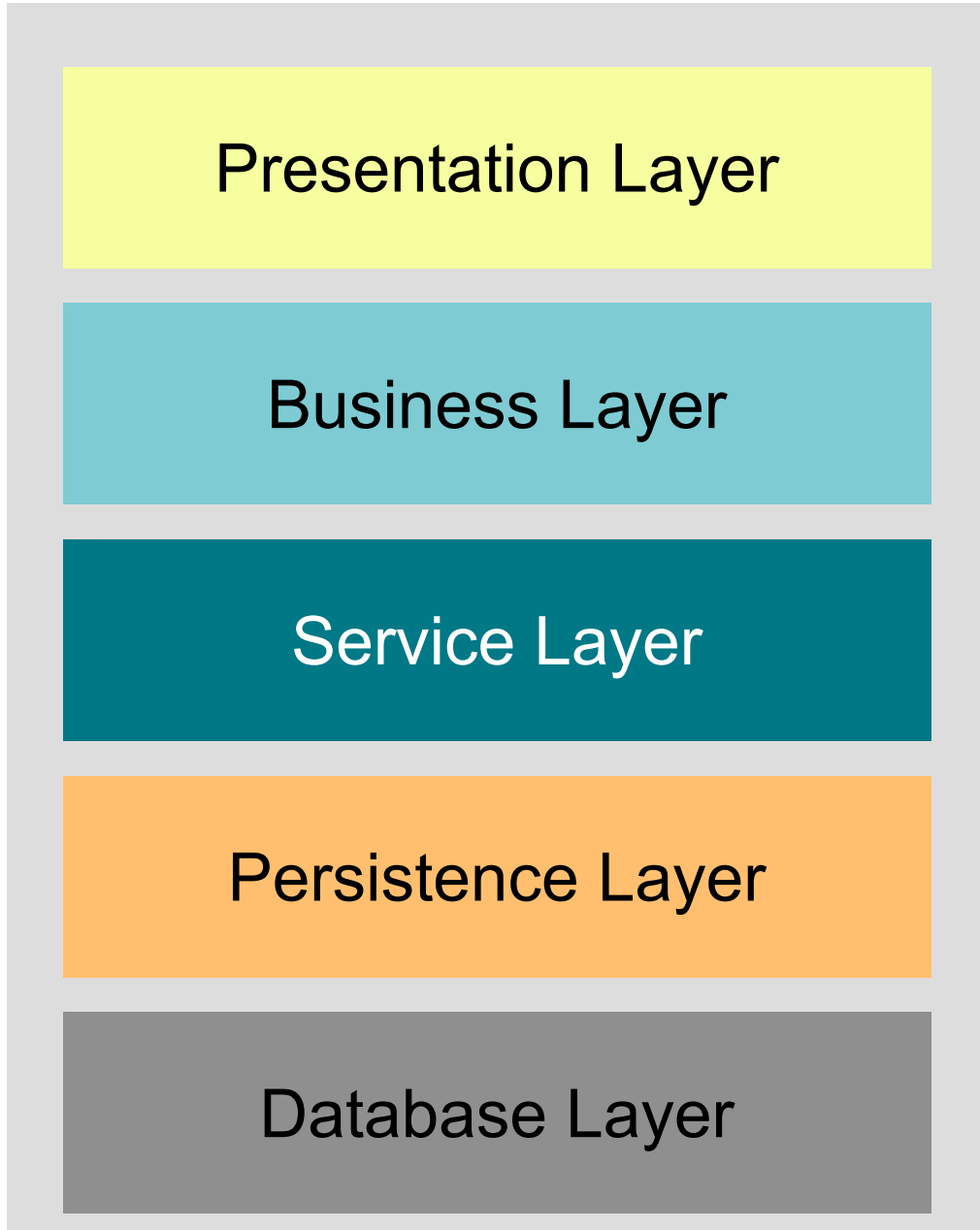


Skillbox

Sinkhole Anti-Pattern

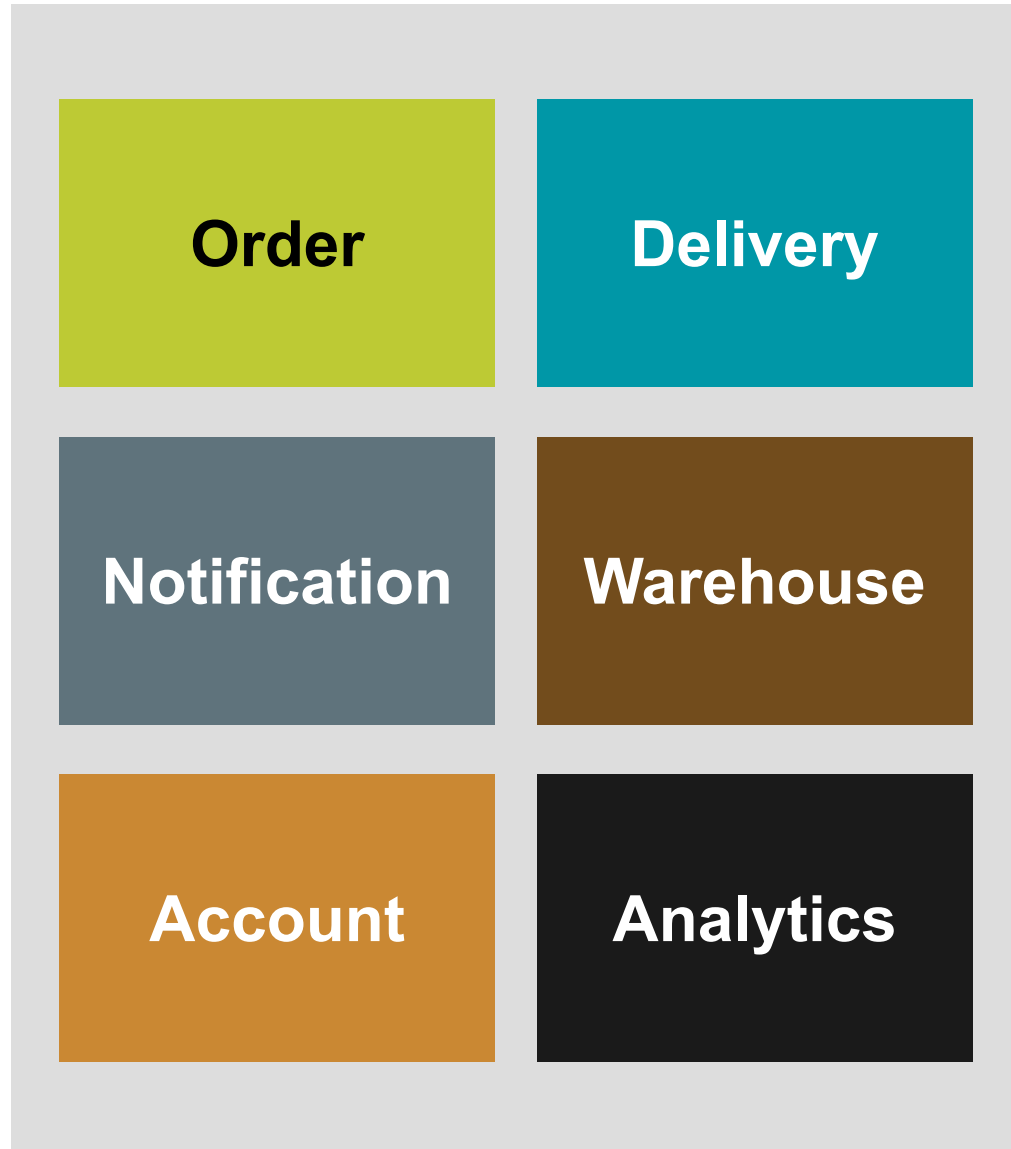
Layered.
Когда использовать

- Если вы не знаете,
  какую архитектуру выбрать
- Маленькие веб-приложения
- Нет чётких требований к системе



Modular

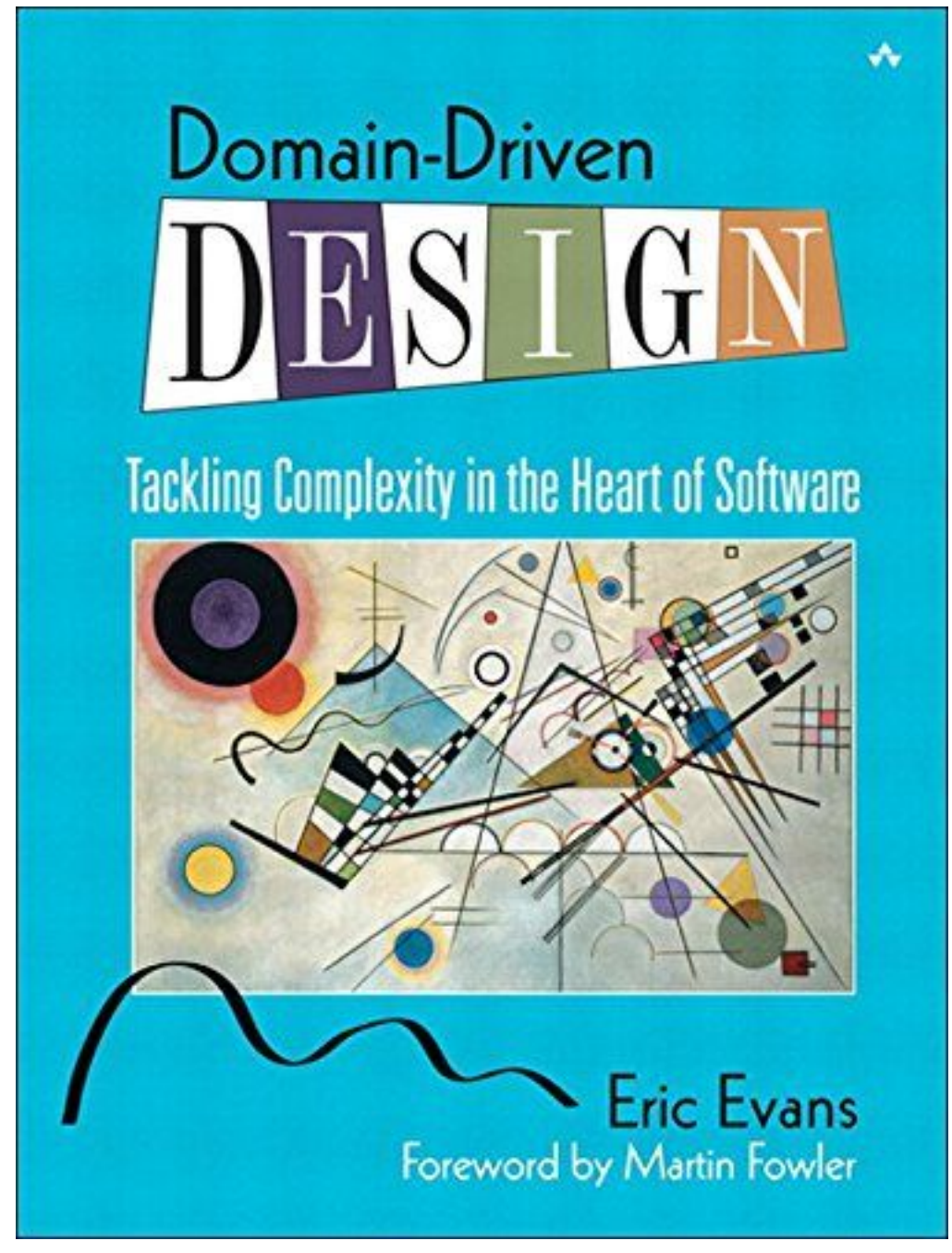
- Разделение кода по доменным областям бизнеса
- Ядро системы — бизнес-домены



Modular

Domain-Driven Design

By Eric Evans, 2003

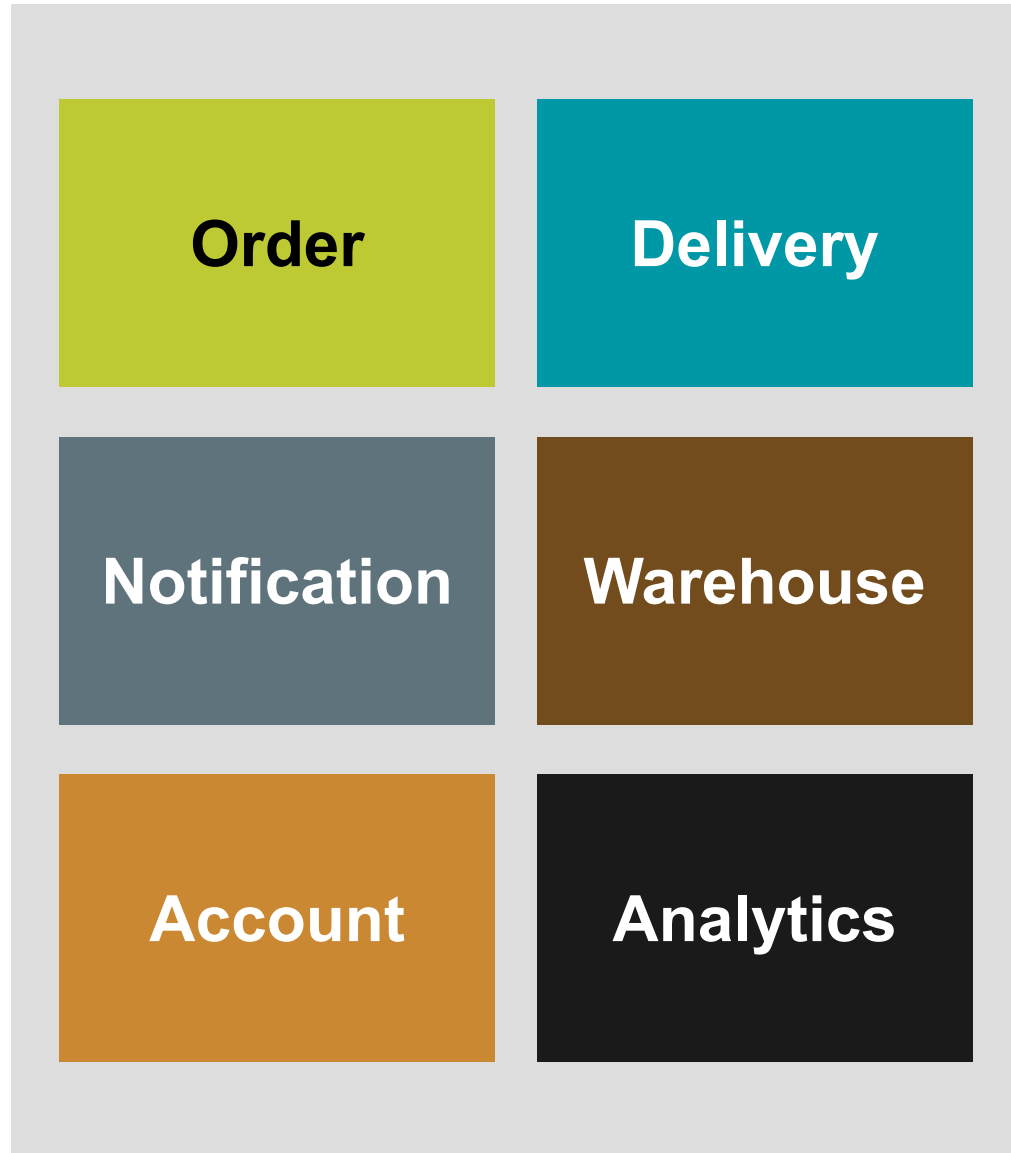


Modular

- Фокус в сторону бизнеса,
  а не технологии

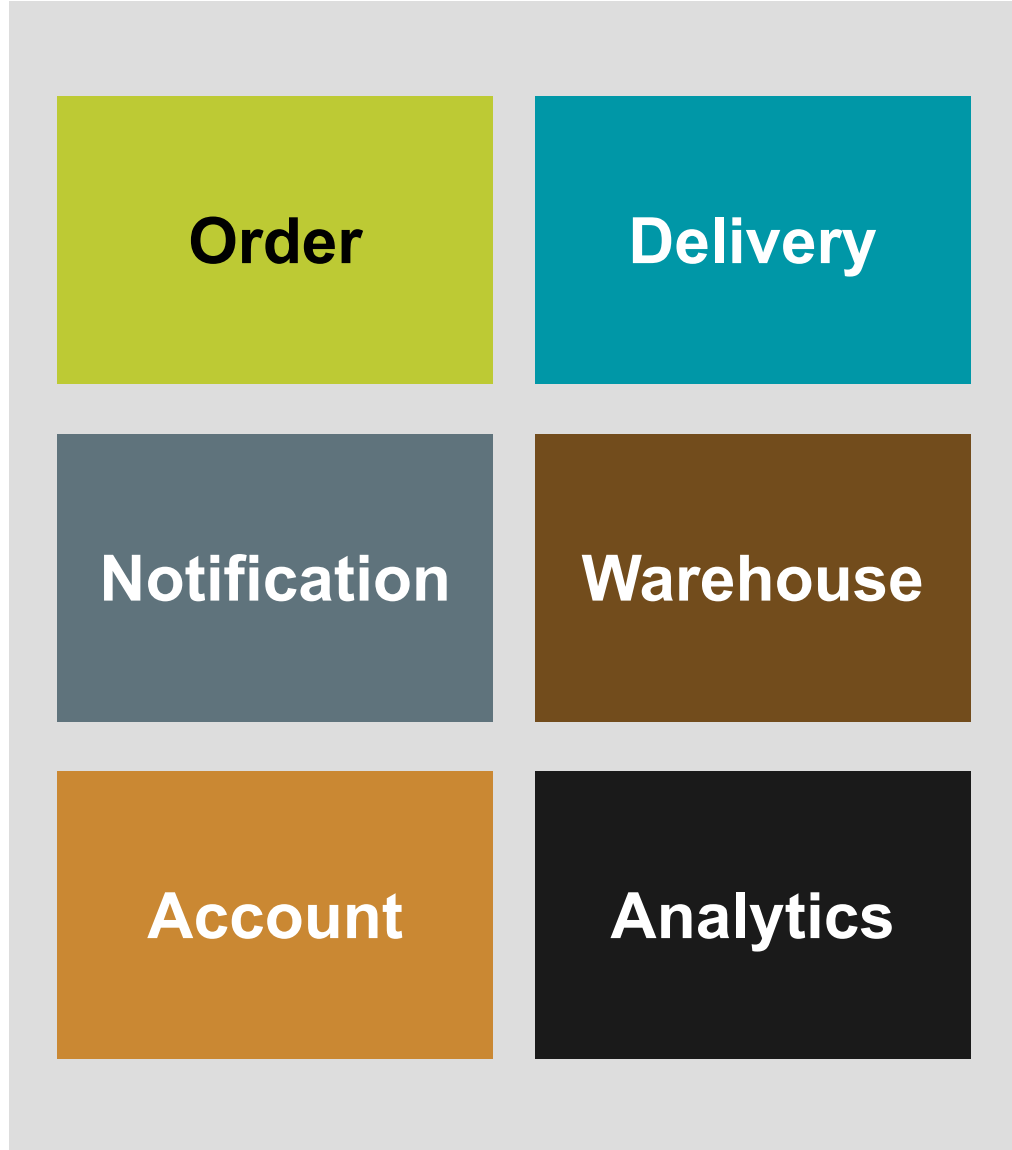
- Можно использовать обратный
  закон Конвея для организации
  кросс-функциональных команд

- Проще мигрировать
  в распределённую архитектуру



Modular.
Когда использовать

- Вы проектируете большое приложение со множеством команд
- В планах — переход на микросервисы



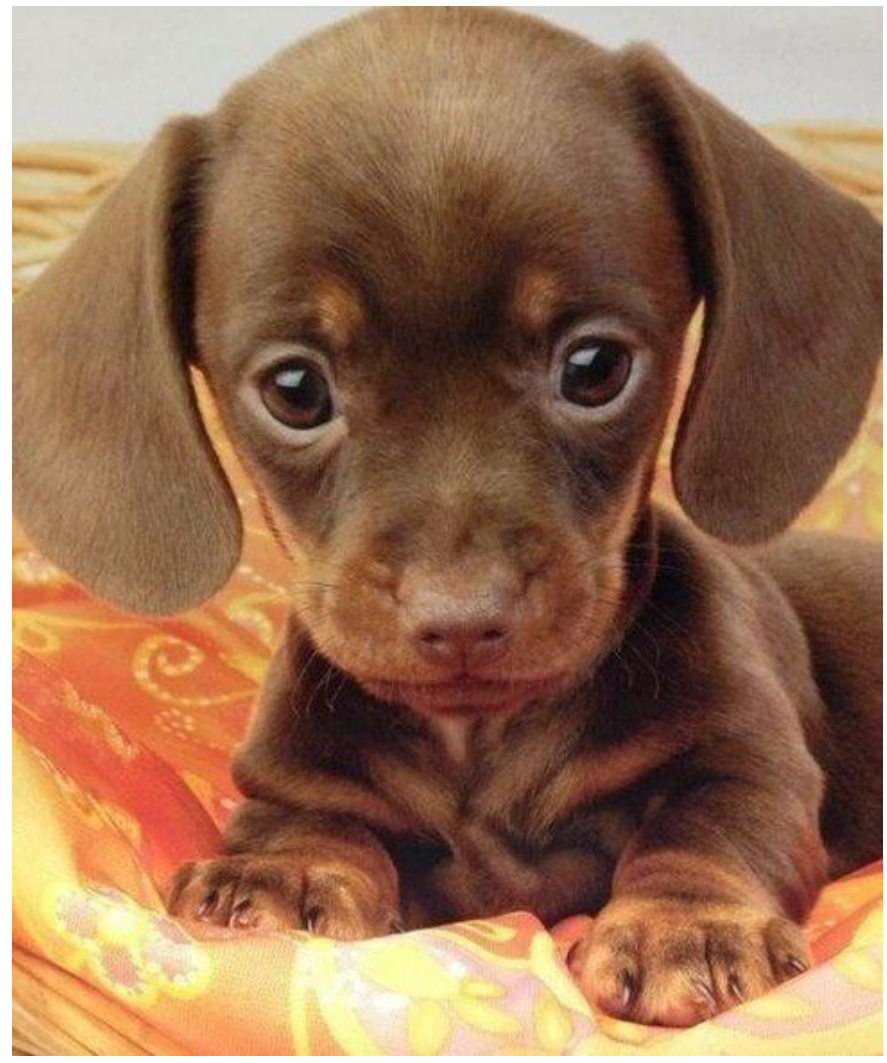
Skillbox

Монолит — это хорошо или плохо?

Это зависит от размеров монолита.

Пока он маленький

- ***Проще разрабатывать***
- ***Проще делать изменения***
- ***Проще тестировать***
- ***Проще развёртывать***
- ***Проще масштабировать***



Как только он подрос

- *Сложно разрабатывать*
- *Сложно делать изменения*
- *Сложно тестировать*
- *Сложно масштабировать*

Выводы

- Познакомились с архитектурными стилями построения монолитных приложений.
- Выяснили, что монолит — это не всегда плохо.

На следующем занятии

- Рассмотрим архитектуру текущего решения и то, что оно умеет.