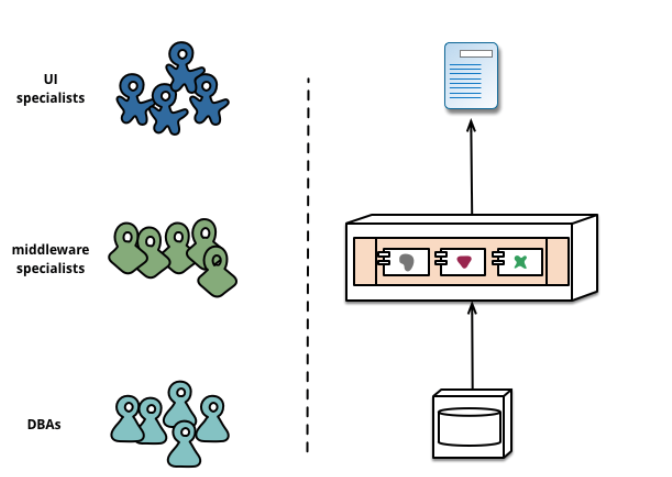
ПWS

ПОИТ-4

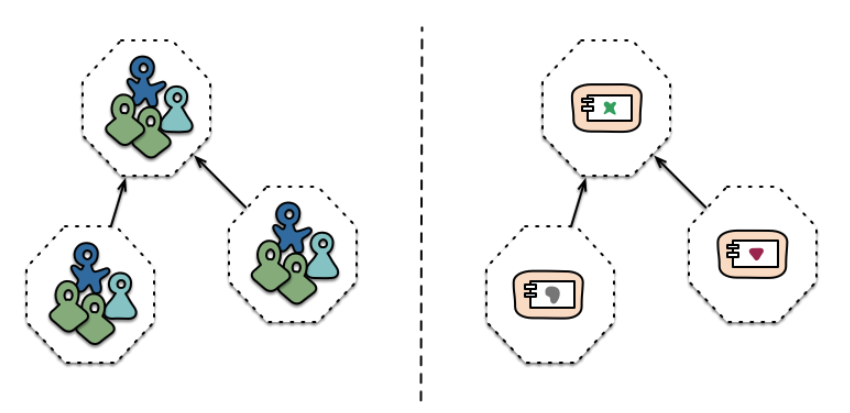
Лекция 15

**Микросервисы**

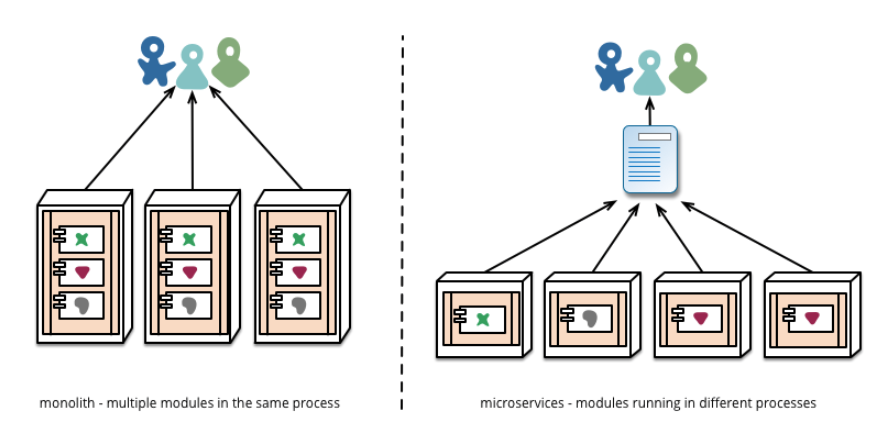
1. ***Правила существуют для того, чтобы им следовали дурачье и ими руководствовались умные люди. Дуглас Бадер.***
2. **SOA:** сервис-ориентированная архитектура; программное обеспечение представляет собой набор независимых компонентов, оснащенных стандартными интерфейсами (REST, RPC) для взаимодействия по стандартным протоколам. Технологии: DCOM, ASMX, WCF, CORBA, Java EE EJB. Альтернатива большим монолитным пиложениям.
3. **Микросервисы:** один из подходов к разработке SOA-приложений, основной принцип – сервисы должны быть легкими (сервисы маленькие и протоколы легковесные). Легкий сервис – сервис, который может быть переписан за 2 недели (Real Estate).
4. **Микросервисы:** цель: 1) устойчивость к сбоям; 2) облегчить понимание и поддержку кода; 3) усилить работу команды программистов.
5. **Микросервисы:** микросервис – сервис, выполняющий одну элементарную функцию; основной принцип разбиения – изменение сервиса не затрагивает другие сервисы.
6. **Микросервисы:** микросервисная архитектура - набор принципов, которым должны соответствовать сервисы. Задается или в форме правил или основывается на применении готовых фреймворков (например, Karyon, Dropwiard,…).
7. **Микросервисы:**  умные приемники и глупые каналы. Enterprise Service Bus (ESB) – не является приемлемым решением. Обычно применяется простой HTTP/REST, но необязательно. Есть легковесные шины для обеспечения асинхронности (MOM: RabbitMQ, ZeroMQ).
8. **Микросервисы:** HTTP+HATEOAS, XML/JSON.
9. **Микросервисы:** ошибки проектирования, ***закон*** ***Мелвина Конвея*** «Любая организация, которая проектирует какую-то систему, получит решение, структура которого копирует структуру коммуникаций этой организации».

****

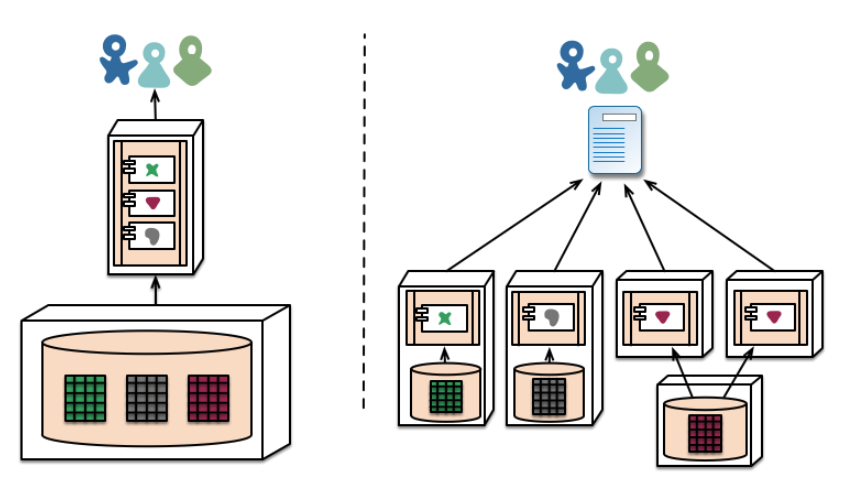
1. **Микросервисы:** правильное распределение функциональности на микросервисы в соответствием с ***потребностями бизнеса***. Каждый микросервис включает в себя полный набор технологий, но обеспечивает одну технологию. Организация разрабатывающая приложения с микросервисной архитектурой должны иметь ***кросс-функциональные*** команды.

****

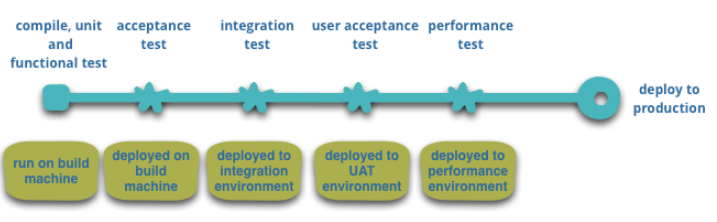
1. **Микросервисы:** распределенные процессы, монолит - несколько модулей в одном процессе; микросервисы – каждый модуль в отдельном удаленном процессе.

****

1. **Микросервисы:** применение подходящего инструментария для разработки микросервисов (дает возможность привлекать разные команды).
2. **Микросервисы:** применение микросервисной архитектуры, как правило, приводит к необходимости поддержки системы разработчиками (правило «вы разработали, вам и поддерживать», Amazon),
3. **Микросервисы:** тенденция, передача разработчикам – поддержку микросервисов (Amazon, Netflix).
4. **Микросервисы:** популярные паттерны ***Tolerant Reader***, ***Consumer Driver***.
5. **Микросервисы:** синхронные вызовы сервисов считаются опасными; как правило, синхронные вызовы не применяются или «один синхронный запрос к сервису на один пользовательский запрос».
6. **Микросервисы:** распределенное управление данными. Подход ***Polyglot Persistence***.

****

1. **Микросервисы:** проблема – транзакции. Часто без применения транзакций. Микросервисы в основном на чтение, а пишет отдельный сервис.
2. **Микросервисы:** проектирование под отказ; разработка систем мониторинга работоспособности сервисов; разработка систем протоколирования работы сервисов; консоль сервиса: статус, тестирование, получение различных метрик.
3. **Микросервисы:** клиенты работающие с микросервисом могут реализованы с помощью ***оркестрового*** (прямые вызовы сервисов) или ***хореографического*** (сервисы подписываются на события клиента) принципов;
4. **Микросервисы:** требуется ***DevOps*** (Development & Operation)- набор технологий нацеленных на интеграцию процессов разработки и информационно-техническому обслуживанию. Цели DevOps: сокращение выхода продукта на рынок, снижение частоты отказов релизов, сокращение времени на изменения, сокращение времени на восстановление. Задача DevOps сделать согласованным процесс разработки и эксплуатации приложений. Основные задачи DevOps: контроль версий, непрерывная сборка, непрерывное тестирование, поддержка репозиториев артефактов, конфигурация инфраструктуры, мониторинг работоспособности и производительности. DevOps – следствие увеличения релизов (Agile-технологии) и усложнения инфраструктуры.
5. **Микросервисы:** автоматизация, продукты DevOps: ***Continuous Delivery***, ***Continuous Integration***.



1. **Микросервисы:** все новое – хорошо забытое старое.
2. **рр**
3. **рр**
4. **рр**
5. **мм**