Архитектура программного обеспечения

Школа лингвистики, ФГН, НИУ ВШЭ
Продвинутый Питон, 15.04.2020

План лекции

1. Место архитектуры при разработке программного обеспечения
2. Виды архитектур программного обеспечения
3. Что такое хорошая архитектура?

Место архитектуры в разработке ПО

Архитектура - это **набор значимых решений** по поводу организации системы программного обеспечения, набор **структурных элементов** и их интерфейсов, при помощи которых компонуется система, вместе с их **поведением**, определяемым во взаимодействии между этими элементами, **компоновка** элементов в постепенно укрупняющиеся подсистемы, а также **стиль архитектуры** который направляет эту организацию - элементы и их интерфейсы, взаимодействия и компоновку.

https://www.ibm.com/developerworks/ru/library/eeles/

Место архитектуры в разработке ПО

1. Архитектура определяет структуру
2. Архитектура определяет поведение
3. Архитектура концентрируется на значимых элементах
4. На архитектуру оказывает влияние ее окружение
5. Архитектура оказывает влияние на структуру коллектива
6. Архитектура может соответствовать некоторому архитектурному стилю

https://www.ibm.com/developerworks/ru/library/eeles/

Место архитектуры в разработке ПО

Архитектура определяет структуру

При создании архитектуры необходимо определить основные блоки, из которых состоит система, и связи между ними.

Место архитектуры в разработке ПО

Архитектура определяет поведение

Связи между блоками определяют направление, протоколы и форматы передачи данных. В зависимости от этих трех параметров, а также архитектуры системы в целом, поведение системы будет меняться.

Место архитектуры в разработке ПО

Архитектура концентрируется на значимых элементах

Архитектура - это вид с самого верха. Как следствие, при создании архитектуры необходимо сосредоточиться на наиболее крупных фрагментах системы и их взаимодействии.

Место архитектуры в разработке ПО

На архитектуру оказывает влияние ее окружение

Система должна взаимодействовать с внешним миром, то есть в архитектуре системы должны быть связи с внешним миром. Иногда внешний мир может диктовать протоколы и форматы взаимодействия с ним. Как следствие, архитектуру системы необходимо привязывать к этому взаимодействию.

Место архитектуры в разработке ПО

Архитектура оказывает влияние на структуру коллектива

Обычно архитектура системы составляется из независимых блоков, взаимодействующих между собой по некоторому интерфейсу. Следовательно, и разрабатываться эти блоки могут независимо отдельными командами.

Место архитектуры в разработке ПО

Архитектура может соответствовать некоторому архитектурному стилю

За долгую историю разработки программного обеспечения было придумано много шаблонов для архитектуры программных систем. Сейчас мы разберем некоторые из них.

Архитектурные стили

- Монолитная
- Многослойная
- Компонентная
  - Процедурная
  - Объектно-ориентированная
  - Проблемно-ориентированная

См.
https://www.intuit.ru/studies/courses/64/64/lecture/1878?page=3
https://studref.com/320288/informatika/arhitekturnye_stili_proektirovaniya

Архитектурные стили

- Конвейер обработки данных
  - Пакетная обработка
  - Канал сообщений
  - Каналы и фильтры
- Вызов-возврат
  - Клиент-серверная архитектура
  - Сервисно-ориентированная
  - Многоуровневая архитектура
- Интерактивные системы
  - Model-View-Controller
  - Presentation-Abstraction-Control
- Системы на основе хранилища данных
  - Репозиторий
  - Классная доска

Архитектурные стили

Монолитная система

Скорее антишаблон проектирования: составные блоки не выделяются, протоколы взаимодействия не нужны, форматы данных какие придутся.

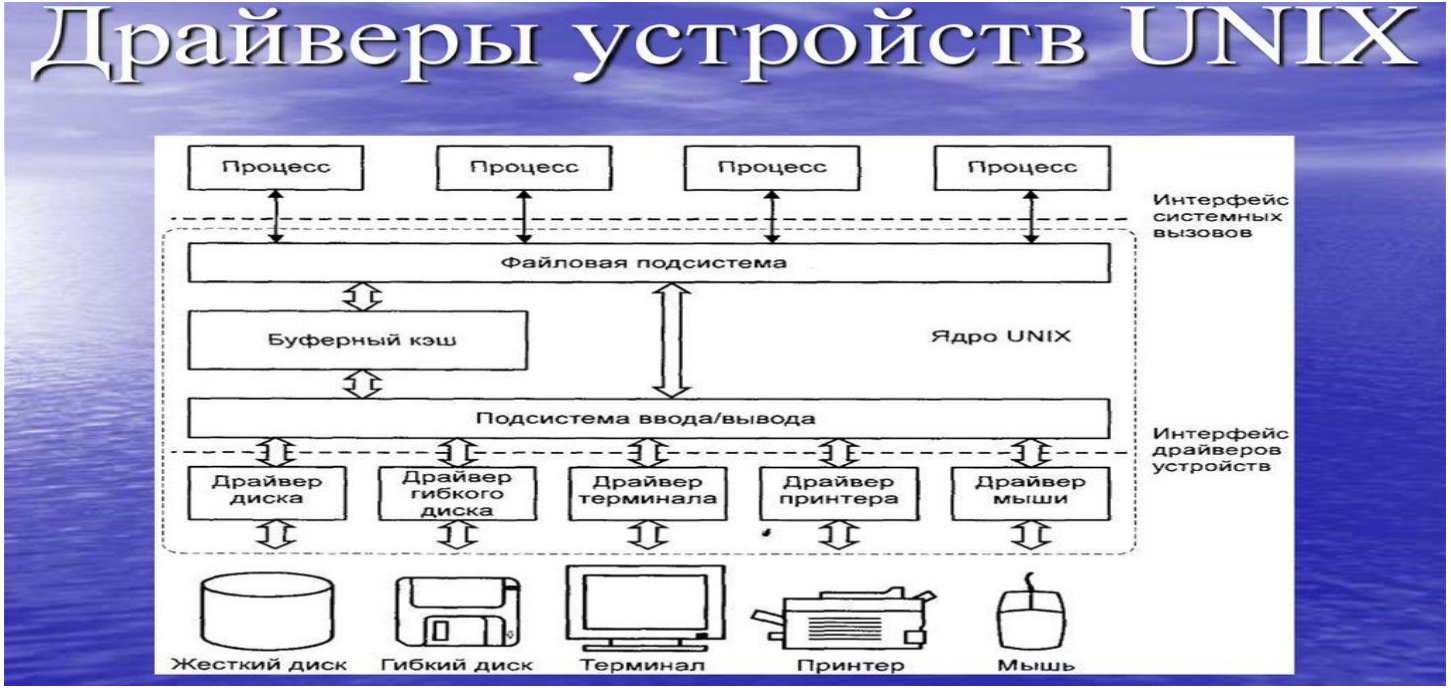
Архитектурные стили

Многослойная система

Имеется естественное расслоение задач системы на наборы задач, которые можно было бы решать последовательно. Компоненты разделяются на несколько уровней таким образом, что компоненты данного уровня могут использовать для своей работы только соседей или компоненты предыдущего уровня. Могут быть более слабые ограничения, например, компонентам верхних уровней разрешено использовать компоненты всех нижележащих уровней

Архитектурные стили

Многослойная система (пример)



Архитектурные стили

Многослойная система (пример)

https://habr.com/ru/company/it-grad/blog/276297/

Архитектурные стили

Компонентная система

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

Архитектурные стили

Компонентная система (комментарий)

На самом деле, многослойная система является разновидностью компонентной, просто компоненты в ней упорядочены по слоям.

Архитектурные стили

Компонентная система (пример)

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

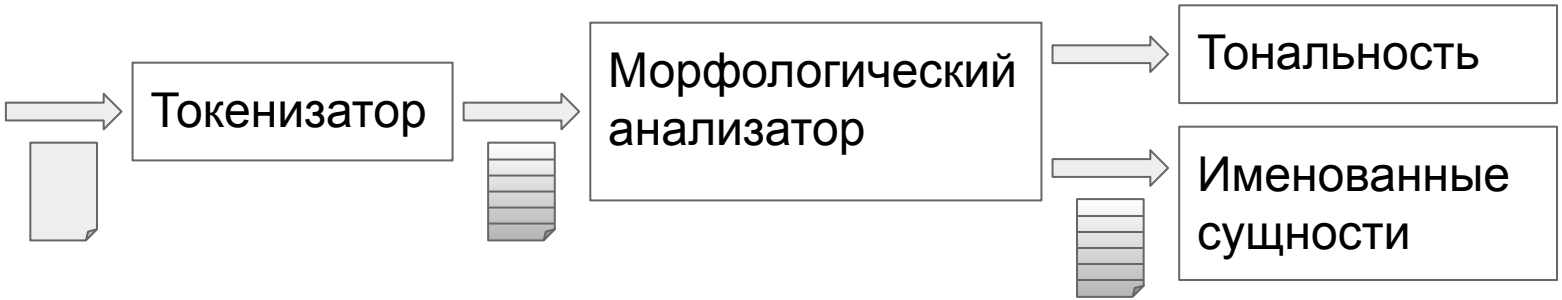
Система анализа текстов состоит из токенизатора, морфологического анализа, модулей выделения именованных сущностей, анализа тональности,...

Оформим их в виде компонент, которые можно будет повторно использовать в следующих проектах, оговорим интерфейс взаимодействия между ними, объединим в единую систему.

Архитектурные стили

Компонентная система (пример)

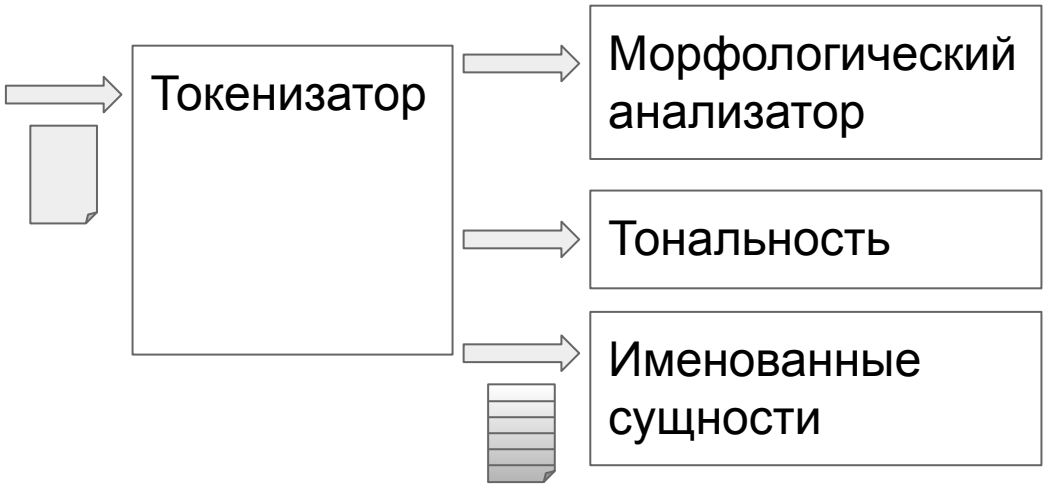
Система анализа текстов состоит из токенизатора, морфологического анализа, модулей выделения именованных сущностей, анализа тональности,...



Архитектурные стили

Компонентная система (альтернативный пример)

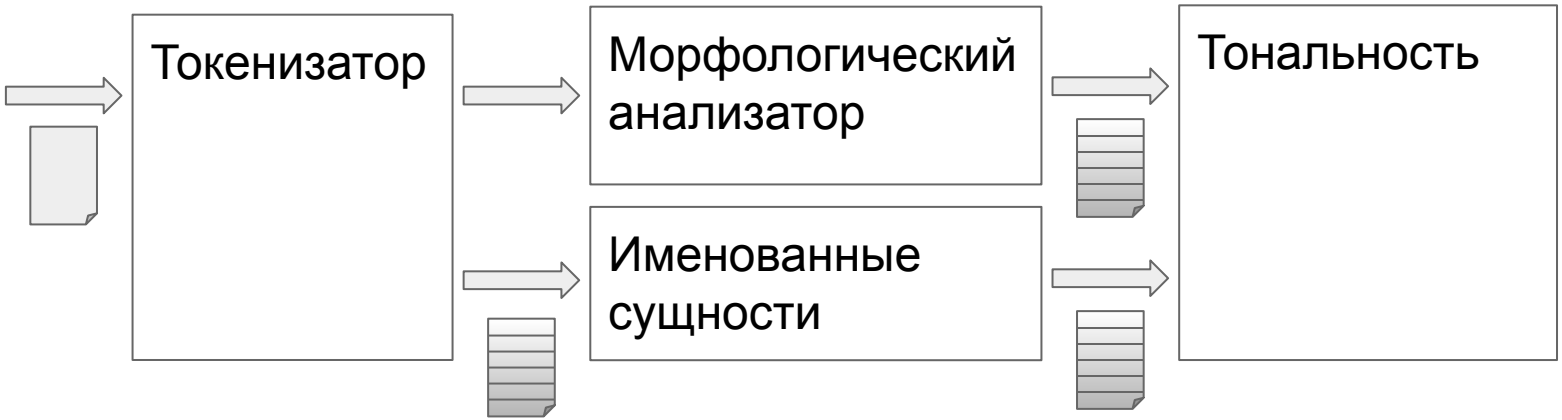
Система анализа текстов состоит из токенизатора, морфологического анализа, модулей выделения именованных сущностей, анализа тональности,...



Архитектурные стили

Компонентная система (альтернативный пример)

Система анализа текстов состоит из токенизатора, морфологического анализа, модулей выделения именованных сущностей, анализа тональности,...



Архитектурные стили

Компонентная система (нюансы)

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

Создание повторно используемого модуля примерно в два раза затратнее: необходимо лучше продумать интерфейс модуля, написать документацию и комментарии, расширить систему тестов… На это сложно найти время в режиме цейтнота. Но третье использование модуля окупит его создание. Мудрость состоит в том, чтобы отличить разовую утилиту от полезного модуля.

Архитектурные стили

Компонентная система (процедурная)

Данные неизменны, процедуры работы с ними могут немного меняться, могут возникать новые. Выделяется набор процедур, схема передачи управления между которыми представляет собой дерево с основной процедурой в его корне.

Архитектурные стили

Компонентная система (абстрактные данные)

В системе много данных, структура которых может меняться. Важны возможности внесения изменений и интеграции с другими системами.

Выделяется набор абстрактных типов данных, каждый из которых предоставляет набор операций для работы с данными такого типа. Внутреннее представление данных скрывается.

Архитектурные стили

Компонентная система (объектно-ориентированная)

Парадигма проектирования, основанная на разделении ответственностей приложения или системы на самостоятельные пригодные для повторного использования объекты, каждый из которых содержит данные и поведение (методы), относящиеся к этому объекту. При объектно-ориентированном проектировании система рассматривается не как набор подпрограмм и процедурных команд, а как наборы взаимодействующих объектов. Объекты обособлены, независимы и слабо связаны. Обмен данными между ними происходит через интерфейсы путем вызова методов (свойств) других объектов и отправки (приема) сообщений.

Архитектурные стили

Компонентная система (проблемно-ориентированное проектирование)

Подход к проектированию ПО, основанный на предметной области, ее элементах, их поведении и отношениях между ними. Целью является создание программных систем, реализующих модель предметной области, выраженной на языке специалистов в этой области. Модель предметной области может рассматриваться как каркас, на основе которого будут реализовываться программные решения.

Архитектурные стили

- Конвейер обработки данных
  - Пакетная обработка
  - Каналы и фильтры
  - Канал сообщений
- Вызов-возврат
  - Клиент-серверная архитектура
  - Сервисно-ориентированная
  - Многоуровневая архитектура
- Интерактивные системы
  - Model-View-Controller
  - Presentation-Abstraction-Control
- Системы на основе хранилища данных
  - Репозиторий
  - Классная доска

Архитектурные стили

Конвейер обработки данных

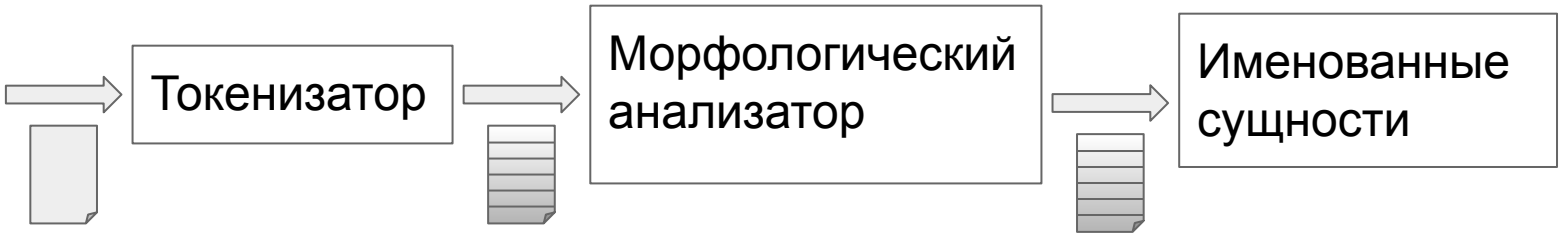
Система выдает четко определенные выходные данные в результате обработки четко определенных входных данных, при этом процесс обработки не зависит от времени, применяется многократно, одинаково к любым данным на входе. Обработка организуется в виде набора (необязательно последовательности) отдельных компонентов-обработчиков, передающих свои результаты на вход другим обработчикам или на выход всей системы.

Важными свойствами являются четко определенная структура данных и возможность интеграции с другими системами.

Архитектурные стили

Конвейер обработки данных (пакетная обработка)

Один-единственный вывод производится на основе чтения некоторого одного набора данных на входе, промежуточные преобразования организуются в виде последовательности.



Архитектурные стили

Конвейер обработки данных (каналы и фильтры)

Нужно обеспечить преобразование непрерывных потоков данных. При этом преобразования инкрементальны и следующее может быть начато до окончания предыдущего. Имеется, возможно, несколько входов и несколько выходов.

В дальнейшем возможно добавление дополнительных преобразований.

Архитектурные стили

Конвейер обработки данных (каналы и фильтры - пример)

Мы хотим создать универсальный движок, в котором пользователь сам может создавать последовательность по обработке текста из стандартных блоков. Если блоки оформить стандартным образом, так, чтобы была известна информация о их входах и выходах, то можно состыковывать их между собой когда это разрешено.

Архитектурные стили

Конвейер обработки данных (каналы и фильтры - пример)

Требования к такой системе:

- блоки должны возвращать информацию о типах входа и выхода;
- → стандартный интерфейс у всех блоков;
- необходимо хранить онтологию или тезаурус данных;
- → все типы данных выводятся из одного со стандартным интерфейсом;
- ...

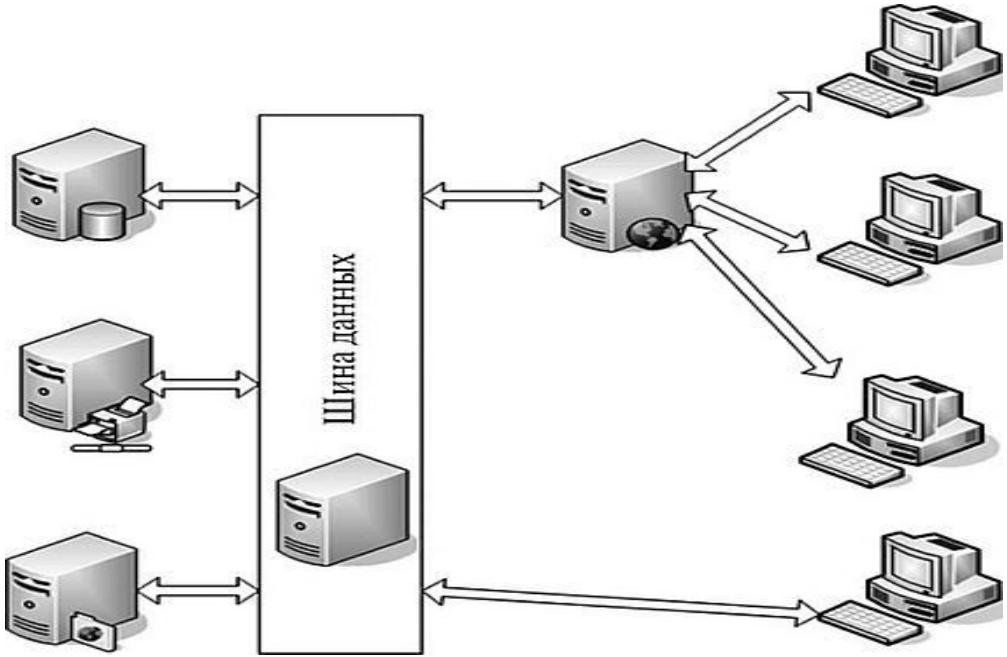
Архитектурные стили

Конвейер обработки данных (канал сообщений)

Может принимать и отправлять сообщения по одному или более каналам связи, обеспечивая таким образом приложениям возможность взаимодействия без необходимости знания конкретных деталей друг о друге. Взаимодействия между приложениями осуществляются путем передачи сообщений (обычно асинхронной — отправитель сообщения не дожидается результата его обработки получателем) через «общую шину». Шины данных используются для обеспечения сложных правил обработки данных уже давно. Такой дизайн обеспечивает архитектуру, которая позволяет улучшить масштабируемость системы, вводить приложения в процесс обработки, подключая к шине несколько экземпляров одного и того же приложения.

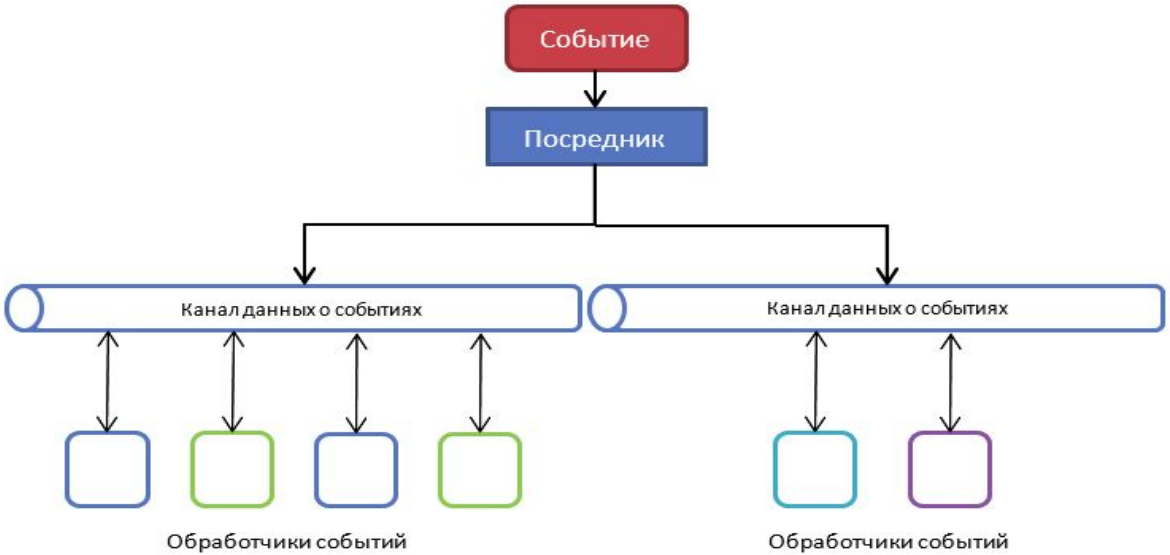
Архитектурные стили

Конвейер обработки данных (канал сообщений)



Архитектурные стили

Конвейер обработки данных (канал сообщений / диспетчер событий)



Архитектурные стили

Вызов-возврат

Порядок выполнения действий четко определен, отдельные компоненты не могут выполнять полезную работу, не получая обращения от других.

(На самом деле я всё расположил так, что этот принцип не всегда будет выполняться.)

Архитектурные стили

Вызов-возврат (клиент-сервер)

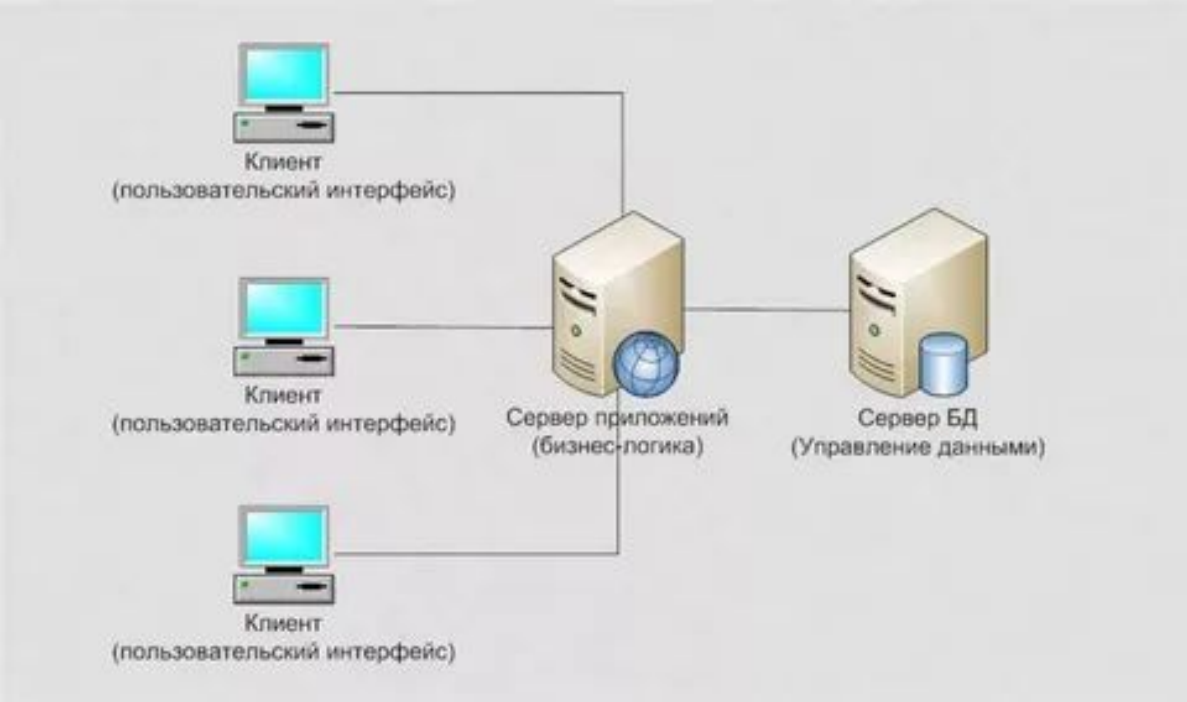
Решаемые задачи естественно распределяются между инициаторами и обработчиками запросов, возможно изменение внешнего представления данных и способов их обработки.

Архитектурные стили

Вызов-возврат (клиент-сервер - двухзвенная архитектура)

Архитектурные стили

Вызов-возврат (клиент-сервер - трехзвенная архитектура)



Архитектурные стили

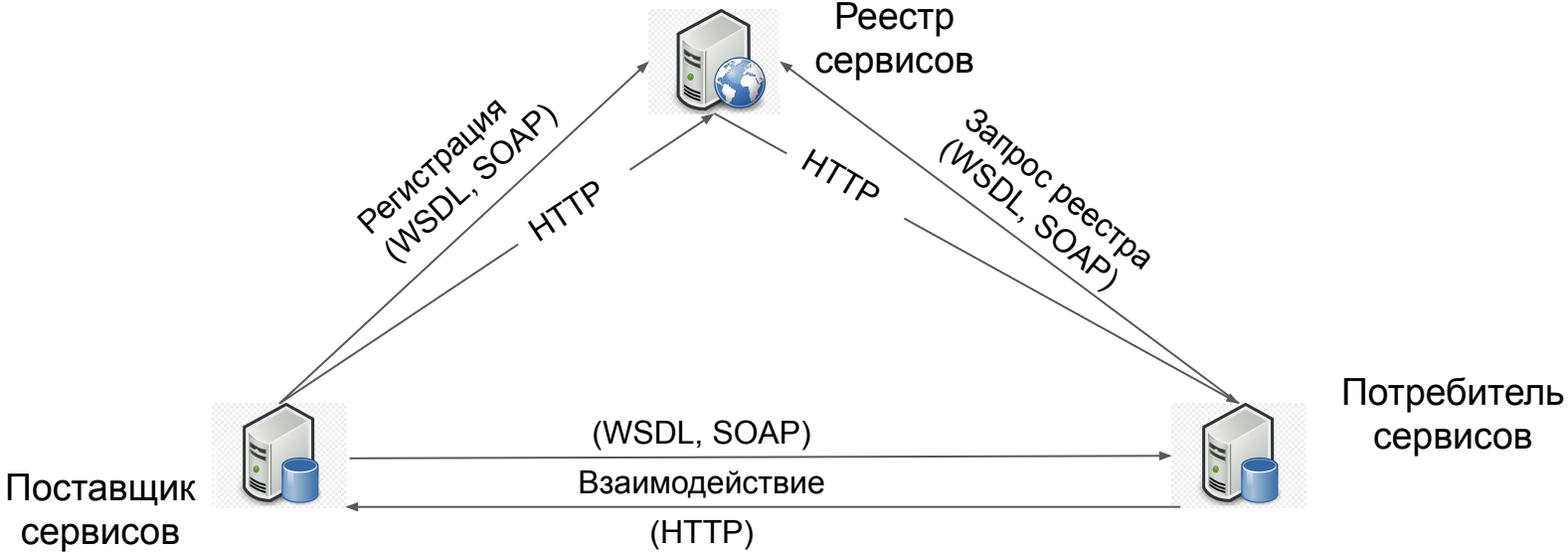
Вызов-возврат (сервисно-ориентированная)

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

Сервисы слабо связаны, используют интерфейсы, основанные на определенных стандартах, могут быть опубликованы, обнаружены и вызваны. Основная задача сервисов — предоставление взаимодействия с приложением посредством сообщений через интерфейсы, областью действия которых является приложение, а не компонент или объект.

Архитектурные стили

Вызов-возврат (сервисно-ориентированная)



Архитектурные стили

Вызов-возврат (микросервисная) https://habr.com/ru/post/249183/

Приложение строится как набор небольших сервисов, каждый из которых работает в собственном процессе и коммуницирует с остальными используя легковесные механизмы, как правило HTTP. Эти сервисы построены вокруг бизнес-потребностей и развертываются независимо с использованием полностью автоматизированной среды. Существует абсолютный минимум централизованного управления этими сервисами. Сами по себе эти сервисы могут быть написаны на разных языках и использовать разные технологии хранения данных.

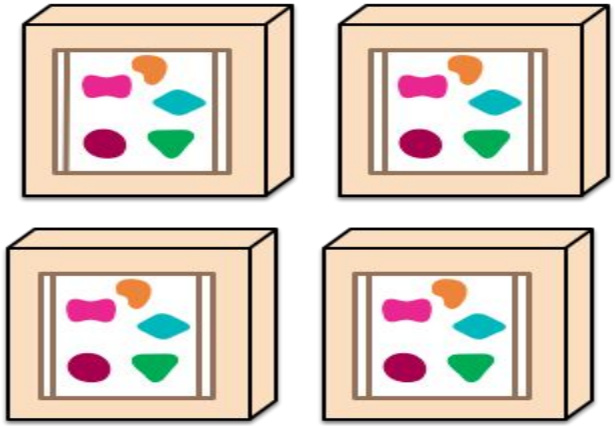
https://habr.com/ru/post/249183/

Архитектурные стили

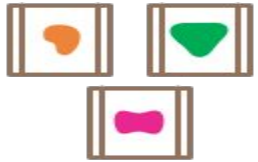
Вызов-возврат (микросервисная)

A monolithic application puts all its functionality into a single process...

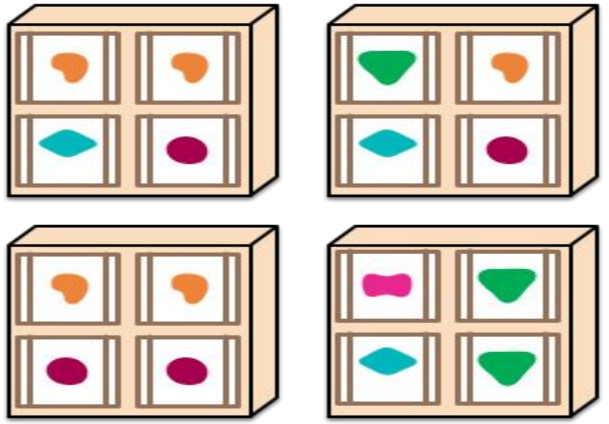
... and scales by replicating the monolith on multiple servers



A microservices architecture puts each element of functionality into a separate service...

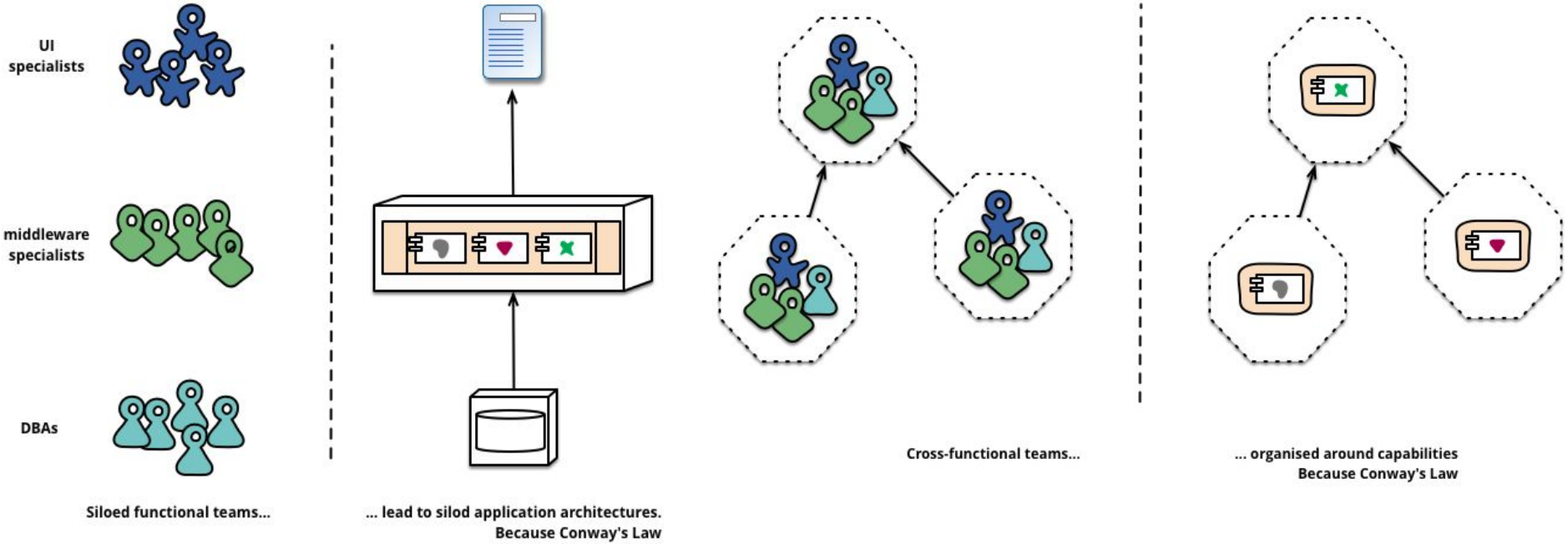


... and scales by distributing these services across servers, replicating as needed.



Архитектурные стили

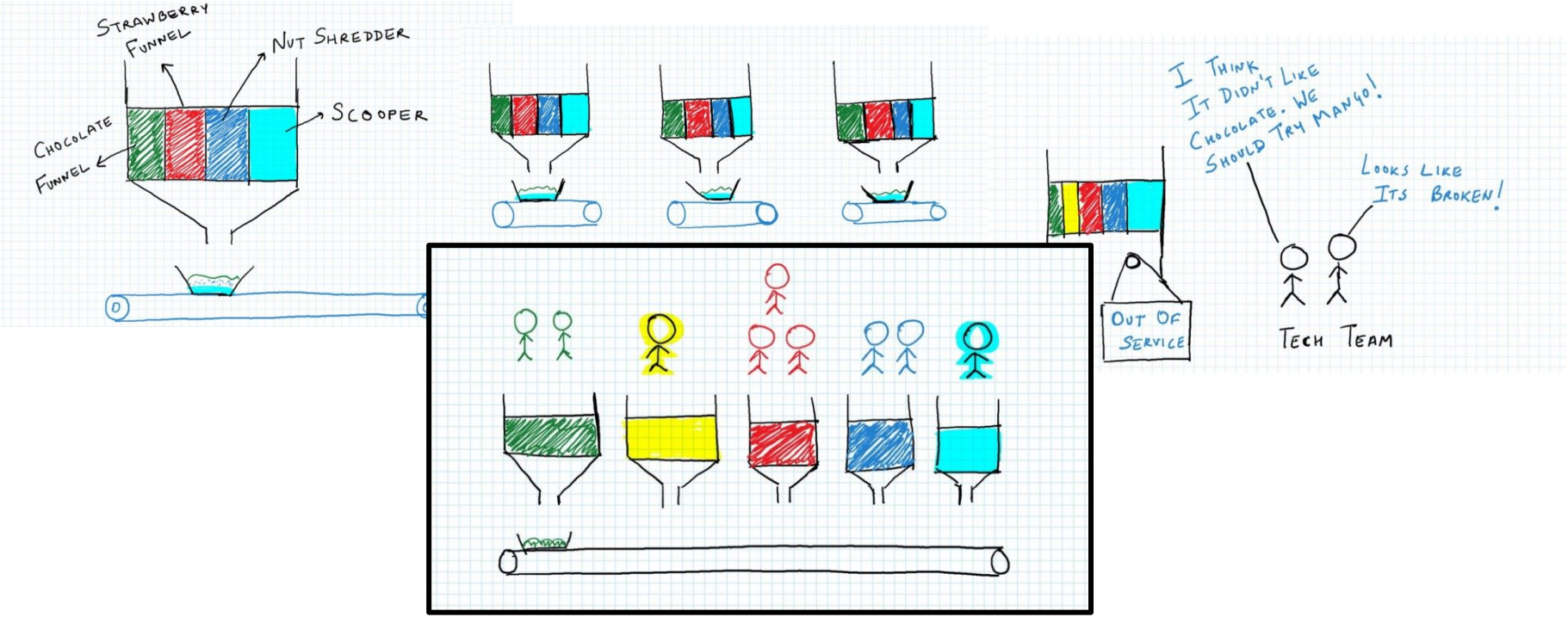
Вызов-возврат (микросервисная)



Архитектурные стили

Вызов-возврат (микросервисная)

https://nuancesprog.ru/p/4102/



Архитектурные стили

Интерактивные системы

Необходимость достаточно быстро реагировать на действия пользователя, изменчивость пользовательского интерфейса.

Архитектурные стили

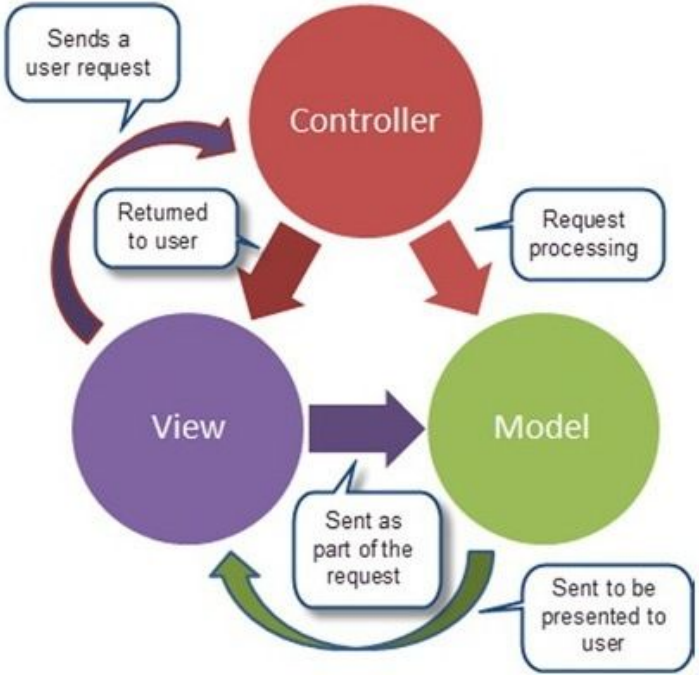
Интерактивные системы (данные–представление–обработка (model–view-controller, MVC))

Изменения во внешнем представлении достаточно вероятны, одна и та же информация представляется по-разному в нескольких местах, система должна быстро реагировать на изменения данных.

Выделяется набор компонентов, ответственных за хранение данных, компоненты, ответственные за их представления для пользователей, и компоненты, воспринимающие команды, преобразующие данные и обновляющие их представления.

Архитектурные стили

Интерактивные системы (MVC)



Архитектурные стили

Системы на основе хранилища данных

Основные функции системы связаны с хранением, обработкой и представлением больших количеств данных.

Архитектурные стили

Системы на основе хранилища данных (репозиторий)

Порядок работы определяется только потоком внешних событий.

Выделяется общее хранилище данных — репозиторий. Каждый обработчик запускается в ответ на соответствующее ему событие и как-то преобразует часть данных в репозитории.

Архитектурные стили

Системы на основе хранилища данных (классная доска)

Способ решения задачи в целом неизвестен или слишком трудоемок, но известны методы, частично решающие задачу, композиция которых способна выдавать приемлемые результаты, возможно добавление новых потребителей данных или обработчиков.

Отдельные обработчики запускаются, только если данные репозитории для их работы подготовлены. Подготовленность данных определяется с помощью некоторой системы шаблонов. Если можно запустить несколько обработчиков, используется система их приоритетов.

Архитектурные стили

Системы на основе хранилища данных (классная доска)