Нижегородский государственный университет им. Н.И. Лобачевского

Разработка архитектуры программного обеспечения. Основные архитектурные стили

При поддержке компании Intel

Кустикова В.Д., кафедра математического обеспечения ЭВМ, факультет ВМК

Содержание

- Понятие архитектуры ПО
- Важность архитектуры ПО
- Основные вопросы, определяющие архитектуру ПО
- Цели архитектуры ПО
- Основные архитектурные принципы
- Архитектурные стили
  - Понятие архитектурного стиля, классификация
  - Архитектура «файл-сервер»
  - Клиент-серверная архитектура
  - Компонентная архитектура
  - Послойная архитектура
  - Архитектура «шина сообщений»
  - Многозвенная архитектура
  - Объектно-ориентированная архитектура
  - Сервис-ориентированная архитектура

Цели

- Объяснить понятие архитектурного стиля, пояснить основные вопросы, определяющие архитектуру ПО, изучить наиболее известные архитектурные стили, рассмотреть ситуации их использования, преимущества и недостатки.

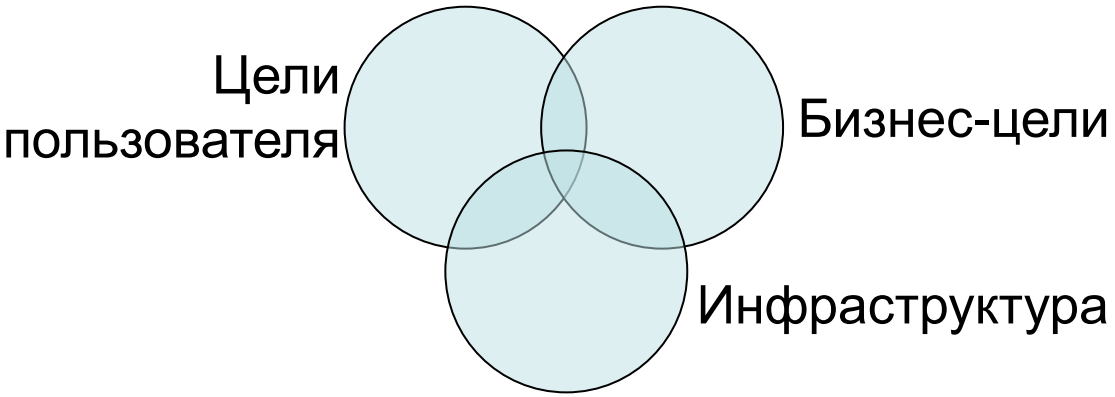
АРХИТЕКТУРА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Архитектура ПО

- ***Архитектура программного обеспечения*** – базовая организация программной системы, которая включает сторонние системы, программы, программные компоненты и отношения между ними и окружением, а также удовлетворяет всем техническим и функциональным требованиям при условии, что оптимизированы общие атрибуты качества (производительность, безопасность, управляемость).

Важность архитектуры ПО

- Архитектура ПО должна быть разработана с учетом целей пользователя, системы (инфраструктуры) и бизнес-целей.



- Для каждой области необходимо определить
  - ключевые сценарии,
  - первостепенные атрибуты качества,
  - ключевые зоны удовлетворения требованиям.
- По возможности разработать метрики, отражающие степень соответствия требованиям.

Основные вопросы, определяющие архитектуру ПО

- Как пользователи будут использовать приложение?
- Как приложение будет развертываться и управляться?
- Каковы требования к атрибутам качества (безопасность, производительность, параллелизм, конфигурация)?
- Как следует спроектировать приложение, чтобы оно было гибким и легко сопровождаемым с течением времени?
- Какие архитектурные направления могут повлиять на приложение сейчас и после его развертывания?

Цели архитектуры ПО

- Отражать структуру системы и скрывать детали ее реализации.
- Делать наглядным исполнение всех сценариев использования системы.
- Обращаться к требованиям различных заинтересованных сторон.
- Удовлетворять функциональным требованиям и требованиям к качеству.

Архитектурные принципы

- ***Проектирование архитектуры и ее модификация вместо проектирования в конце разработки.*** Необходимо прогнозировать изменения требований к системе (модификация или расширение функционала), чтобы обеспечить гибкость архитектуры.
- ***Моделирование для анализа и оценки рисков.*** Необходимо применять средства проектирования (например, Unified Modeling Language), чтобы определять требования, архитектурные и проектные решения.
- ***Использование визуальных моделей*** как инструментов эффективного взаимодействия и совместной работы заинтересованных сторон.
- ***Определение основных инженерных решений.***

АРХИТЕКТУРНЫЙ СТИЛЬ

Архитектурный стиль

- ***Архитектурный стиль*** – множество принципов, обеспечивающих абстрактную структуру для семейства программных систем. Архитектурный стиль повышает качество декомпозиции и способствует повторному использованию проекта, обеспечивая решение для часто возникающих задач.

Классификация архитектурных стилей…

- ***Архитектура «файл-сервер» (File-Server Architecture).*** Извлекает данные из файлов базы данных и передает клиенту для обработки.
- ***Клиент-серверная архитектура (Client-Server Architecture).*** Разделяет систему на два приложения. Клиент отправляет запросы к серверу. Во многих случаях сервер – это база данных, а логика приложения представлена в виде хранимых процедур на сервере.
- ***Компонентная архитектура (Component-Based Architecture).*** Разделяет систему на повторно используемые функциональные или логические блоки с определенными интерфейсами взаимодействия.

Классификация архитектурных стилей…

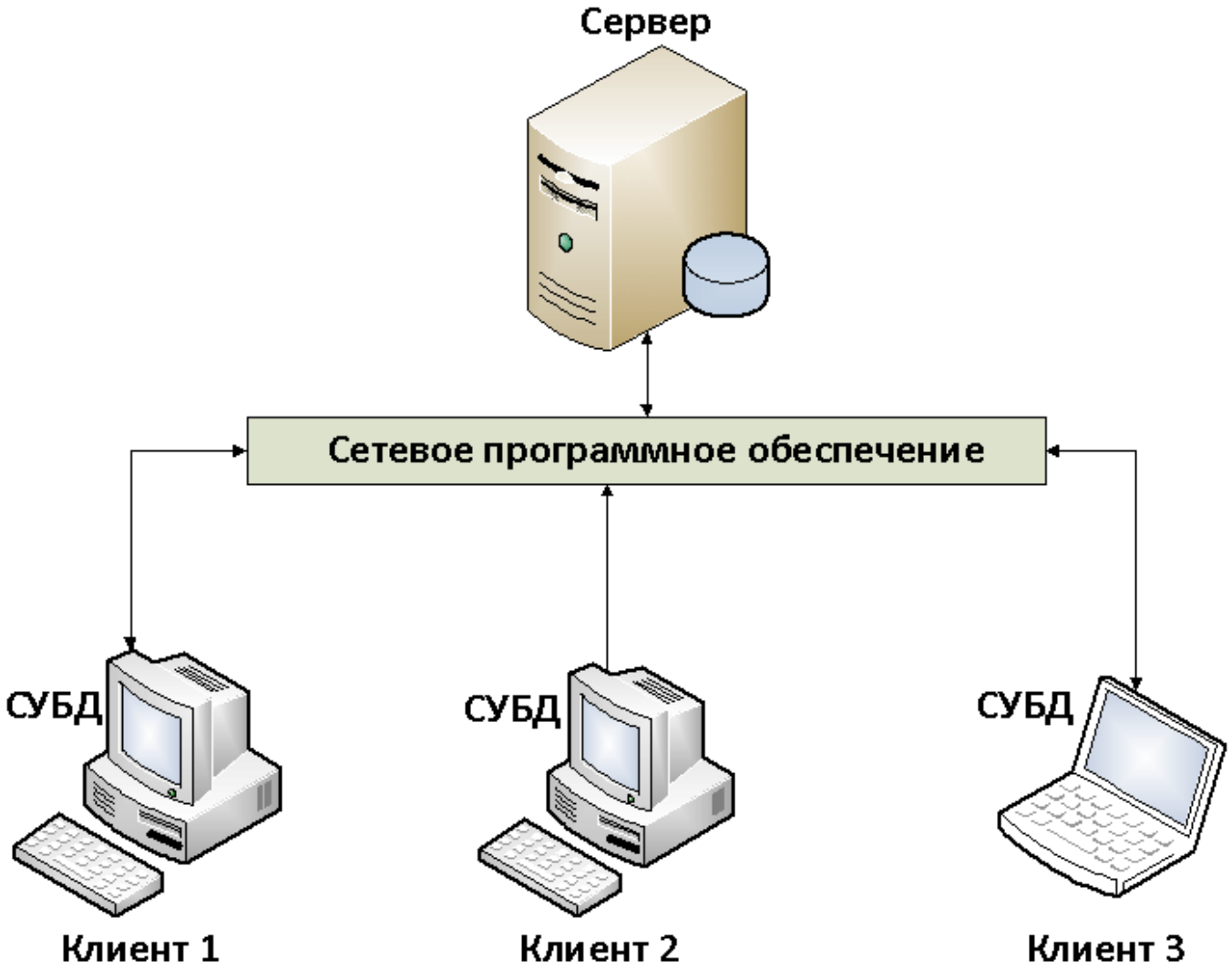
- ***Послойная архитектура (Layered Architecture).*** Система представляется в виде стека слоев (приложений).
- ***Архитектура «шина сообщений» (Message Bus).*** Система представляется в виде множества приложений, взаимодействующих посредством получения/отправки сообщений через один или несколько каналов.
- ***Многозвенная архитектура (N-tier/3-tier).*** Разделяет функциональность на отдельные сегменты (звенья) подобно послойной архитектуре, но каждое звено физически расположено на разных компьютерах.

Классификация архитектурных стилей

- ***Объектно-ориентированная архитектура (Object-Oriented Architecture).*** Парадигма, основанная на разделении ответственностей в системе между отдельными повторно используемыми самостоятельными объектами, каждый из которых содержит набор данных и функций, определяющих поведение объекта.
- ***Сервис-ориентированная архитектура (Service-Oriented Architecture, SOA).*** Приложение представляется в виде службы, предоставление информации потребителю выполняется посредством контрактов и сообщений.

АРХИТЕКТУРА «ФАЙЛ-СЕРВЕР»

Архитектура «файл-сервер»…



Архитектура «файл-сервер»…

- ***Описание:***
  - Функции сервера ограничиваются хранением данных.
  - Обработка данных полностью ложится на клиента.
  - Сервер по запросу клиента извлекает данные из файла (файлов) базы данных и передает их для обработки клиенту.
- ***Ситуации и примеры использования:***
  - Архитектура «файл-сервер» применима исключительно при работе с небольшими объемами данных и небольшим количество клиентов, осуществляющих операции записи.

Архитектура «файл-сервер»

- ***Преимущества:***
  - Простота освоения и установки.
  - Удобство использования.
  - Доступность.
- ***Недостатки:***
  - Избыточность передаваемых данных.
  - Падение производительности с ростом объема данных и числа пользователей, обусловленное увеличением объема сетевого трафика и времени обработки данных на клиенте.
  - Значительные задержки при выполнении операции записи данных.

КЛИЕНТ-СЕРВЕРНАЯ АРХИТЕКТУРА

Клиент-серверная архитектура…

Клиент-серверная архитектура…

- ***Описание:***
  - Клиент-серверная архитектура описывает взаимодействие между клиентом и одним или несколькими серверами.
  - Клиент отправляет запросы серверу, ожидает отклика и обрабатывает ответ.
  - Сервер обычно авторизует пользователя, затем выполняет обработку, необходимую для генерации результата.

Клиент-серверная архитектура…

- ***Ситуации и примеры использования:***
  - Настольные приложения с графическим интерфейсом, которые взаимодействуют с сервером баз данных, содержащим большую часть бизнес-логики в виде хранимых процедур (клиенты электронной почты, FTP-клиенты).
  - Веб-приложения.
  - Инструменты и утилиты, которые управляют удаленными системами (системы управления, инструменты мониторинга сети).

Клиент-серверная архитектура…

- ***Преимущества:***
  - Высокая степень безопасности, т.к. все данные и важные функции их обработки хранятся на сервере.
  - Централизованный доступ к данным.
  - Простота поддержки. Роли и ответственности разделены между несколькими серверами, которые объединены в сеть и имеют информацию друг о друге. Это гарантирует, что клиент не знает, когда происходит ремонт, обновление или перемещение сервера.
  - Уменьшение сетевого трафика, т.к. обработка выполняется на сервере.
  - Увеличение производительности за счет кэширования данных на сервере.

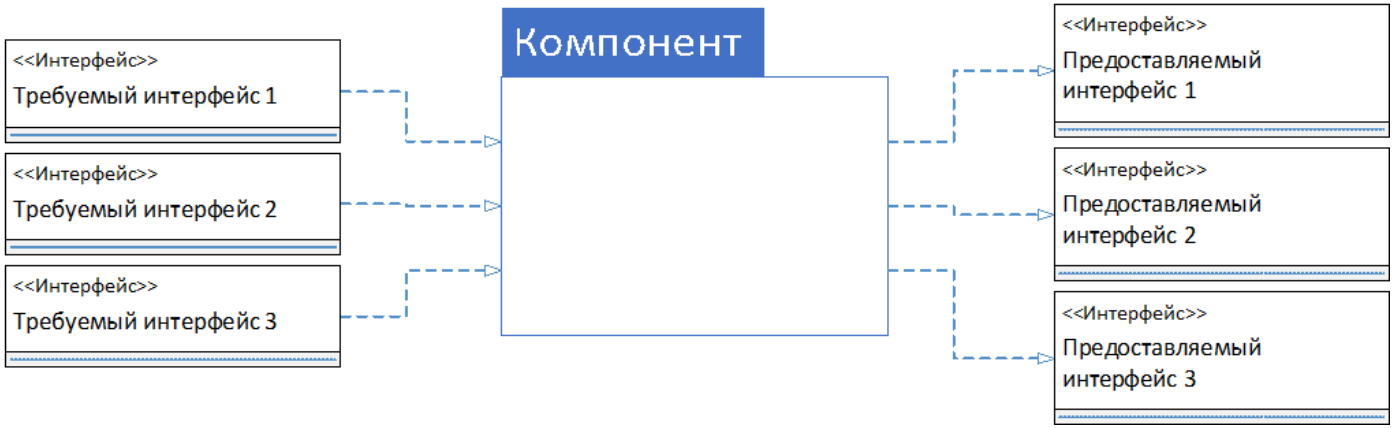
Клиент-серверная архитектура

- ***Недостатки:***
  - Для классической двухзвенной клиент-серверной архитектуры характерна тесная связь данных и бизнес-логики, что может негативно сказываться на расширяемости и масштабируемости.
  - Зависимость от центрального сервера также оказывает негативное влияние на надежность системы.

КОМПОНЕНТНАЯ АРХИТЕКТУРА

Компонентная архитектура…

- ***Описание:***
  - Декомпозиция системы на отдельные функциональные и логические компоненты с четко выделенными интерфейсами взаимодействия, содержащими методы, события и свойства.
  - Каждая компонента имеет два связанных интерфейса:
    - Сервисы, предоставляемые компонентой.
    - Сервисы, необходимые для корректного функционирования



Компонентная архитектура…

- ***Ключевые свойства компонент:***
  - Повторное использование в различных сценариях и приложениях.
  - Заменимость. Компоненты должны легко заменяться аналогичными.
  - Независимость от контекста. Компоненты проектируются с целью функционирования в различных средах и контекстах. Специфическая информация должна передаваться в компоненту, а не включаться.
  - Расширяемость для обеспечения нового поведения.
  - Инкапсуляция. Компоненты предоставляют внешние интерфейсы для взаимодействия и скрывают детали реализации.
  - Максимальная независимость от других компонент.

Компонентная архитектура…

- ***Ситуации и примеры использования:***
  - Компоненты пользовательского интерфейса (таблицы, кнопки и другие элементы управления).
  - Компоненты в рамках одной платформы, обеспечивающей единую среду исполнения (the component object model (COM), the distributed component object model (DCOM), Common Object Request Broker Architecture (CORBA), Enterprise JavaBeans (EJB)).

Компонентная архитектура…

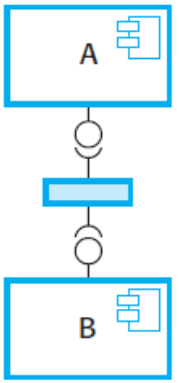
- ***Модель компонент*** – определение стандартов для реализации, документирования и развертывания.
- Основные элементы модели компонент:
  - Интерфейсы. Модель определяет способ описания интерфейса и его элементов (именование свойства, операций, исключений), фиксирует язык описания.
  - Информация для использования:
    - Метаданные – данные о самом компоненте (интерфейсы, атрибуты), которые позволяют пользователю определить предоставляемые и требуемые сервисы.
    - Информация для конфигурирования бинарных компонент под определенную среду.
  - Развертывание. Модель определяет комплектацию пакета для развертывания компонента.

Компонентная архитектура…

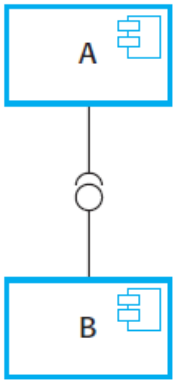
- ***Реализация модели компонент включает две категории сервисов:***
  - Платформенные сервисы – сервисы, позволяющие компонентам взаимодействовать в распределенной среде. Примеры:
    - Определение интерфейса.
    - Управление исключениями.
    - Взаимодействие компонент.
  - Вспомогательные сервисы – общие сервисы, которые требуются многим компонентам системы. Примеры:
    - Безопасность.
    - Управление ресурсами.
    - Параллелизм.

Компонентная архитектура…

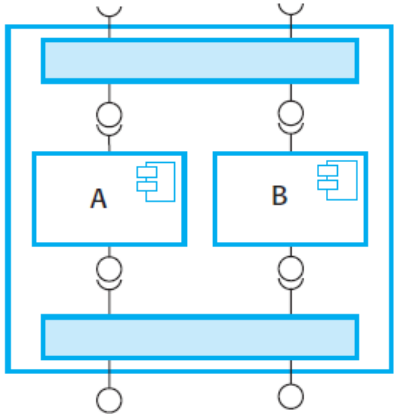
- ***Способы объединения компонент:***
  - ***Последовательное объединение.*** Новый компонент создается из двух существующих, сервисы реализуются посредством вызова сервисов первой и второй компонент.



  - ***Иерархическое объединение.*** Одна компонента вызывает сервисы другой.



  - ***Аддитивная композиция.*** Новый компонент создается из нескольких существующих.



**Соммервиль И. Инженерия программного обеспечения, 6-е издание.: Пер. с англ. – М.: Издательский дом "Вильямс", 2002. – 624 с.*

Компонентная архитектура…

- ***Преимущества:***
  - Простота развертывания. С выходом новой версии достаточно заменить соответствующую компоненту.
  - Уменьшение стоимости разработки и сопровождения за счет использования компонент сторонних разработчиков.
  - Простота разработки вследствие независимости разрабатываемых компонент.
  - Повторное использование компонент в нескольких приложениях или системах.

Компонентная архитектура

- ***Недостатки:***
  - Предопределенная обработка сообщений.
  - Доверие сторонним разработчикам встраиваемых компонент.
  - Сложность организации взаимодействия между компонентами.
  - Сложность тестирования компонент сторонних разработчиков, поставляемых без привязке к среде исполнения.
  - Синдром второй системы. Успех создания первой системы приводит к преждевременному обобщению решения. Совершенствуются устаревшие функции системы, новые решения не развиваются.

ПОСЛОЙНАЯ АРХИТЕКТУРА

Послойная архитектура…

Пользовательский интерфейс /
Слой представления данных

Управление пользовательским интерфейсом
Авторизация и аутентификация

Основная логика / Функционал приложения
Системные утилиты

Системная поддержка (ОС, базы данных и т.п.)

Послойная архитектура…

- ***Описание:***
  - Основывается на группировке связанной функциональности (общая роль или ответственность) внутри приложения на отдельных слоях, которые сложены в виде стека.
  - Слои явно, но слабо связаны, и располагаются на одном физическом компьютере или распределены по разным.
  - Компоненты слоя взаимодействуют с компонентами других слоев через четко определенные интерфейсы.

Послойная архитектура…

- ***Принципы проектирования послойной архитектуры:***
  - Абстракция. Послойная архитектура отделяет представление данных от логики, при этом четко выделяются роли и ответственности отдельных слоев, а также отношения между ними.
  - Инкапсуляция. Нет необходимости делать предположения о типах данных, методах, свойствах или реализации во время проектирования, т.к. эти детали не отображаются на границах слоев.

Послойная архитектура…

- Четко выделенные функциональные слои. Верхние слои отправляют команды нижним слоям, могут реагировать на события в этих слоях, обеспечивая движение данных между слоями.
- Высокая связность компонент слоя. Каждый слой содержит функциональность, которая напрямую связана с задачами слоя.
- Повторное использование. Нижние слои не зависят от верхних слоев, поэтому их можно повторно использовать в других системах.
- Низкая связность слоев. Взаимодействие между слоями основано на абстрагировании и применении механизма событий.

Послойная архитектура…

- ***Ситуации и примеры использования:***
  - Применима для приложений, которые должны поддерживать различные типы клиентов и устройств.
  - Используется при реализации сложных настраиваемых бизнес-правил и процессов при наличии слоев, которые предоставляют необходимые сервисы и которые можно повторно использовать для разработки приложений.

Послойная архитектура…

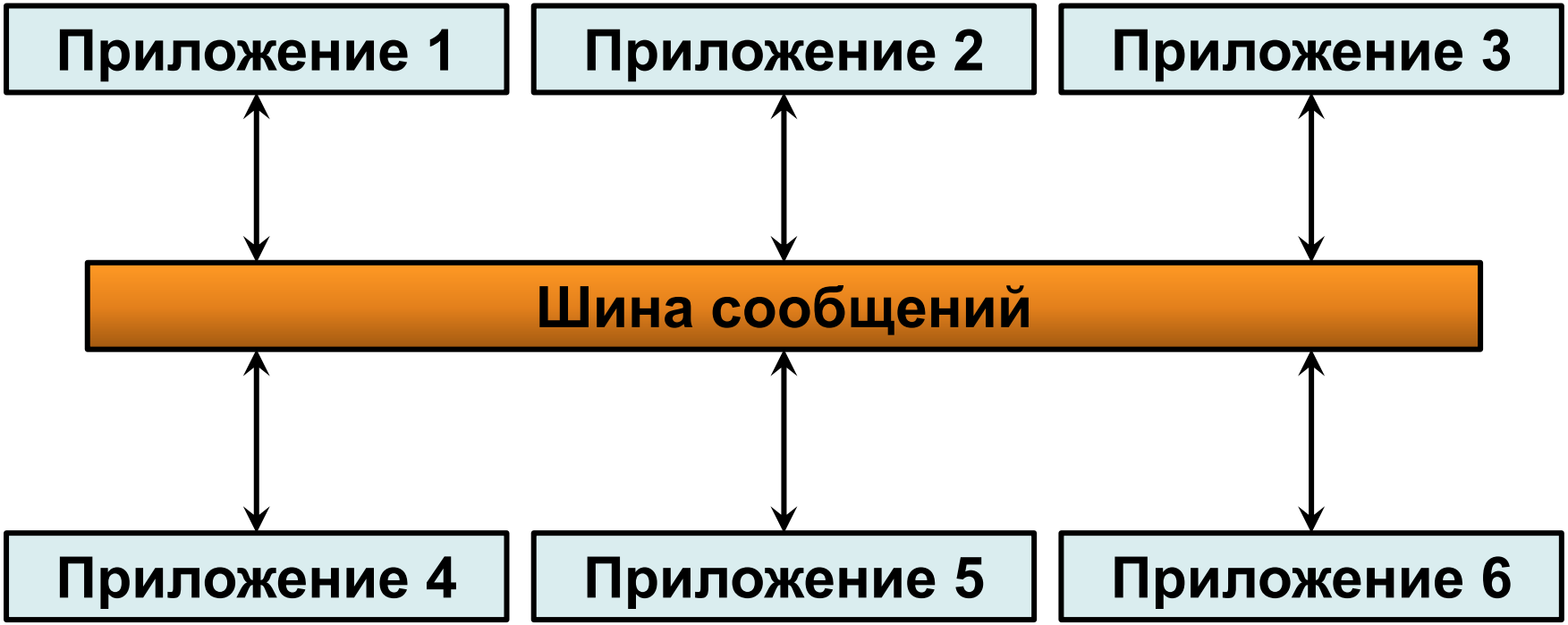
- ***Преимущества:***
  - Абстракция, автономность и управляемость. Возможна замена целых слоев без изменения интерфейсов взаимодействия между соседними слоями.
  - Производительность. Распределение слоев между физическими компьютерами может повысить масштабируемость и отказоустойчивость.
  - Наличие избыточных средств (например, аутентификация) на каждом слое повышает надежность системы в целом.
  - Повторное использование.
  - Простота тестирования через замену реализаций слоев.

Послойная архитектура

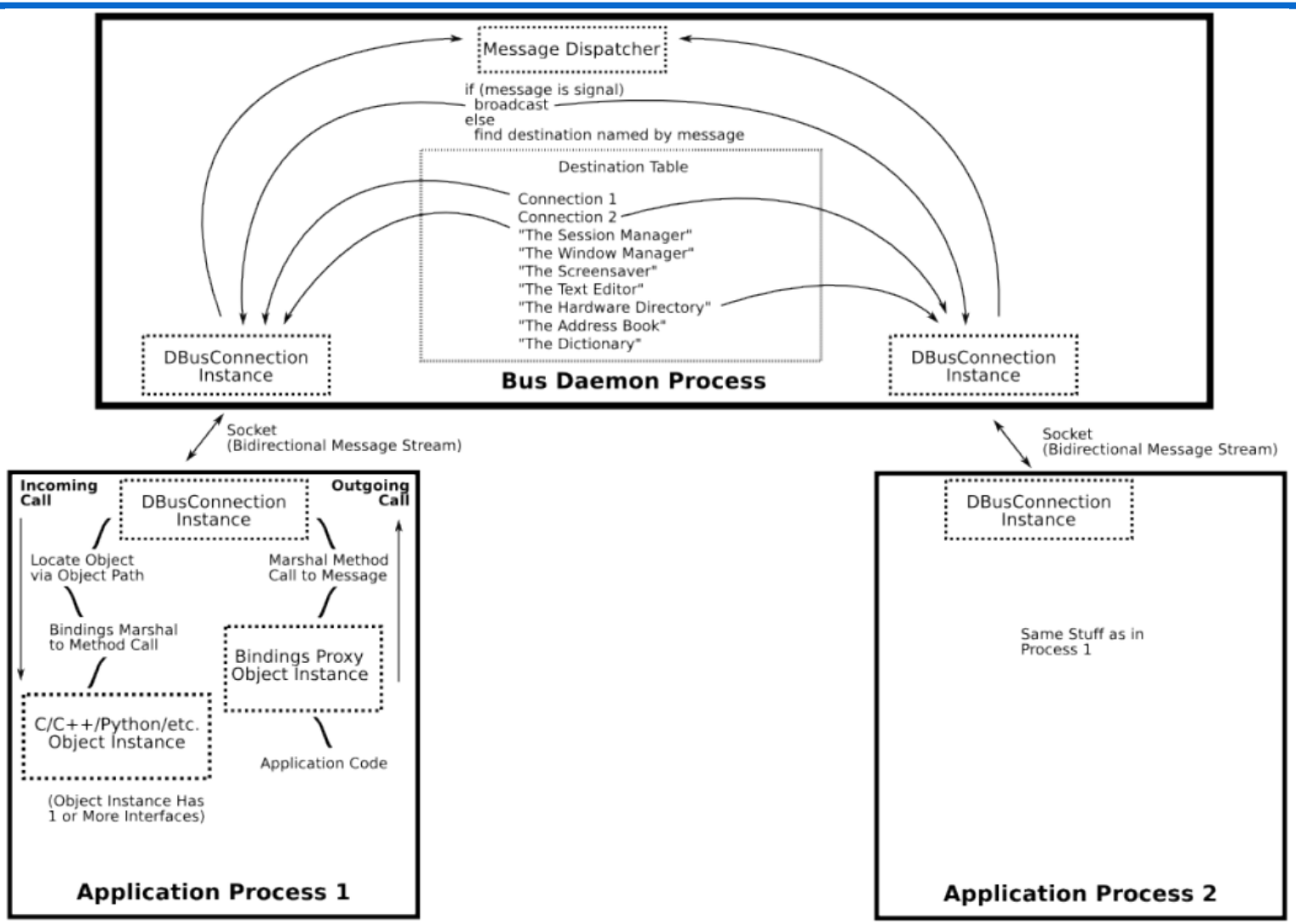
- ***Недостатки:***
  - На практике сложно выделить отдельные слои и зачастую слой верхнего уровня может непосредственно взаимодействовать со слоями нижнего уровня, а не через промежуточные слои.
  - Производительность может быть проблемой из-за множества слоев, т.к. запрос обрабатывается на каждом уровне архитектуры.

АРХИТЕКТУРА «ШИНА СООБЩЕНИЙ»

Архитектура «шина сообщений»…



Архитектура «шина сообщений»…



*http://kremer.cpsc.ucalgary.ca/courses/seng403/W2013/papers/04ArchitectureStyles.pdf

Архитектура «шина сообщений»…

- ***Описание:***
  - Архитектура «шина сообщений» описывает принцип использования программной системы, в основе которой лежит передача и получение сообщений по одному или нескольким каналам взаимодействия, вследствие чего приложения могут общаться, не имея дополнительной информации друг о друге.
  - Взаимодействие между приложениями, как правило, осуществляется посредством асинхронной передачи сообщений.

Архитектура «шина сообщений»…

- Разновидности архитектуры «шины сообщений»:
  - ***Сервисная шина предприятия*** (Enterprise Service Bus). Приложения в сети предприятия. Использует для взаимодействия компонент и шины сервисы, закрепленные за шиной. Шина обычно обеспечивает сервисы, которые преобразуют сообщения из одного формата в другой, позволяя клиентам использовать несовместимые форматы для взаимодействия.
  - ***Сервисная шина сети Интернет*** (Internet Service Bus). Приложения в облаке. Ключевая концепция – использование универсальных идентификаторов ресурсов (URIs) и политик управления маршрутизацией логики через приложения и сервисы в облаке.

Архитектура «шина сообщений»…

- ***Ситуации и примеры использования:***
  - Централизованное управление распределенной инфраструктурой.
  - Системы, в которых для решения задач требуется взаимодействие с внешними приложениями или приложениями, которые физически расположены в разных средах.

Архитектура «шина сообщений»…

- ***Преимущества:***
  - Расширяемость. Приложения добавляются/удаляются в шину, не оказывая влияния на другие приложения.
  - Низкая сложность. Каждому необходимо иметь информацию о том, как взаимодействовать с шиной.
  - Гибкость. Приложения, составляющие часть сложного процесса, можно легко изменять в соответствии с требованиями бизнеса или пользователя путем изменения конфигурации или параметров маршрутизации.
  - Низкая связность. Приложения имеют только доступный интерфейс взаимодействия с шиной.

Архитектура «шина сообщений»…

- Масштабируемость. С шиной могут взаимодействовать одновременно несколько экземпляров приложения, чтобы обрабатывать несколько запросов одновременно.
- Простота приложения. Приложению необходимо поддерживать только одно соединение к шине сообщений вместо нескольких соединений к каждому приложению, с которым оно взаимодействует.

Архитектура «шина сообщений»

- ***Недостатки:***
  - Сложность организации шины сообщений. Шина является разделяемым ресурсом, поэтому должна обеспечивать параллелизм обработки сообщений и решать проблему организации очередей сообщений.
  - Сложность модификации интерфейсов доступа к шине.
  - Низкая степень безопасности в отсутствии шифрования сообщений.

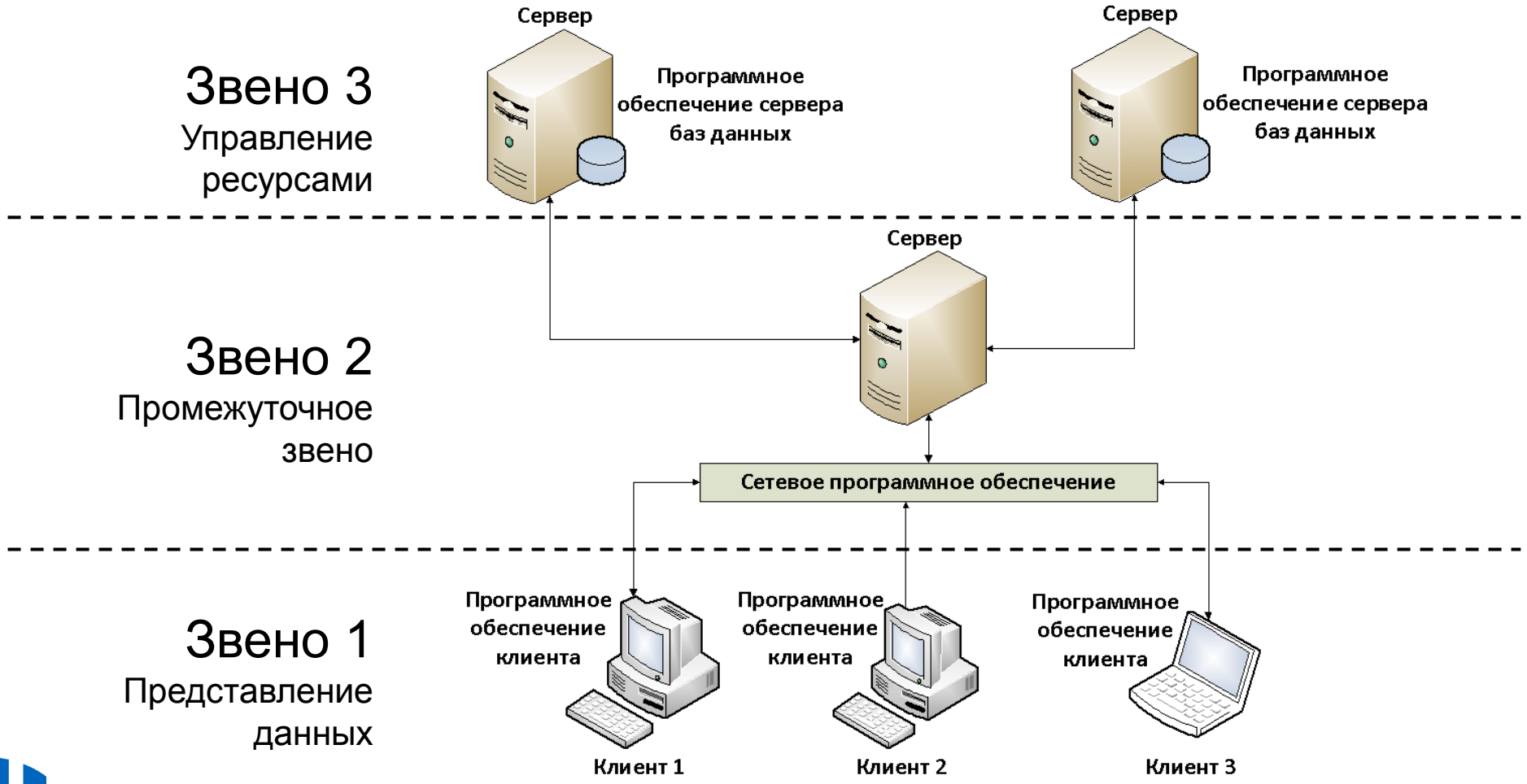
МНОГОЗВЕННАЯ АРХИТЕКТУРА

Многозвенная архитектура…

- ***Описание:***
  - Многозвенная архитектура разделяет функциональность системы на сегменты подобно послойной архитектуре, но каждый сегмент (звено) физически расположен на отдельном компьютере.
  - Для взаимодействия звеньев обычно используются платформенно зависимые методы (обычно асинхронные) вместо передачи сообщений.
  - Каждое звено N независимо от остальных звеньев, за исключением звеньев N+1 и N-1. Звену N необходимо знать, как обрабатывать запросы звена N-1 и пересылать их к звену N+1, и как обрабатывать результаты собственных запросов.

Многозвенная архитектура…

- ***Трехзвенная клиент-серверная архитектура:***



Многозвенная архитектура…

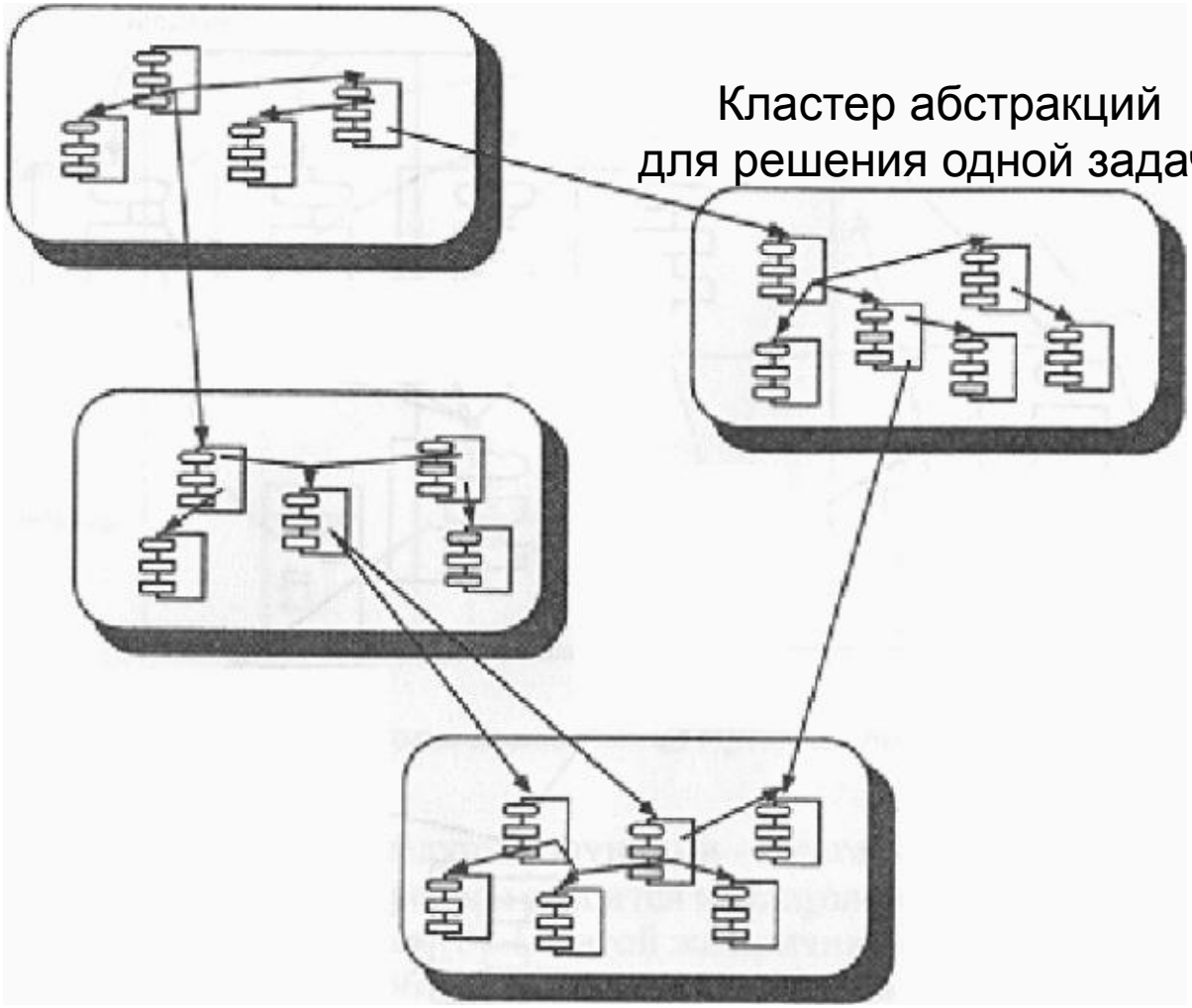
- ***Ситуации и примеры использования:***
  - Используется, когда обработка данных на каком-либо слое требует достаточно много ресурсов, что замедляет обработку в других слоях.
  - Приложения, в которых интегрируются данные из нескольких ресурсов.
  - Используется при условии, что требования к безопасности на каждом слое различны.
  - Типичные финансовые веб-приложения, в которых бизнес-слой защищен программными и аппаратными средствами межсетевой защиты, а слой представления – отдельное звено в рамках той же сети.
  - Приложения с сотнями или тысячами клиентов.

Многозвенная архитектура…

- ***Преимущества:***
  - Сопровождаемость. Звенья независимы, поэтому изменения и обновления одного звена не влияют на работу других.
  - Масштабируемость вследствие организации звеньев в виде слоев.
  - Гибкость. Обусловлена тем, что звенья могут управляться и масштабироваться независимо.

ОБЪЕКТНО-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА

Объектно-ориентированная архитектура…



**Буч Г. Объектно-ориентированный анализ и проектирование с примерами на C++. 2000.*

Объектно-ориентированная архитектура…

- ***Описание:***
  - Основана на разделении ответственностей внутри приложения между отдельными повторно используемыми и самодостаточными объектами, каждый из которых содержит данные и методы, определяющие поведение объекта.
  - Система рассматривается как последовательность взаимодействующих объектов.
  - Объекты независимые и слабосвязанные, взаимодействуют через интерфейсы посредством вызова методов, обращения к свойствам других объектов или отправки/получения сообщений.

Объектно-ориентированная архитектура…

- ***Принципы проектирования объектно-ориентированной архитектуры:***
  - Абстракция. Абстракция выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов.
  - Композиция. Объект можно представить в виде набора других объектов, к которым можно ограничить доступ.
  - Инкапсуляция. Объекты предоставляют функционал через методы, свойства или события и скрывают внутренние детали (состояние, переменные, объекты).
  - Полиморфизм. Обеспечивает возможность переопределения поведения объекта.

Объектно-ориентированная архитектура…

- Наследование. Объекты могут наследоваться от других объектов, использовать функциональность базовых объектов или переопределять ее с целью получения нового поведения.
- Расщепление. Функционал объектов разделяется посредством определения абстрактных интерфейсов, которые реализуются объектом. Введение абстрактных интерфейсов обеспечивает возможность разработки альтернативных реализаций. Изменение реализации не затрагивает интерфейс, как следствие, не влияет на объекты, использующие этот интерфейс.

Объектно-ориентированная архитектура…

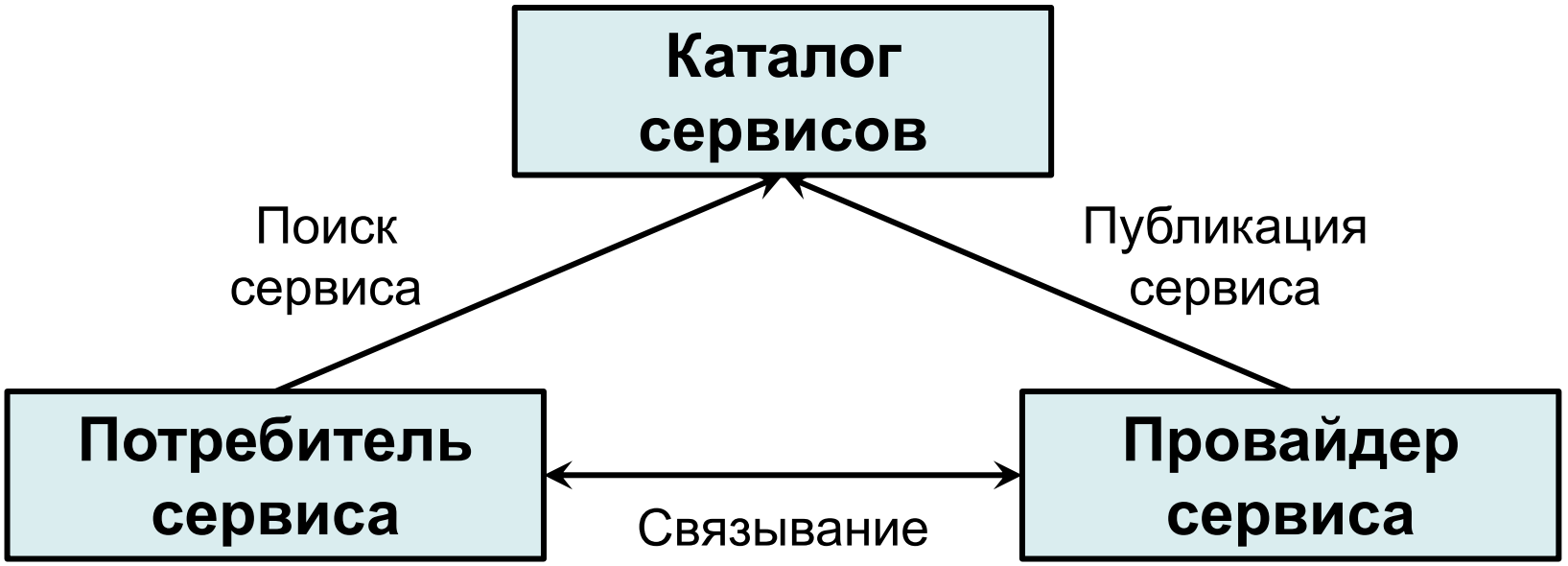
- ***Ситуации и примеры использования:***
  - Используется для моделирования приложения, основанного на объектах реального мира и их взаимодействии.
  - Объектно-ориентированный стиль применим, если необходимо объединить логику и данные в виде повторно используемых компонент.

Объектно-ориентированная архитектура

- ***Преимущества:***
  - Понятность. Объекты системы легко сопоставимы с объектами реального мира.
  - Повторное использование объектов.
  - Простота расширения путем добавления новых объектов и отношений.
  - Простота тестирования.
- ***Недостатки:***
  - Сильная связь между базовыми классами и их наследниками. Перемещение базовых классов ведет к разрушению иерархии.
  - Для небольших приложений применение не оправдано.

СЕРВИС-ОРИЕНТИРОВАННАЯ АРХИТЕКТУРА

Сервис-ориентированная архитектура…



- ***Каталог*** содержит информацию по всем службам и сервисам, которая публикуется провайдером.
- ***Потребитель*** отправляет запрос к ***Каталогу***, который обеспечивает связывание его с ***Провайдером***.

Сервис-ориентированная архитектура…

- ***Описание:***
  - Система представляется в виде множества сервисов и приложений, использующих функционал этих сервисов.
  - Сервисы используют стандартизированные интерфейсы взаимодействия. Взаимодействие сервисов и приложений основано на передаче сообщений определенного формата (например, протокол SOAP, стандарт WSDL).
  - Архитектура в основном базируется на технологии веб-сервисов.

Сервис-ориентированная архитектура…

- ***Принципы сервис-ориентированной архитектуры:***
  - Автономность и независимость сервисов в отношении разработки, развертывания, поддержки и обновления.
  - Распределенность сервисов. Сервисы физически могут располагаться на любом доступном компьютере, в сети локально или удаленно при условии, что поддерживаются необходимые протоколы передачи данных.
  - Сервисы предоставляют схему и контракт взаимодействия, а не внутренние классы.
  - Совместимость обеспечивается посредством определения средств передачи, протокола и уровня безопасности.

Сервис-ориентированная архитектура…

- ***Ситуации и примеры использования:***
  - Сервис-ориентированные приложения обеспечивают распределение информации, обработку многошаговых процессов (системы предварительных заказов, онлайн-магазины).
  - Создание бизнес-приложений, которые представляют результаты работы многих сервисов, содержащих различную бизнес-логику.

Сервис-ориентированная архитектура…

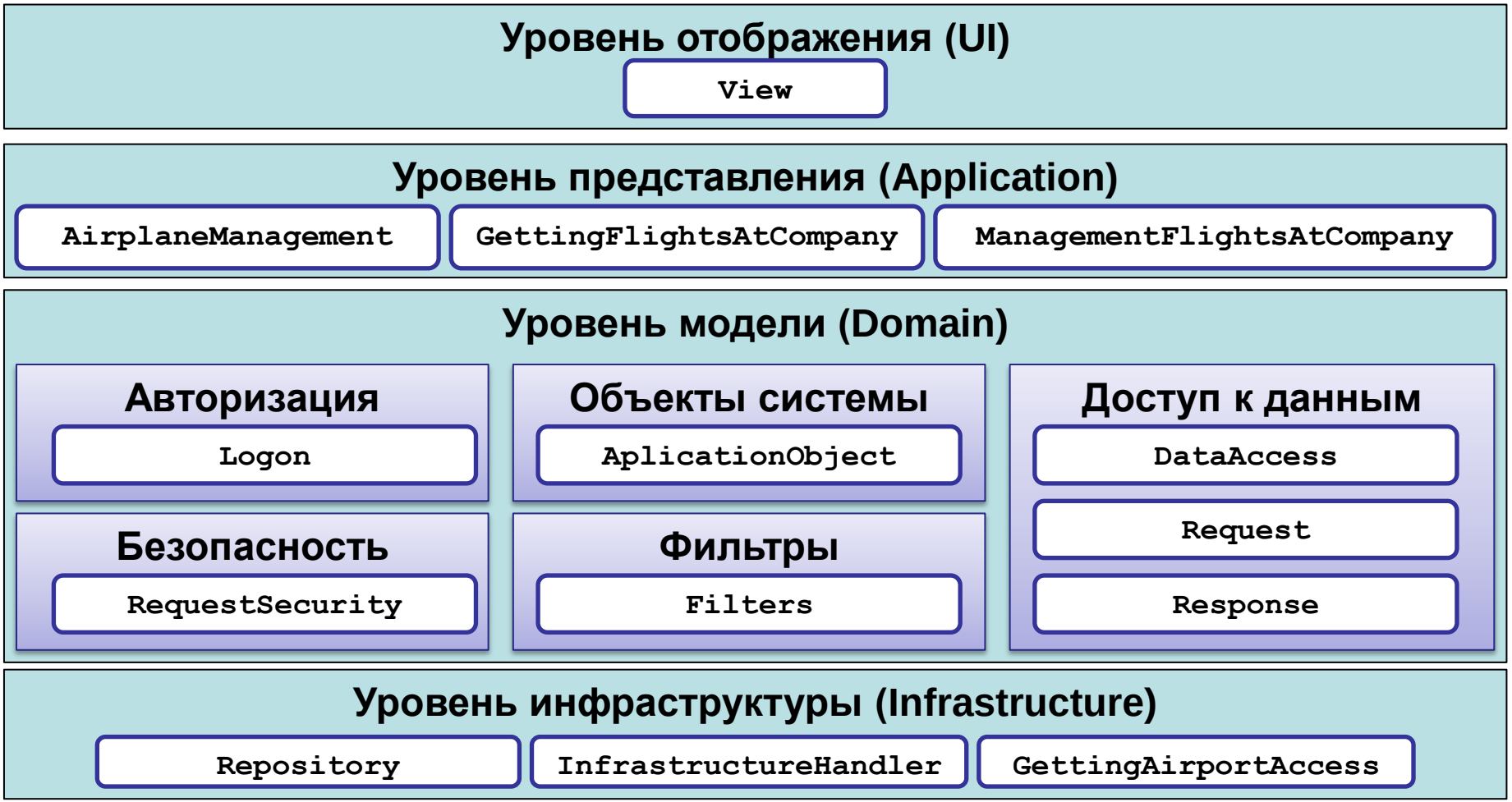
- ***Преимущества:***
  - Отсутствие жестких связей между программными компонентами (сервисами).
  - Повторное использование общих сервисов со стандартными интерфейсами расширяет возможности приложений и снижает стоимость разработки.
  - Совместимость. Вследствие того, что протоколы передачи и форматы данных являются промышленными стандартами, сервис провайдера и приложение потребителя могут функционировать на разных платформах.
  - Рациональная организация функциональности сервисов, обеспечивающая отсутствие дублирований.

Сервис-ориентированная архитектура

- ***Недостатки:***
  - Скорость обработки сообщений.
  - Необходимость разграничения доступа и построения распределенной архитектуры защиты сервисов системы.
  - Проблемы интеграции с приложениями, спроектированными не под сервис-ориентированную архитектуру.

ПРИМЕР РАЗРАБОТКИ АРХИТЕКТУРА

Разработка архитектуры на примере…



Разработка архитектуры на примере

База данных системы (пользователи, рейсы, самолеты)

Заключение

- Введено понятие архитектурного стиля.
- Рассмотрены следующие архитектурные стили: архитектура «файл-сервер», клиент-серверная архитектура, компонентная архитектура, послойная архитектура, архитектура «шина сообщений», многозвенная архитектура, объектно-ориентированная и сервис-ориентированная архитектуры.
- Приведены ситуации и примеры использования указанных архитектурных стилей.
- Отмечены преимущества и недостатки рассмотренных архитектурных стилей.

Контрольные вопросы…

- Дайте определение понятию архитектурного стиля.
- Приведите классификацию архитектурных стилей.
- Приведите основные особенности клиент-серверной архитектуры и ее принципиальные отличия от архитектуры «файл-сервер».
- В чем смысл компонентной архитектуры? Какими особенностями должны обладать компоненты?
- Что такое модель компонента? Приведите основные элементы модели компонента.
- Приведите типичные составляющие звенья послойной архитектуры. Как организовано взаимодействие между слоями.

Контрольные вопросы

- В чем преимущества и недостатки архитектуры «шина сообщений»? В каких ситуациях уместно применение данного архитектурного стиля?
- Какое принципиальное отличие двухзвенной и трехзвенной клиент-серверной архитектур? Чем оно обусловлено?
- Приведите принципы, лежащие в основе объектно-ориентированного архитектурного стиля. Поясните смысл каждого.
- Из каких блоков состоит сервис-ориентированная архитектура? Как они взаимодействуют друг с другом?
- При разработке каких приложений следует применять сервис-ориентированный архитектурный стиль?

Литература к лекции

1. **Sommerville I.** Software engineering (9th edition). – Pearson, 2011. – 790 p.
2. **Буч Г.** Объектно-ориентированный анализ и проектирование с примерами на C++. – Изд-во: Бином, Невский диалект. – 1998. – 560с.
3. **Martin R.C.** Design Principles and Design Patterns [http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf].
4. **Martin R.C.** SRP: The Single Responsibility Principle [http://objectmentor.com/resources/articles/srp.pdf].