Лекция 5. Классификация архитектур ПС

Основные понятия и классы архитектур ПС

1. Архитектурные модели и архитектурные стили
2. Паттерны
3. Фреймворки

Архитектурные модели. Классификация

Структурные модели

- Модель репозитория
- Модель клиент-сервер
- Модель абстрактной машины

Модели управления

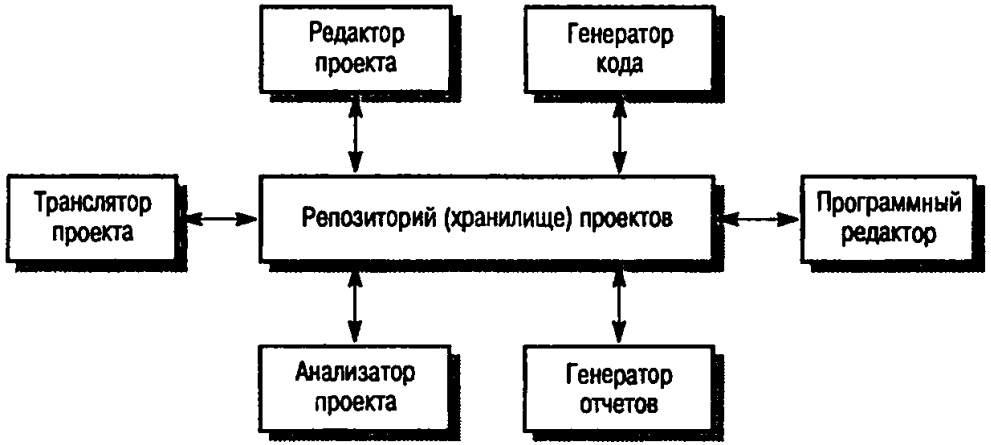
- Модель вызова-возврата
- Модель диспетчера
- Модель передачи сообщений
- Модель управления прерываниями

Проблемно-зависимые архитектуры

- Модели классов систем
- Базовые модели

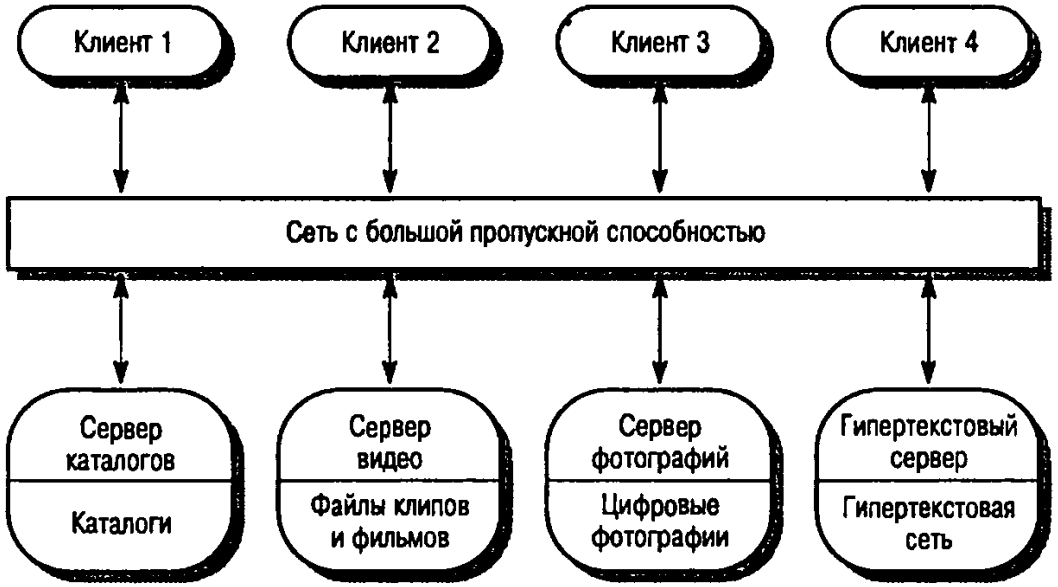
Структурные модели. Модель репозитория

- Не требуется передача данных от одной подсистемы к другой
- Подсистемы должны быть согласованы с моделью репозитория данных
- Подсистемам, создающим данные, не требуется знать как эти данные используются в других
- Модернизация систем сложна и проблематична
- Средства резервного копирования, обеспечения безопасности, восстановления данных, управления доступом централизированы и подчинены единой политики репозитория
- Модель совместного использования репозитория прозрачна
- Сложность с размещением репозитория на нескольких машинах



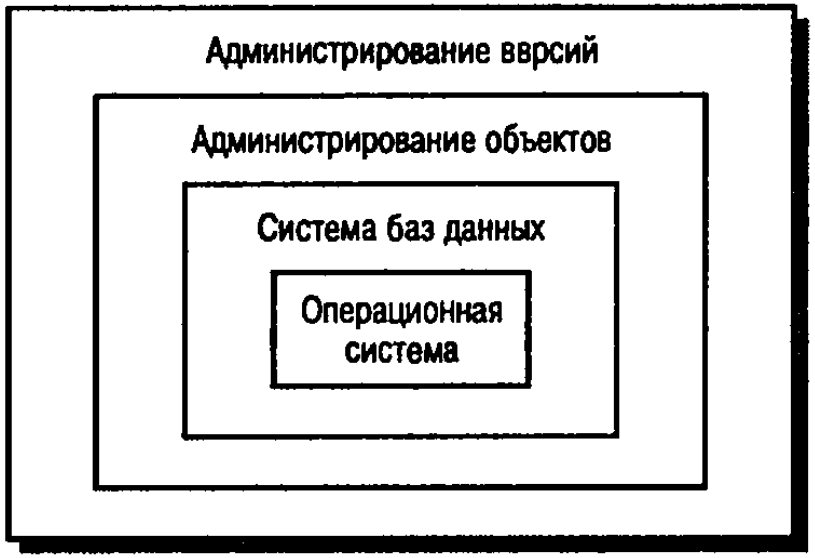
Структурные модели. Модель клиент-сервер

- Набор автономных серверов, предоставляющих сервисы другим подсистемам
  - Файловый сервер, сервер печати, сервер-компилятор …
- Набор клиентов, которые вызывают эти сервисы
- Сеть, посредством которой клиенты получают доступ к сервисам
- Является распределенной архитектурой



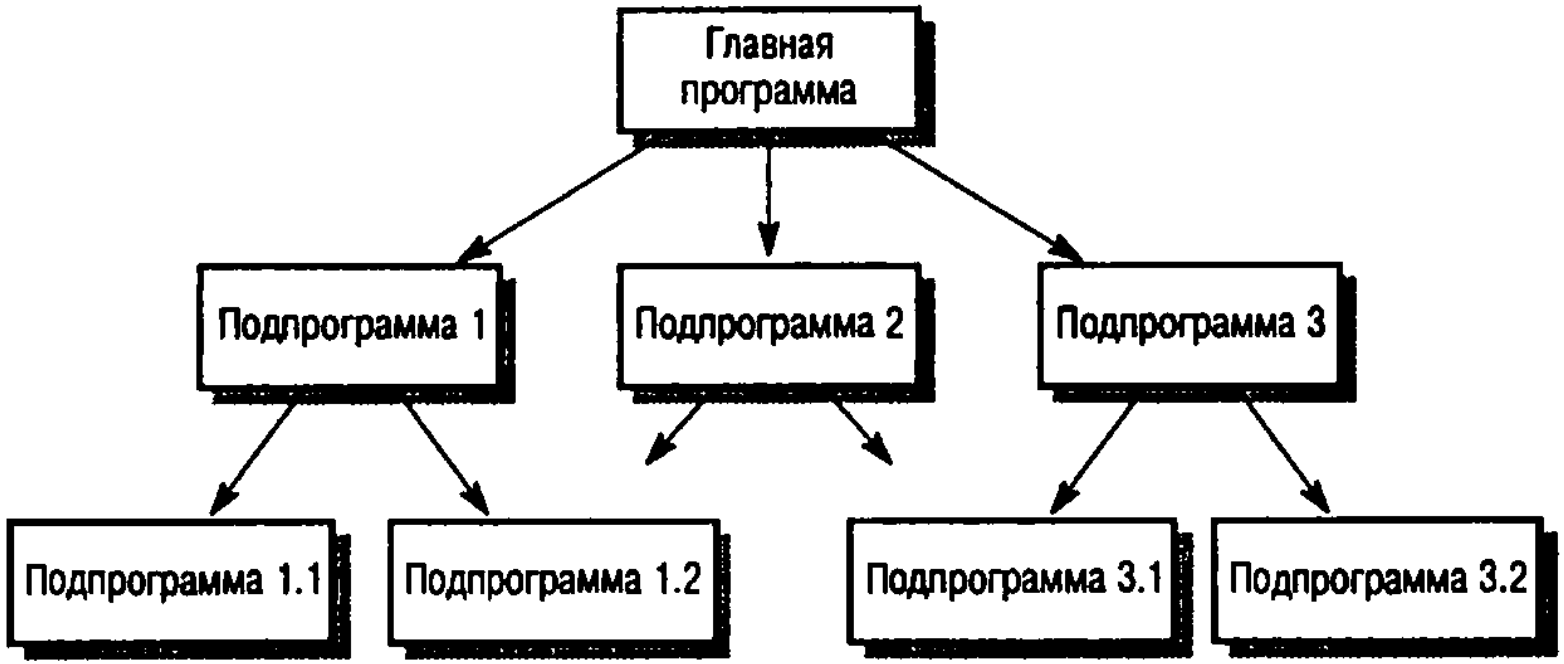
Структурные модели. Модель абстрактной машины

- Многоуровневая модель
  - OSI – Open System Interconnection
- Обеспечивает пошаговое развитие системы
- Архитектура легко изменяема и переносима на другие платформы
- Сложная структура системы



Модели управления.
Централизованное управление.
Модель вызова-возврата

- Относительная простота анализа потока управления
- Относительная простота выбора системы, отвечающей за конкретный ввод данных
- Сложность обработки исключительных ситуаций

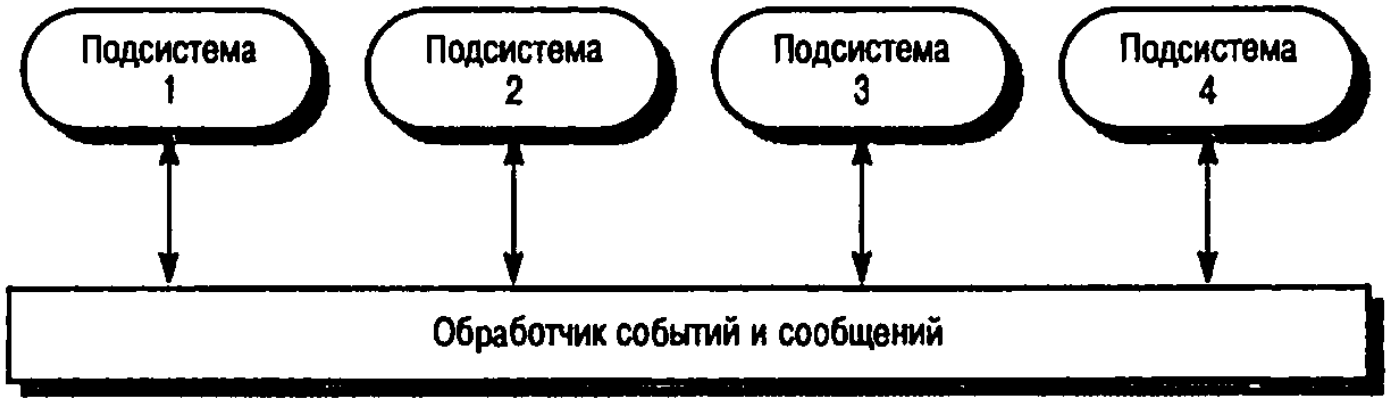


Модели управления. Централизованное управление. Модель диспетчера

- В «мягких» системах реального времени, в которых нет чересчур строгих временных ограничений
- Модель с обратной связью

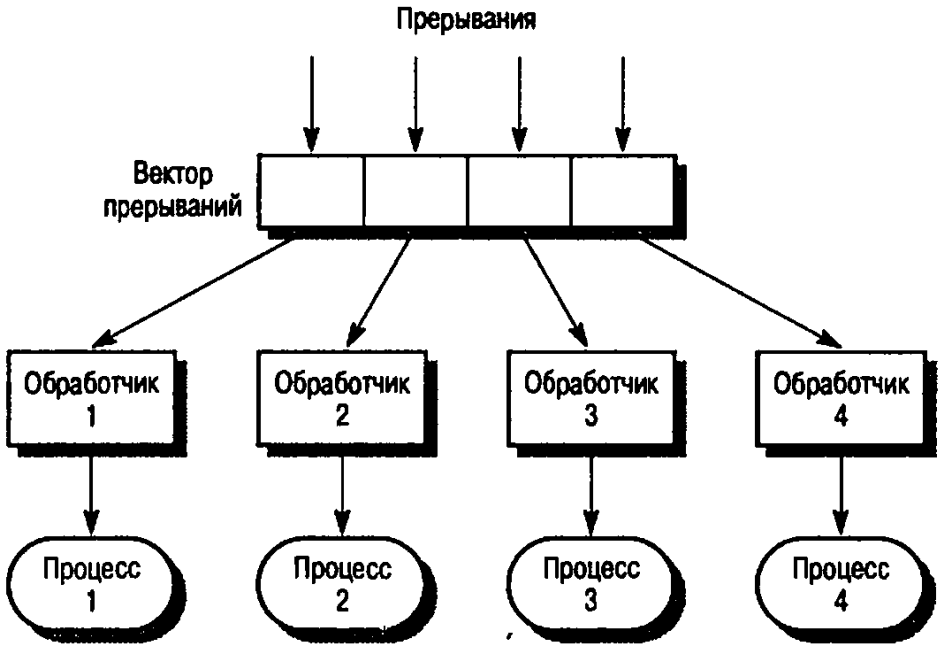
Модели управления.
Системы, управляемые событиями.
Модель передачи сообщений

- Простота модернизации систем
- Новая подсистема интегрируется путем регистрации ее событий в обработчике событий
- Каждая подсистема может активировать другую, не зная ее имени и размещения
- Подсистемы можно реализовать на разных машинах
- недостатком является то, что подсистемы не знают когда произойдет обработка события
- Неизвестно какая подсистема среагирует на событие
- Разные подсистемы могу среагировать на одно и тоже событие



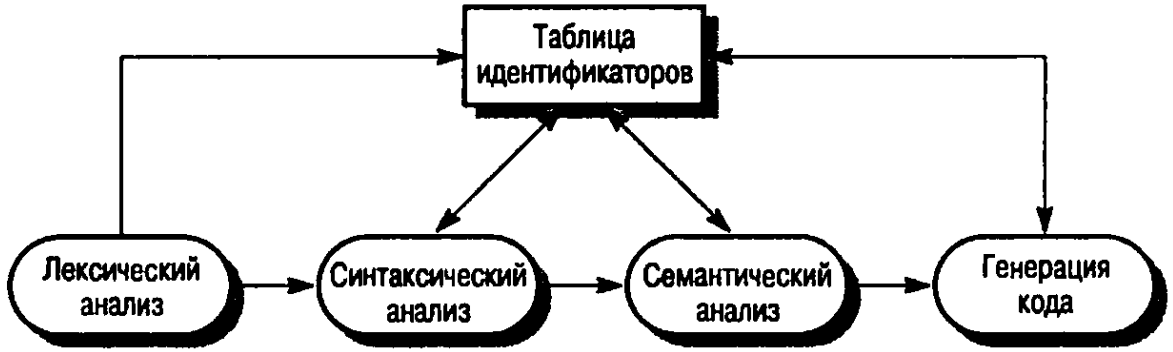
Модели управления. Системы, управляемые событиями. Модель, управляемая прерываниями

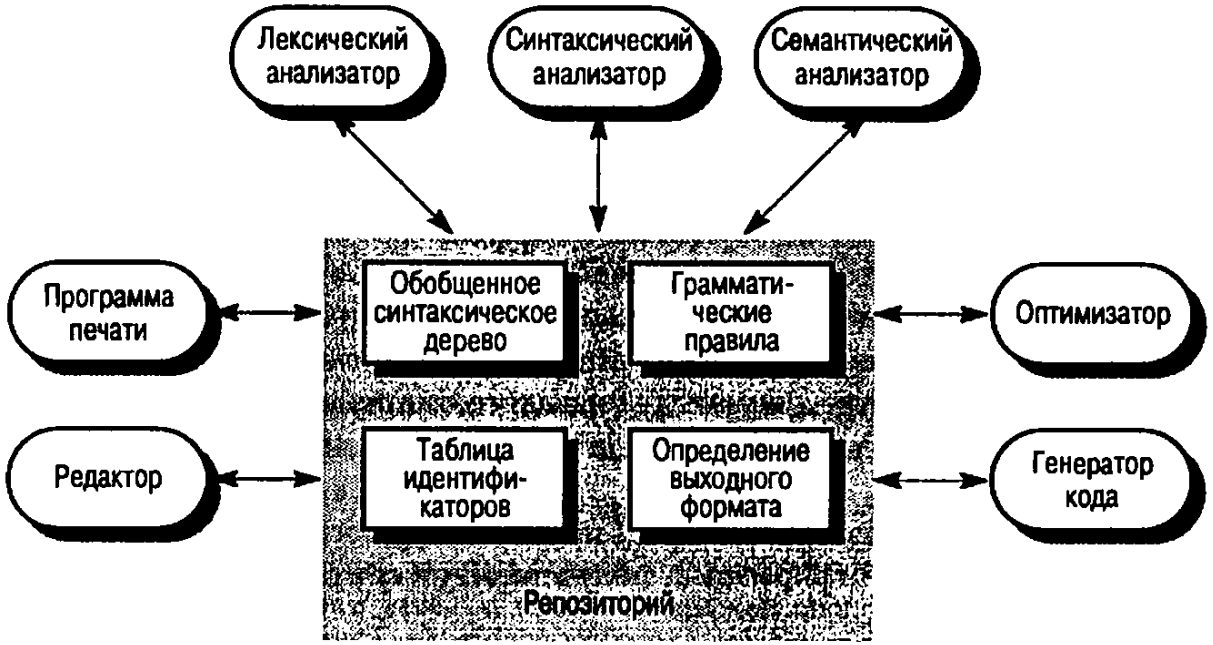
- Используется в жестких системах реального времени, где требуется немедленная реакция на определенные события
- Сложность программирования и аттестации системы
- Сложность изменения системы



Проблемно-зависимые архитектуры. Модели классов систем

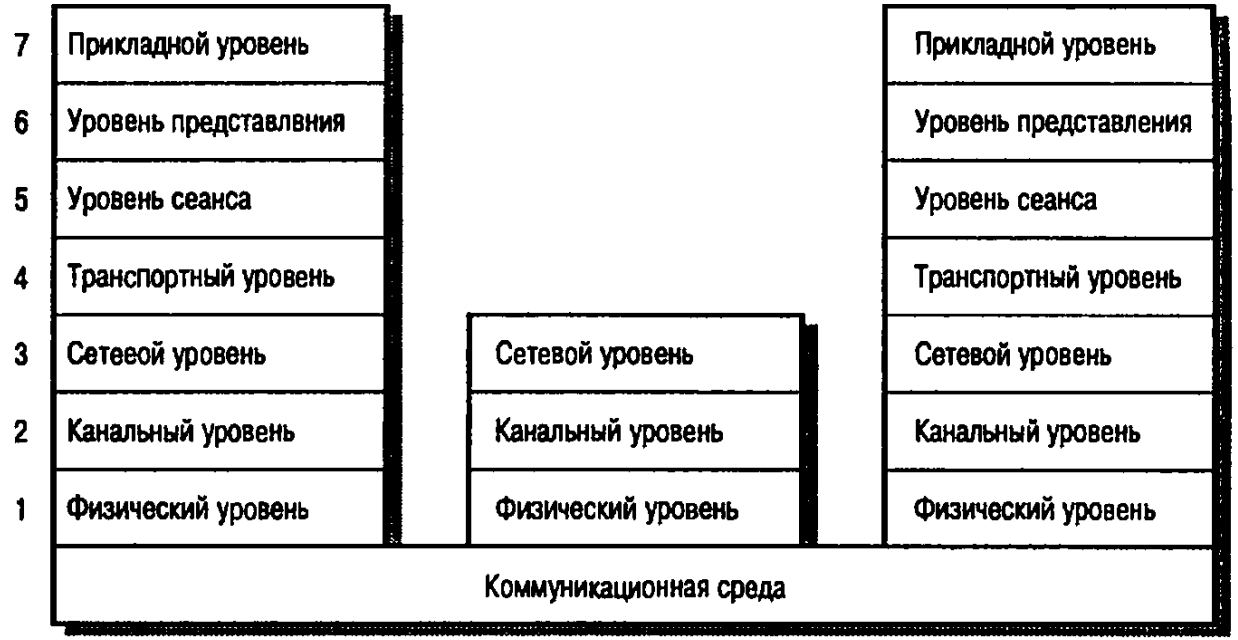
- Отображают классы реальных систем, вобрав в себя основные характеристики этих классов.
- Системы реального времени
  - Системы сбора данных, системы мониторинга, модель компилятора



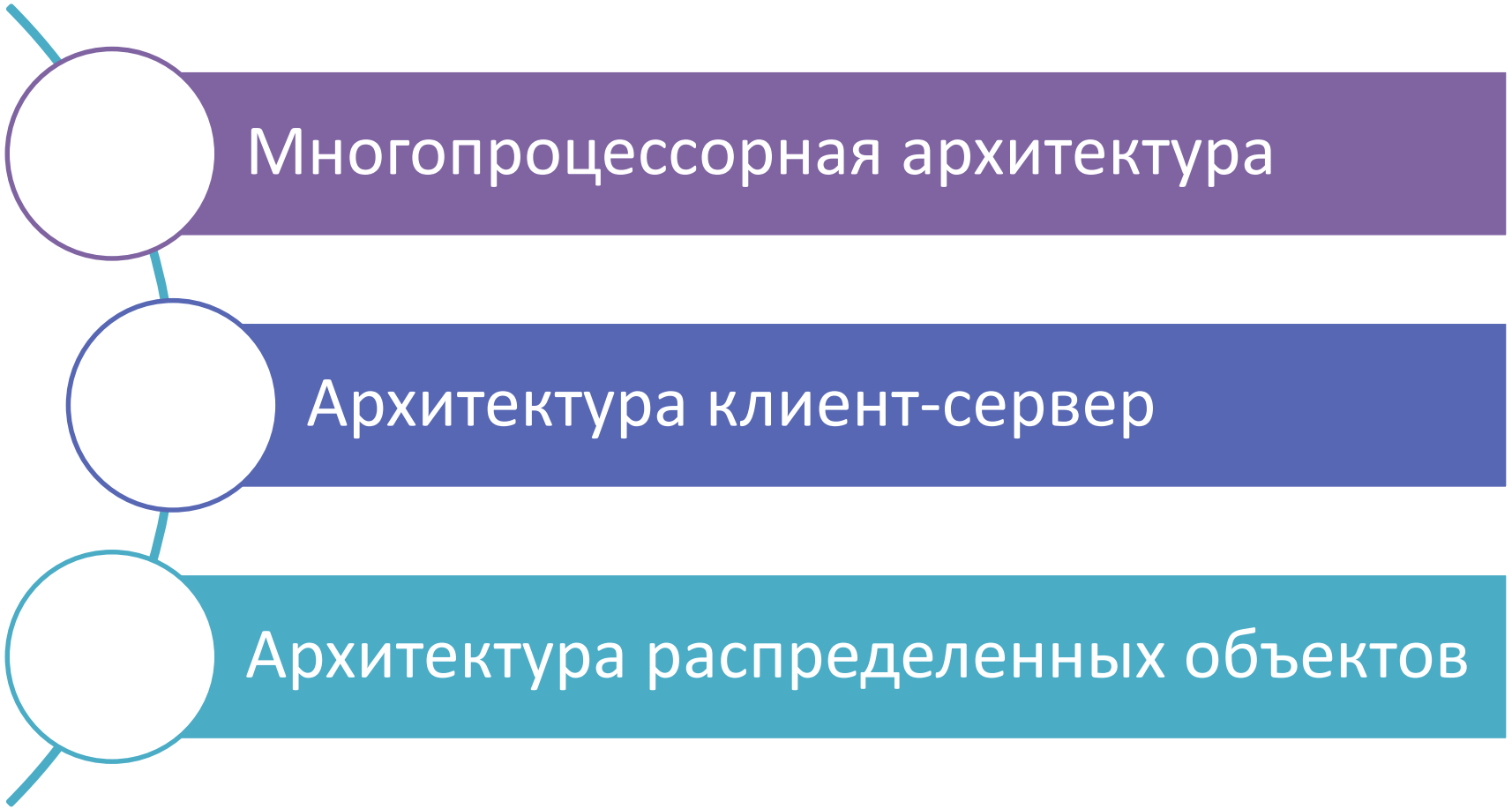


Проблемно-зависимые архитектуры. Базовые модели

- Идеализированная архитектура, отражающая особенности систем, работающих в определенной предметной области
- Служат эталоном для сравнения различных системы в определенной предметной области
  - Стандарт при оценки различных систем



Распределенные модели



Распределенные модели

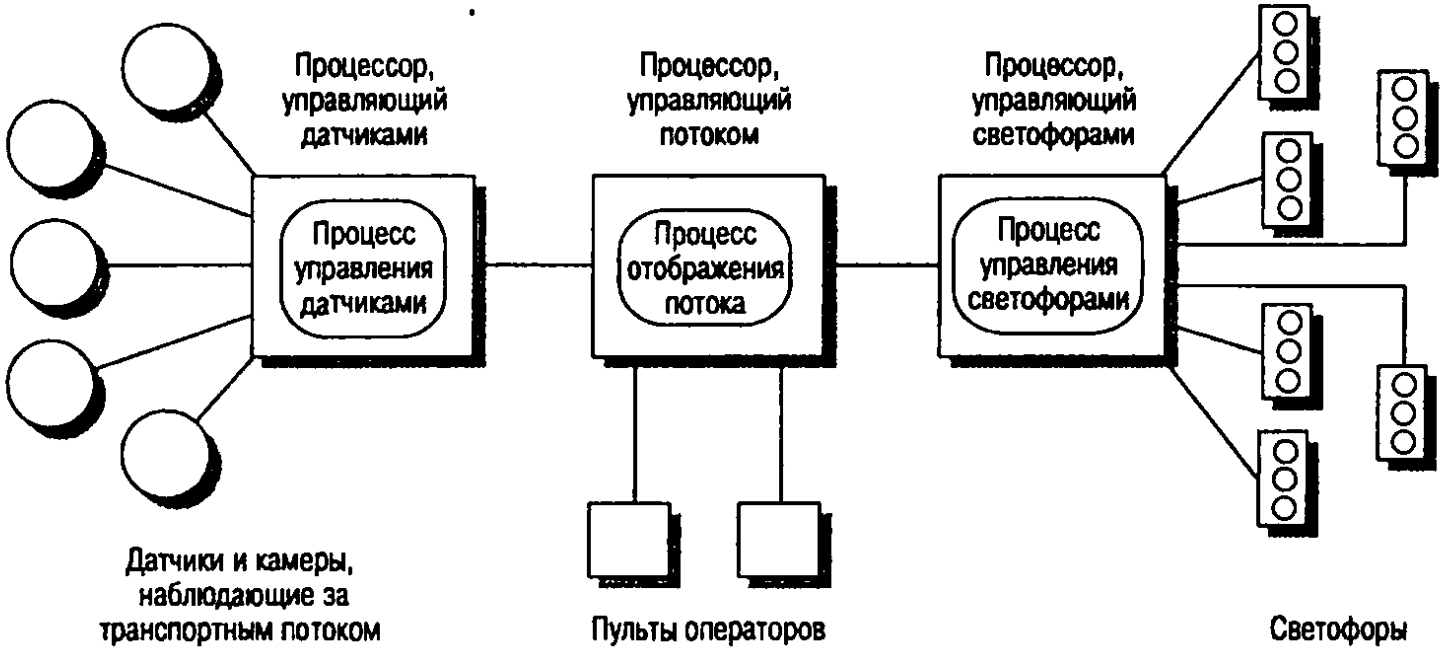
Характеристики

- Совместное использование ресурсов
- Открытость
- Параллельность
- Масштабируемость
- Отказоустойчивость
- Прозрачность

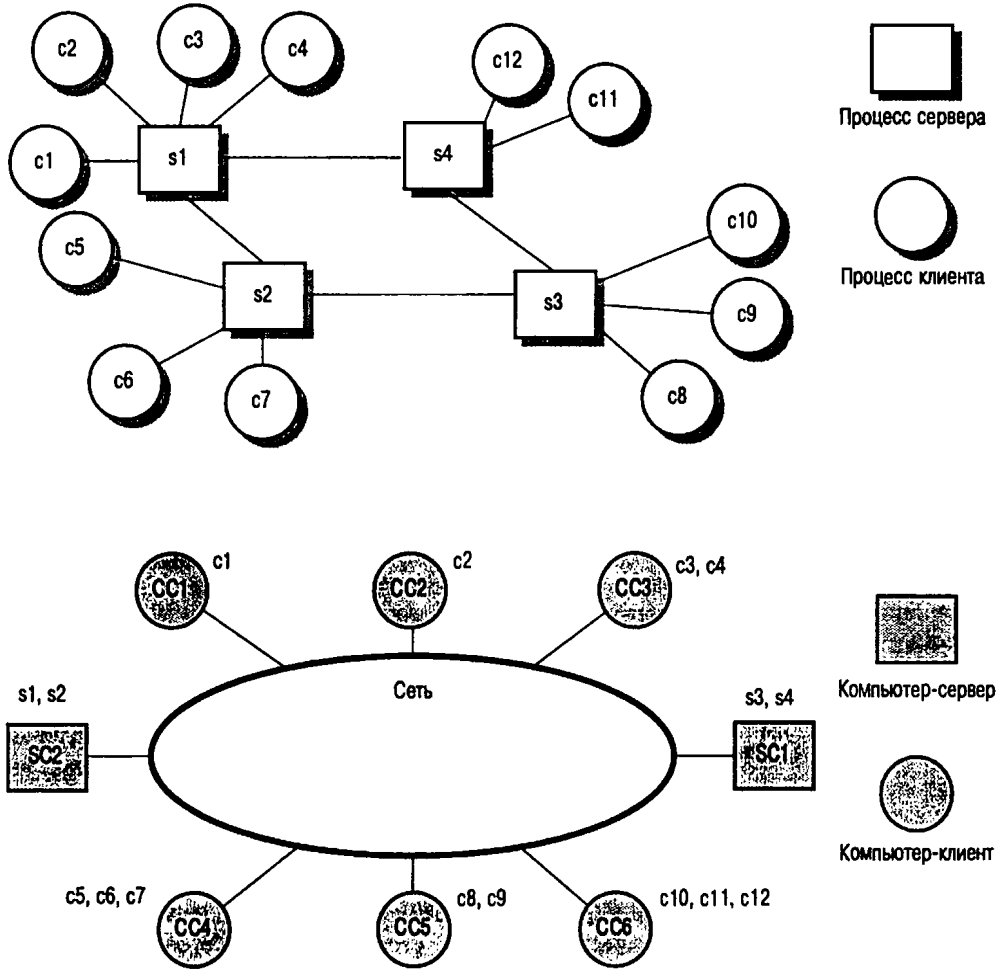
Недостатки

- Сложность
- Безопасность
- Управляемость
- Непредсказуемость

Распределенные модели. Многопроцессорная архитектура



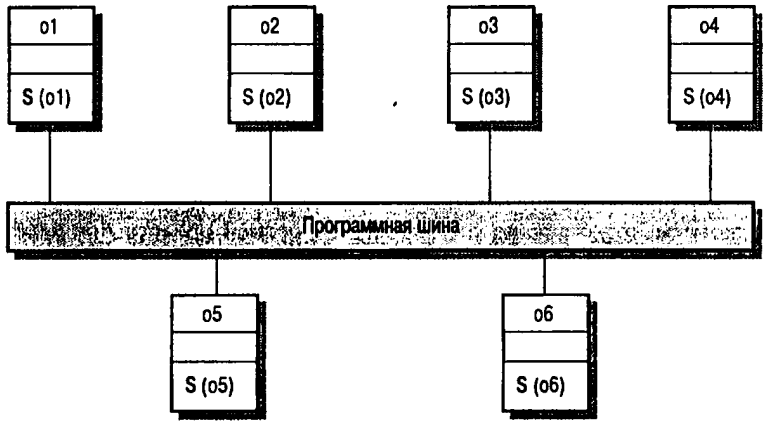
Распределенные модели. Архитектура клиент-сервер

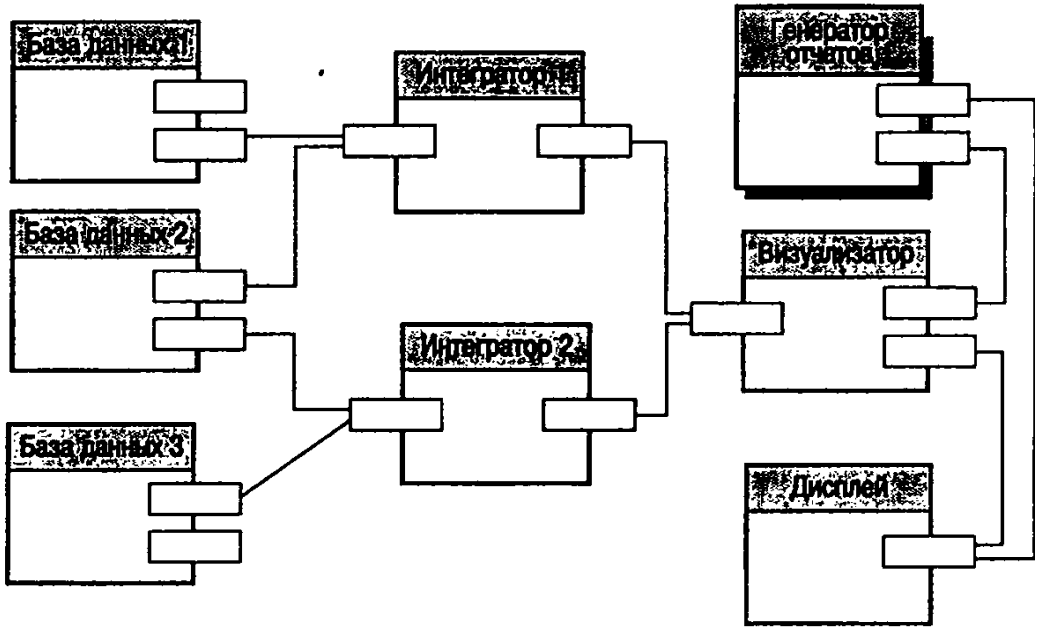


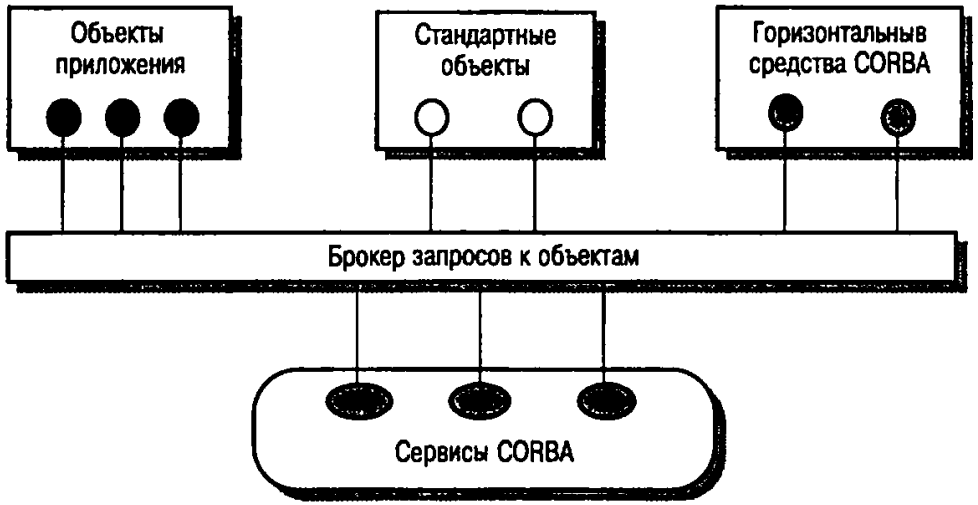
Применение распределенных моделей

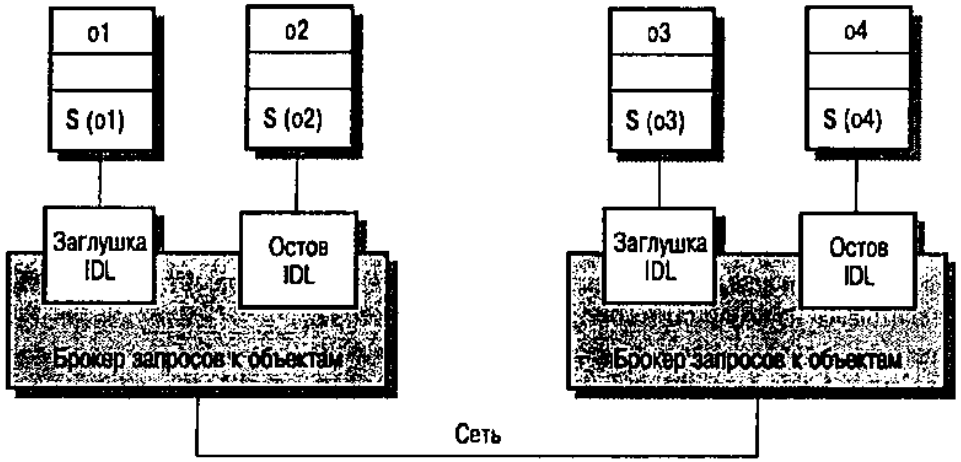
Архитектура	Приложения
Двухуровневая архитектура тонкого клиента	Приложения с интенсивными вычислениями, но незначительным объемом управления данными. Приложения в которых обрабатываются большие массивы данных, но с небольшим объемом вычислений в самом приложении.
Двухуровневая архитектура толстого клиента	Приложения, где требуется интенсивная обработка (визуализация данных / большие объемы вычислений) Приложения с относительно постоянным набором функций на стороне пользователя, применяемых в среде с хорошо отлаженным системным управлением
Трехуровневая и многоуровневая архитектура клиент/сервер	Большие приложения с сотнями и тысячами клиентов Приложения в которых часто меняются и данные, и методы обработки Приложения в которых выполняется интеграция данных из многих источников

Архитектура распределенных объектов









Архитектурный стиль

- Семейство систем в терминах шаблона организации структуры
- Номенклатура компонентов и типов соединительных звеньев, набор условий, в соответствии с которыми они могут соединяться
- Архитектурный стиль, иногда называемый архитектурным шаблоном
  - это набор принципов,
  - высокоуровневая схема, обеспечивающая абстрактную инфраструктуру для семейства систем

Классификация архитектурных стилей

Категория	Архитектурные стили
Связь	SOA, шина сообщений
Развертывание	Клиент/сервер, N-уровневая, 3-уровневая
Предметная область	Domain Driven Development
Структура	Компонентная, объектно-ориентированная, многоуровневая архитектура

Потоки данных

- Data Flow Systems

Вызов с возвратом

- Call-and-Return Systems

Независимые компоненты

- Independent Component Systems

Централизованные данные

- Data-Centric Systems

Виртуальные машины

- Virtual Machines

Архитектурный стиль. Потоки данных

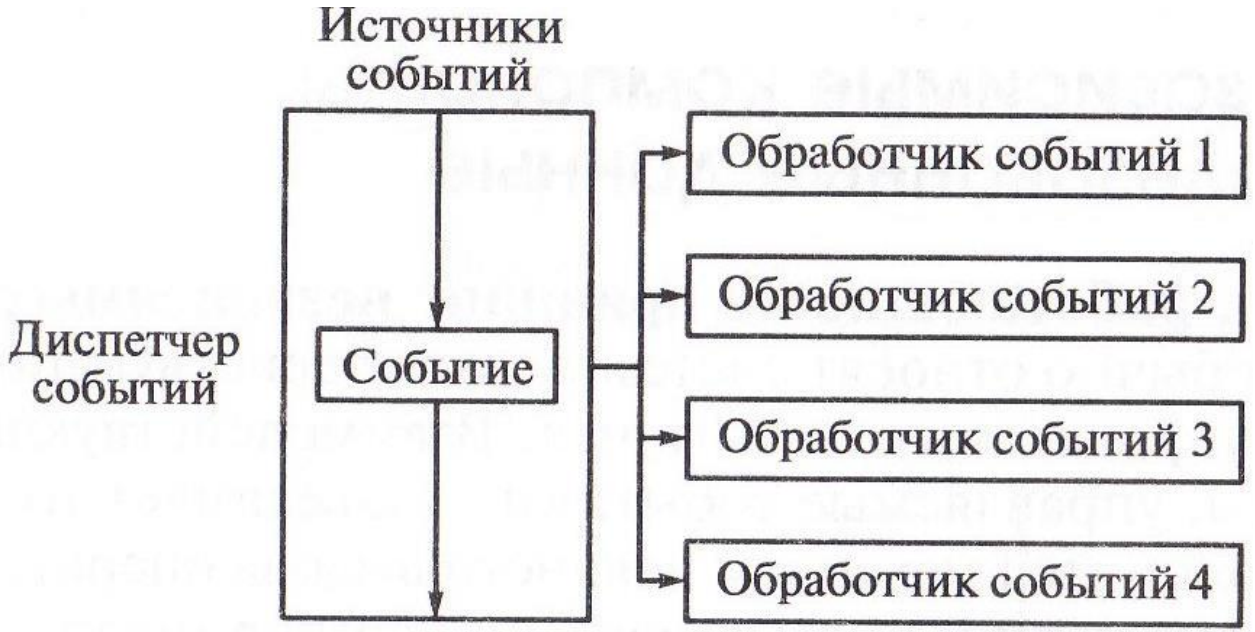
- Системы пакетно-последовательной обработки
  - Batch Sequential Systems
- Системы типа конвейеры и фильтры
  - Pipe and Filter Architecture
    - Компиляторы, системы обработки сигналов и изображений и т.п.

Архитектурный стиль. Вызов с возвратом

- Программа-сопрограмма
  - Main Program and Subroutines
- Клиент-серверные системы
  - Client-Server Systems
- Объектно-ориентированные системы
  - Object-Oriented Systems
- Иерархические многоуровневые системы
  - Hierarchically Layered Systems

Архитектурный стиль. Независимые компоненты

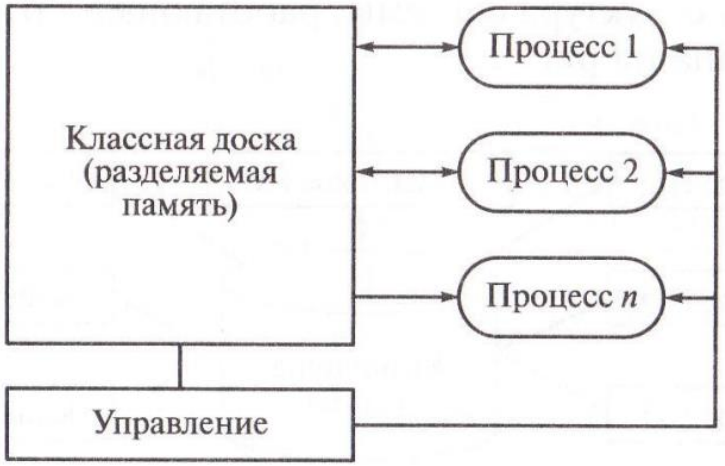
- Системы взаимодействующих процессов
  - Communicating Sequential Processes
- Системы, управляемые событиями
  - Event-Based Systems



Архитектурный стиль. Централизованные данные

- Системы, основанные на использовании базы данных
  - Database Systems

- Системы, использующие принцип классной доски
  - Blackboard Systems



Архитектурный стиль. Виртуальные машины

- Интерпретаторы
  - Interpreters
- Системы, основанные на правилах
  - Rule-Based Systems

Использование стилей

Название стиля	Условия целесообразности использования
Системы пакетно-последовательной обработки	Задачу можно разделить на четко определенные подзадачи, каждая из которых использует единственную операцию ввода-вывода. Результаты операции либо отправляются пользователю, либо поступают на вход другой подзадачи. Имеются готовые приложения, которые можно использовать для решения отдельных подзадач
Системы типа конвейеры и фильтры	Алгоритм решения задачи можно представить как совокупность повторяющихся преобразований над однотипными и независимыми друг от друга наборами данных. Число ветвлений и обратных связей минимально
Системы типа программа-сопрограмма	Порядок вычислений фиксирован, компоненты не могут делать ничего полезного, пока ждут результатов своих запросов к другим компонентам. Использование механизма наследования не дает существенных выгод.
Объектно-ориентированные системы	Можно существенно уменьшить трудозатраты на разработку системы за счет использования в процессе разработки механизма наследования. Объекты находятся на разных хостах

Использование стилей

Название стиля	Условия целесообразности использования
Клиент-серверные системы	Задачу можно сформулировать как совокупность запросов, с которыми клиенты обращаются к серверу, т.е. разделить на части, выделив клиентскую часть, отвечающую за формирование задания серверу, и серверную, отвечающую за выполнение запроса. Каждый клиент может обращаться к серверу в любое время. Сервер может только отвечать на запросы. Число клиентов может меняться, и они могут обращаться к серверу с разными запросами
Иерархические многоуровневые системы	Задачу можно представить в виде совокупности слоев с четко определенными интерфейсами, требуется иметь разные варианты бизнес-логики для разных слоев. Важна переносимость приложения между платформами, важна возможность использовать уже существующие реализации слоев
Системы взаимодействующих процессов	Передача сообщений – достаточный механизм взаимодействия для процессов. Нет большого объема долгоживущих централизованных данных.
Системы управляемые событиями	Система по своей природе асинхронна. Функционирование системы инициируется асинхронными событиями. Система может быть реализована как совокупность независимых процессов, функционирующих на разных платформах. Потребители событий отделены от сигнализаторов; важна масштабируемость в форме добавления процессов, переключаемых событиями

Использование стилей

Название стиля	Условия целесообразности использования
Системы, основанные на использовании централизованной базе данных	Задачи можно разделить между выдающими запросы и отрабатывающими запросы или между производителями и потребителями данных. Главный вопрос – хранение, представление, управление и поиск больших объемов связанных долгоживущих данных. Порядок исполнения компонентов определяется потоком входных запросов по доступу/обновлению данных; данные высокоструктурированы; доступна и экономична коммерческая СУБД
Системы, использующие принцип классной доски	Имеется большое число клиентов, которые общаются между собой для решения общей задачи. Важна масштабируемость в форме добавления потребителей данных.
Интерпретаторы	Проектирование вычислений, когда нет реальной машины, или ее использование затруднено, требуется скрыть специфику платформы, пользователю необходимо предоставить возможность использовать либо стандартный, либо скриптовый язык программирования
Системы, основанные на правилах	Алгоритм решения задачи можно описать в терминах множества правил и способов их применения. Правила могут меняться конечными пользователями

Архитектура клиент/сервер

- 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
- Системы клиент-очередь-клиент
- Одноранговые приложения
- Серверы приложений
- Преимущества:
  - Большая безопасность
  - Централизованный доступ к данным
  - Простота обслуживания

Компонентная архитектура

- Компонентная архитектура
  - Дизайн приложения разлагается на функциональные или логические компоненты с возможностью повторного использования, предоставляющие тщательно проработанные интерфейсы связи
- Преимущества:

  пригодность для повторного использования, замещаемость, независимость от контекста, расширяемость, инкапсуляция, независимость, простота развертывания, меньшая стоимость, простота разработки, возможность повторного использования, упрощение с технической точки зрения

Проблемо-ориентированная архитектура

- Дизайн на основе модели предметной области
  - Объектно-ориентированный архитектурный стиль, ориентированный на моделирование сферы деловой активности и определяющий бизнес-объекты на основании сущностей этой сферы
- Преимущества:

✓ Обмен информацией

✓ Расширяемость

✓ Удобство тестирования

Многослойная архитектура

- Многослойная архитектура
  - Функциональные области приложения разделяются на многослойные группы (уровни)
- Преимущества:

✓ Абстракция

✓ Изоляция

✓ Управляемость

✓ Производительность

✓ Возможность повторного использования

✓ Тестируемость

Архитектура, основанная на шине сообщений

- 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
- Преимущества:

✓ Расширяемость
✓ Невысокая сложность
✓ Гибкость
✓ Слабое связывание
✓ Масштабируемость
✓ Простота приложения
✓ Сервисная шина предприятия ESB
✓ Шина Интернет-сервисов ISB

N-уровневая / 3-уровневая архитектура

- N / 3 - уровневая архитектура
  - Функциональность выделяется в отдельные сегменты, во многом аналогично многослойному стилю, но в данном случае сегменты физически располагаются на разных компьютерах
- Преимущества:

✓ Удобство поддержки

✓ Масштабируемость

✓ Гибкость

✓ Доступность

Объектно-ориентированная архитектура

- Объектно-ориентированная архитектура
  - Парадигма проектирования, основанная на распределении ответственности приложения или системы между отдельными многократно используемыми и самостоятельными объектами, содержащими данные и поведение
- Преимущества:

✓ Понятность

✓ Возможность повторного использования

✓ Тестируемость

✓ Расширяемость

✓ Высокая связность

Сервисно-ориентированная архитектура

- Сервисно-ориентированная архитектура
  - Описывает приложения, предоставляющие и потребляющие функциональность в виде сервисов с помощью контрактов и сообщений
- Преимущества:

✓ Согласование предметных областей

✓ Абстракция

✓ Возможность обнаружения

✓ Возможность взаимодействия

✓ Рационализация

Паттерн (шаблон)

- Набор объектов, организованных определенным образом для решения конкретного класса задач
- Набор абстрактных классов, ориентированных на решение задач, относящихся к определенному домену

Описание паттерна	Название и тип
	Назначение
	Другие названия
	Мотивация (решаемые проблемы)
	Условия применения
	Структура паттерна в ОО нотации
	Используемые объекты и паттерны
	Результаты работы
	Рекомендации по применению
	Пример кода
	Пример использования
	Родственные паттерны

Классификация паттернов

Концептуальные паттерны

- Паттерны в терминах предметной области
- Относятся к приложению в целом, крупным подсистемам ИС

Паттерны проектирования

- паттерны в терминах разработки программных систем

Программные паттерны

- Паттерны для описания низкоуровневых понятий

Архитектурный паттерн

- Структура программной системы
- Определяет состав подсистем, их основные функции и допустимые способы компоновки

Системный паттерны

- Приложение на верхнем (системной) уровне

Структурный паттерн

- Для разделения и ли объединения элементов приложения

Поведенческий паттерн

- Для передачи управления в системе

Производящий паттерн

- Для создания объектов в систем

Системные паттерны

- Модель-Представление-Контроллер (Model-View-Controller)
- Сессия, Рабочая нить (Session, Worker Thread)
- Обратный вызов (Callback)
- Текущие обновления (Successive Update)
- Маршрутизатор (Router)
- Транзакция (Transaction)

Структурные паттерны

- Адаптер (Adapter)
- Мост (Bridge)
- Композит (Composite)
- Декоратор (Decorator)
- Фасад (Facade)
- Приспособление (Flyweight)
- Полуобъект и протокол (Half-Object Plus Protocol)
- Прокси (Proxy)

Поведенческие паттерны

- Цепочка ответственности (Chain of Responsibility)
- Команда (Command)
- Интерпретатор (Interpreter)
- Итератор (Iterator)
- Медиатор (Mediator)
- Моментальный снимок (Memento)
- Состояние (State)
- Посетитель (Visitor)
- Метод шаблона (Template Method)

Производящие паттерны

- Абстрактная фабрика (Abstract Factory)
- Строитель (Builder)
- Метод фабрики (Factory Method)
- Прототип (Prototype)
- Одиночка (Singleton)

Паттерны параллельного программирования

- Однопоточное выполнение
- Охраняемая приостановка
- Объект блокировки
- Отмена
- Планировщик
- Блокировка чтения/записи
- Производитель-потребитель
- Двухфазное завершение
- Двойная буферизация
- Асинхронная обработка

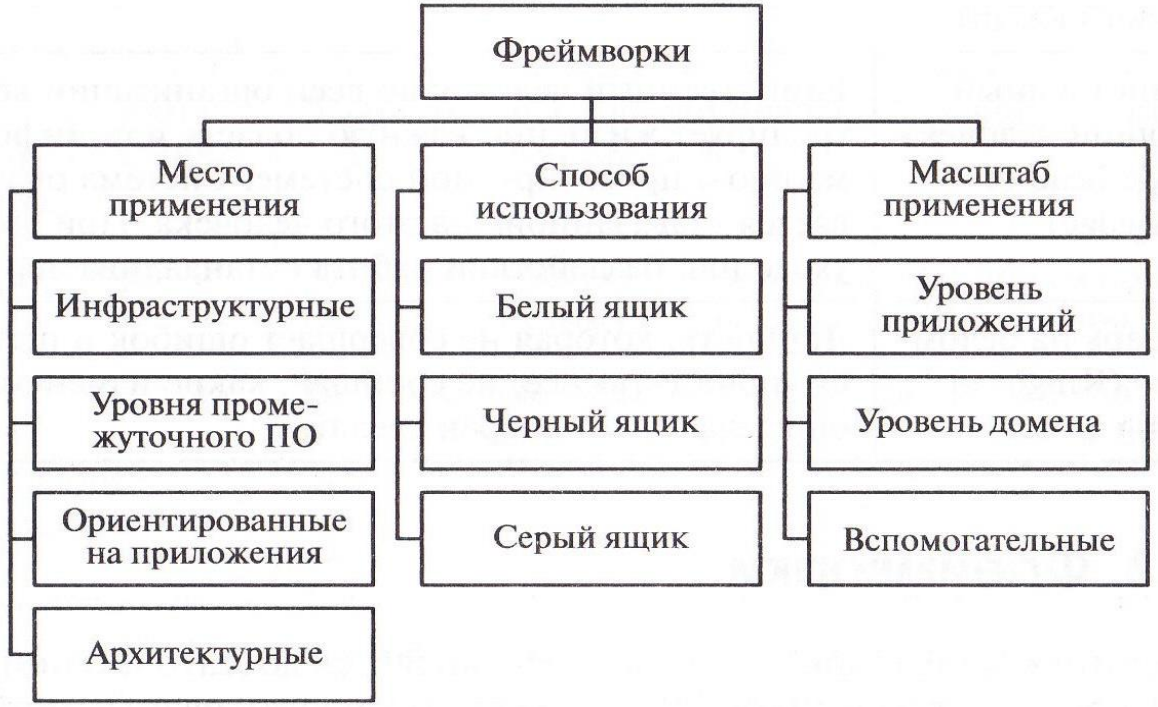
Антипаттерны

Самостоятельно

- или ловушки – классы наиболее часто внедряемых плохих решений проблем
- В управлении разработкой
- В разработке
- В объектно-ориентированном проектировании
- В области программирования
- Методологические
- Организационные

Фреймворки

- Фреймворк (каркас) – набор типовых решений, методик проектирования и классов, которые могут быть использованы для решения множества сходных задач
- Общепринятые архитектурно-структурные решения и подходы к проектированию ИС



Фреймворки. По месту применения

Инфраструктурные

- Упрощает разработку инфраструктурных элементов
- Операционные системы
- Внутренне использование

Уровня промежуточного ПО

- Интеграция распределенных приложений и компонентов

Ориентированные на приложения

- Поддержание процесса разработки систем, ориентированных на конечного пользователя и принадлежащих определенному домену

Архитектурные

- Совокупность соглашений, принципов и практик, используемых для описаний архитектур и принятых применительно к некоторому предметному домену (ISO/IEC 42010)

Фреймворки. По способу использования

Белый ящик

- Архитектурные фреймворки
- Используется наследование и динамическое связывание
- Требуется информация о классах фреймворка

Черный ящик

- Фреймворки, управляемые данными
- Композиция компонентов и их параметризация
- Проще использовать, сложнее разрабатывать

Серый ящик

- Комбинация белого и черного ящика

Фреймворки. По масштабу применения

Уровень приложения

- Полный набор функций
- Microsoft Foundation Classes

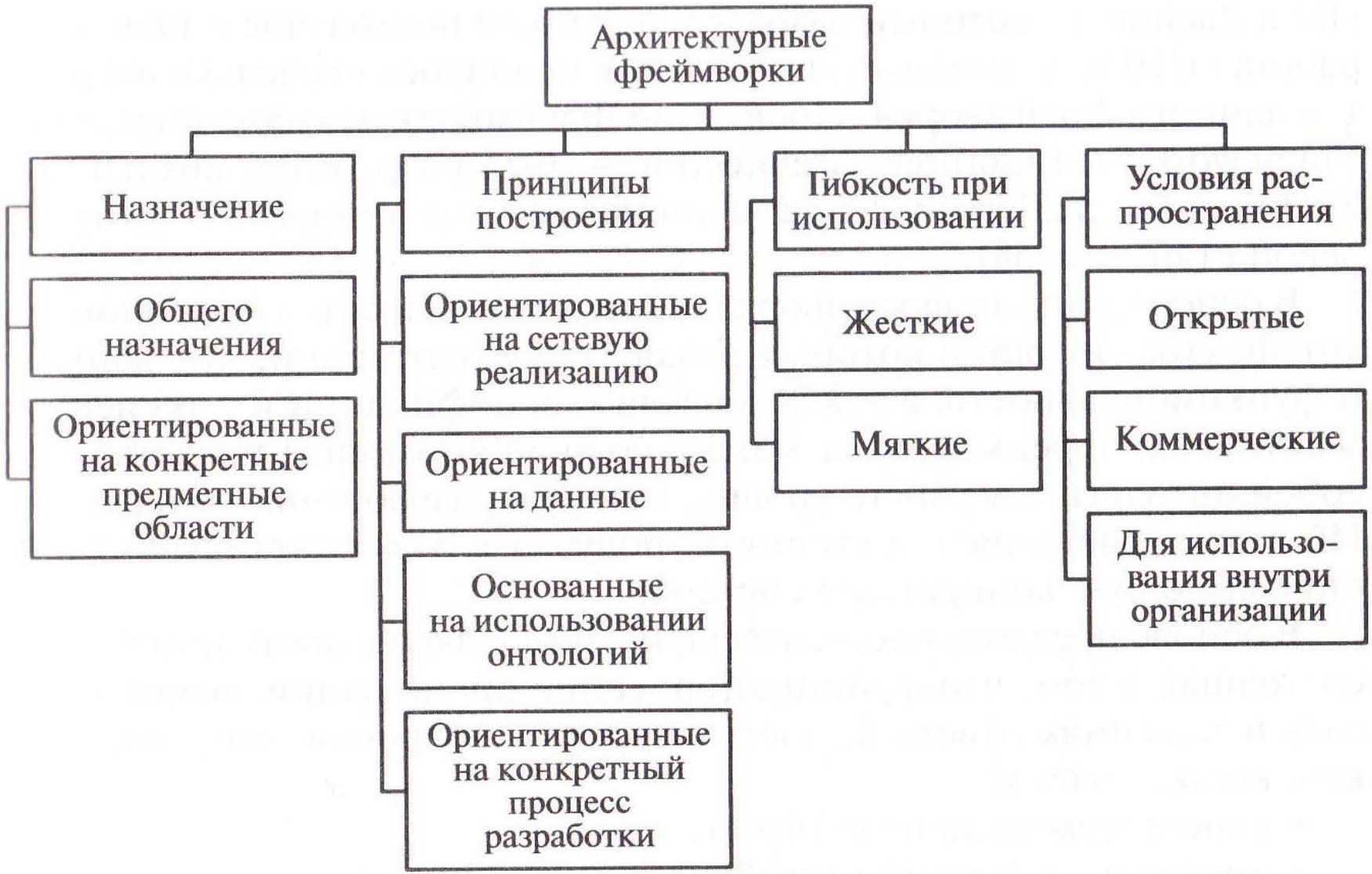
Уровень домена

- Для приложений, относящихся к определенному предметному домену

Вспомогательные

- Решение частных задач
- Управление памятью
- Управление файловой системой

Классификация архитектурных фреймворков



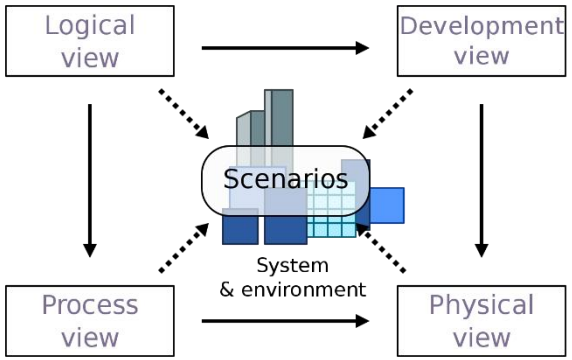
Архитектурные фреймворки

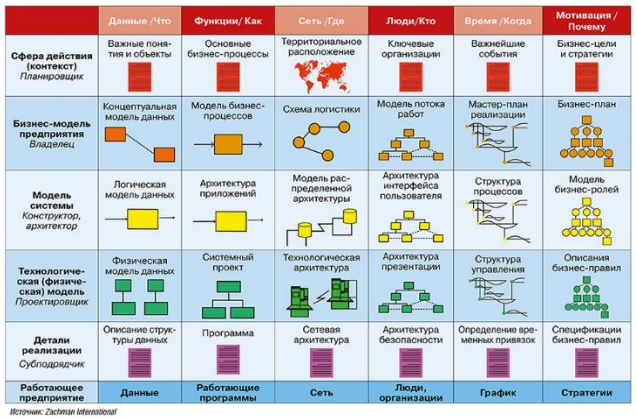
Базовые фреймворки

- 4+1
- RM-ODP (Reference Model of Open Distributed) **модель открытой распределенной обработки**
- SOMF (ProcessingService-Oriented Modeling Framework) **сервис-ориентированная** методология

Фреймворки области архитектуры предприятия

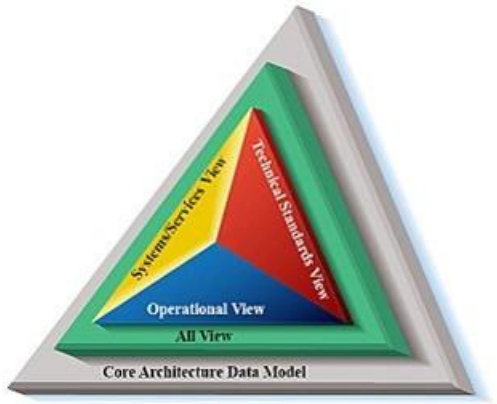
- Модель Захмана Zachman Framework
- DoDAF
- TOGAF

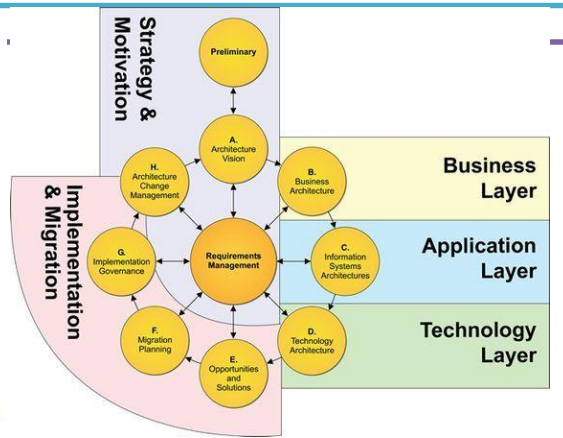




	Данные /Что	Функции/ Как	Сеть /Где	Люди/Кто	Время /Когда	Мотивация / Почему
Сфера действия (контекст) *Планировщик*	Важные понятия и объекты	Основные бизнес-процессы	Территориальное расположение	Ключевые организации	Важнейшие события	Бизнес-цели и стратегии
Бизнес-модель предприятия *Владелец*	Концептуальная модель данных	Модель бизнес-процессов	Схема логистики	Модель потока работ	Мастер-план реализации	Бизнес-план
Модель системы *Конструктор, архитектор*	Логическая модель данных	Архитектура приложений	Модель распределенной архитектуры	Архитектура интерфейса пользователя	Структура процессов	Модель бизнес-ролей
Технологическая (физическая) модель *Проектировщик*	Физическая модель данных	Системный проект	Технологическая архитектура	Архитектура презентации	Структура управления	Описания бизнес-правил
Детали реализации *Субподрядчик*	Описание структуры данных	Программа	Сетевая архитектура	Архитектура безопасности	Определение временных привязок	Спецификации бизнес-правил
Работающее предприятие	Данные	Работающие программы	Сеть	Люди, организации	График	Стратегии

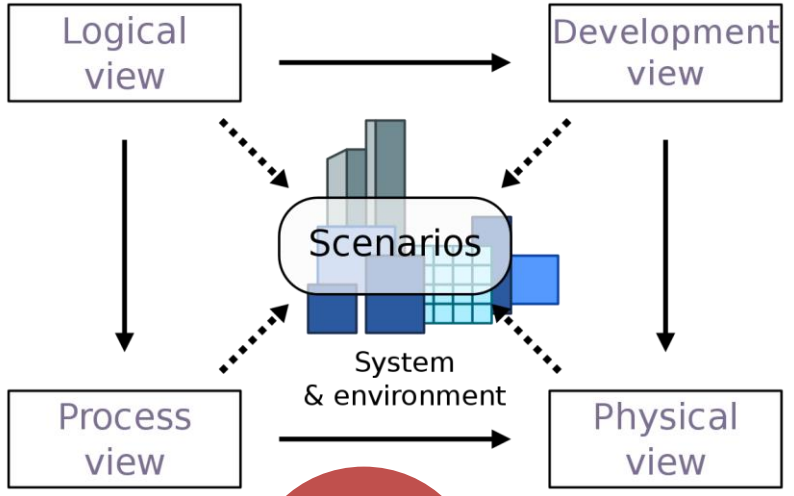
Источник: Zachman International





Модель архитектуры 4+1

- Модель представления, используемая для «описания архитектуры программно-интенсивных систем, основанных на использовании нескольких параллельных представлений»



Логическое представление	Представление процесса	***Представление разработки***	***Физический вид***	Сценарии
• связано с функциональными возможностями, которые система предоставляет конечным пользователям • реализуются в виде **схемы классов и диаграммы состояний**	• объясняет системные процессы и то, как они взаимодействуют, и фокусируется на поведении системы во время выполнения. • рассматривает параллелизм, распределение, интегратор, производительность и масштабируемость и т. д. • представляется **схемой последовательностей, схемой связи, схемой деятельности.**	• иллюстрирует систему с точки зрения программиста и связано с управлением программным обеспечением. • также называется представлением реализации • UML-схемы включают **схему пакетов и схему компонентов**	• он же представление развертывания • изображает систему с точки зрения системного инженера • касается топологии программных компонентов на физическом уровне, а также физических связей между этими компонентами. • представляется схемой развертывания	описывают последовательности взаимодействий между объектами и между процессами используются для идентификации архитектурных элементов, а также для иллюстрации и проверки архитектурного дизайна служат отправной точкой для испытаний прототипа архитектуры. Представлены схемой **вариантов использования**.