Оглавление

[Работа № 1 ПОСТРОЕНИЕ МОДЕЛИ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ. НОТАЦИЯ IDEF0 1](#_Toc175823091)

[Работа №2 ПОСТРОЕНИЕ МОДЕЛИ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ.ОПТИМИЗАЦИЯ БИЗНЕС-ПРОЦЕССОВ 2](#_Toc175823092)

[Работа №3 ПОСТРОЕНИЕ МОДЕЛИ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ В НОТАЦИИ IDEF3. 3](#_Toc175823093)

[Работа №4 «СОЗДАНИЕ DFD-МОДЕЛИ БИЗНЕС ПРОЦЕССА» 6](#_Toc175823094)

[Работа №5 «ПОСТРОЕНИЕ UML-МОДЕЛИ БИЗНЕС-ПРОЦЕССА» 8](#_Toc175823095)

[Варианты предметной области: 14](#_Toc175823096)

[Работа 6 Анализ архитектуры клиент-серверной системы с позиции дальнейшей модификации и масштабирования. Анализ рисков проекта. 15](#_Toc175823097)

[Работа 7. Анализ архитектуры клиент-серверной системы с учетом трех типов архитектур: монолитной, модульной и микросервисной. Оценка их потенциала для модификации и масштабирования. 22](#_Toc175823098)

[Работа 8. Анализ применения шаблонов и паттернов проектирования в клиент-серверных системах 24](#_Toc175823099)

[Список рекомендуемой литературы 25](#_Toc175823100)

# Работа № 1 ПОСТРОЕНИЕ МОДЕЛИ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ. НОТАЦИЯ IDEF0

**Цель работы:**

1 Знакомство с графической нотацией формализации и описания бизнес-процессов IDEF0. Знакомство c понятием функциональной модели AS-IS («как есть»).

2 Описание и построение функциональной модели AS-IS выбранной предметной области с применением нотации IDEF0.

**Постановка задачи:**

Для заданной предметной области разработать модель AS-IS. Вы можете выбрать один из вариантов процессов, описанных в приложении, или предложить свой вариант.

**Модель должна содержать:**

* не менее трёх уровней;
* на диаграмме 1-го уровня (A0) должно быть не менее 4-х функциональных блоков;
* на диаграмме 2-го и далее уровнях должно быть не менее 2-х функциональных блоков.

**Указания:**

Модель AS-IS – это модель «как есть», т. е. модель уже существующего процесса/функции. Анализ процессов является обязательной частью любого проекта создания или развития системы. Построение функциональной модели AS-IS позволяет четко зафиксировать, какие процессы осуществляются на предприятии, какие информационные объекты используются при выполнении функций различного уровня детализации.

Модель AS-IS показывает зоны ответственности исполнителей процесса и ход самого процесса («кто что сделал», как взаимосвязаны этапы между собой и как каждый этап влияет на конечный результат). Функциональная модель AS-IS является отправной точкой для анализа потребностей предприятия, выявления проблем и «узких» мест, разработки проекта совершенствования деловых процессов. Анализ функциональной модели AS-IS позволяет понять, в чем заключается проблема, в чем будут состоять преимущества новых процессов и каким изменениям подвергнется существующая структура организации процесса в результате оптимизации.

# Работа №2 ПОСТРОЕНИЕ МОДЕЛИ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ.ОПТИМИЗАЦИЯ БИЗНЕС-ПРОЦЕССОВ

**Цель работы:**

1. Знакомство с понятием функциональной модели TO-BE («как будет»).
2. Доработка созданной модели AS-IS с учетом выявленных недостатков в организации бизнес-процессов.

**Постановка задачи:**

Для заданной предметной области преобразовать созданную модель AS-IS в модель TO-BE. Внедрив информационную систему или клиент-серверную архитектуру.

**Модель должна содержать:**

* не менее трёх уровней;
* на диаграмме 1-го уровня (A0) должно быть не менее 4-х функциональных блоков;
* на диаграмме 2-го и далее уровнях должно быть не менее 2-х функциональных блоков.

**Указания:**

Найденные в модели AS-IS недостатки исправляются путем создания модели ТО-ВЕ («как будет»), т. е. модели новой организации процессов на предприятии. Создание и внедрение информационной системы (ИС) приводит к изменению условий выполнения отдельных операций, структуры процессов и предприятия в целом.

Функциональная модель TO-BE позволяет уже на стадии проектирования будущей ИС определить эти изменения. Применение функциональной модели TO-BE позволяет не только сократить сроки внедрения информационной системы, но также снизить риски, связанные с невосприимчивостью персонала к информационным технологиям. Модель ТО-ВЕ нужна для анализа альтернативных (лучших) путей выполнения функции и документирования того, как компания будет делать бизнес в будущем.

# Работа №3 ПОСТРОЕНИЕ МОДЕЛИ ДЕЯТЕЛЬНОСТИ ПРЕДПРИЯТИЯ В НОТАЦИИ IDEF3.

**Цель работы:**

Получить практические навыки в построении IDEF3-модели бизнес-процесса.

**Постановка задачи:**

С помощью методологии IDEF3 декомпозировать 1 из функциональных блоков модели окружения (A0), используя все типы перекрестков.

**Модель должна содержать:**

Перекрестки типа AND, OR, XOR

**Указания:**

Для описания логики взаимодействия информационных потоков наиболее подходит IDEF3, называемая также workflow diagramming - методологией моделирования, использующая графическое описание информационных потоков, взаимоотношений между процессами обработки информации и объектов, являющихся частью этих процессов. Диаграммы Workflow могут быть использованы в моделировании бизнес-процессов для анализа завершенности процедур обработки информации. С их помощью можно описывать сценарии действий сотрудников организации, например последовательность обработки заказа или события, которые необходимо обработать за конечное время. Каждый сценарий сопровождается описанием процесса и может быть использован для документирования каждой функции.

IDEF3 - это метод, имеющий основной целью дать возможность аналитикам описать ситуацию, когда процессы выполняются в определенной последовательности, а также описать объекты, участвующие совместно в одном процессе,

Техника описания набора данных IDEF3 является частью структурного анализа. В отличие от некоторых методик описаний процессов IDEF3 не ограничивает аналитика чрезмерно жесткими рамками синтаксиса, что может привести к созданию неполных или противоречивых моделей.

IDEF3 может быть также использован как метод создания процессов. IDEF3 дополняет IDEF0 и содержит все необходимое для построения моделей, которые в дальнейшем могут быть использованы для имитационного анализа.

Каждая работа в IDEF3 описывает какой-либо сценарий бизнес-процесса и может являться составляющей другой работы. Поскольку сценарий описывает цель и рамки модели, важно, чтобы работы именовались отглагольным существительным, обозначающим процесс действия, или фразой, содержащей такое существительное.

Точка зрения на модель должна быть задокументирована. Обычно это точка зрения человека, ответственного за работу в целом. Также необходимо задокументировать цель модели - те вопросы, на которые призвана ответить модель.

**Диаграммы.**

Диаграмма является основной единицей описания в IDEF3.

Единицы работы - Unit of Work (UOW). UOW, также называемые работами (activity), являются центральными компонентами модели. В IDEF3 работы изображаются прямоугольниками с прямыми углами и имеют имя, выраженное отглагольным существительным, обозначающим процесс действия, одиночным или в составе фразы, и номер (идентификатор); другое имя существительное в составе той же фразы обычно отображает основной выход (результат) работы, например, "Изготовление изделия".

**Связи.**

Связи показывают взаимоотношения работ. Все связи в IDEF3 однонаправлены и могут быть направлены куда угодно, но обычно диаграммы IDEF3 стараются построить так, чтобы связи были направлены слева направо. В IDEF3 различают три типа стрелок, изображающих связи:

* Старшая (Precedence) - сплошная линия, связывающая единицы работ (UOW), Рисуется слева направо или сверху вниз. Показывает, что работа-источник должна закончиться прежде, чем работа-цель начнется.
* Отношения (Relational Link) - пунктирная линия, использующаяся для изображения связей между единицами работ (UOW) а также между единицами работ и объектами ссылок.
* Потоки объектов (Object Flow) - стрелка с двумя наконечниками, применяется для описания того факта, что объект используется в двух или более единицах работы, например, когда объект порождается в одной работе и используется в другой.

Старшая связь и поток объектов. Старшая связь показывает, что работа-источник заканчивается ранее, чем начинается работа-цель. Часто результатом работы-источника становится объект, необходимый для запуска работы-цели. В этом случае стрелку, обозначающую объект, изображают с двойным наконечником. Имя стрелки должно ясно идентифицировать отображаемый объект. Поток объектов имеет ту же семантику, что и старшая стрелка.

Перекрестки (Junction). Окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончания нескольких работ. Перекрестки используются для отображения логики взаимодействия стрелок при слиянии и разветвлении или для отображения множества событий, которые могут или должны быть завершены перед началом следующей работы. Различают перекрестки для слияния (Fan-in Junction) и разветвления (Fan-out Junction) стрелок. Перекресток не может использоваться одновременно для слияния и для разветвления. Для внесения перекрестка служит кнопка в палитре инструментов - добавить в диаграмму перекресток Junction. В диалоге Junction Type Editor необходимо указать тип перекрестка. Смысл каждого типа приведен в таблице 1.

Таблица 1 - Типы перекрестков

|  |  |  |  |
| --- | --- | --- | --- |
| **Обозначение** | **Наименование** | **Смысл в случае слияния стрелок** | **Смысл в случае разветвления стрелок** |
|  | Asynchronous AND | Все предшествующие процессы должны быть завершены | Все следующие процессы должны быть запущены |
|  | Synchronous AND | Все предшествующие процессы завершены одновременно | Все следующие процессы запускаются одновременно |
|  | Asynchronous OR | Один или несколько предшествующих процессов должны быть завершены | Один или несколько следующих процессов должны быть запущены |
|  | Synchronous OR | Один или несколько предшествующих процессов завершены одновременно | Один или несколько следующих процессов запускаются одновременно |
|  | XOR (Exclusive OR) | Только один предшествующий процесс завершен | Только один следующий процесс запускается |

В отличие от IDEF0 в IDEF3 стрелки могут сливаться и разветвляться только через перекрестки.

Декомпозиция работ. В IDEF3 декомпозиция используется для детализации работ. Методология IDEF3 позволяет декомпозировать работу многократно, т.е. работа может иметь множество дочерних работ. Это позволяет в одной модели описать альтернативные потоки.

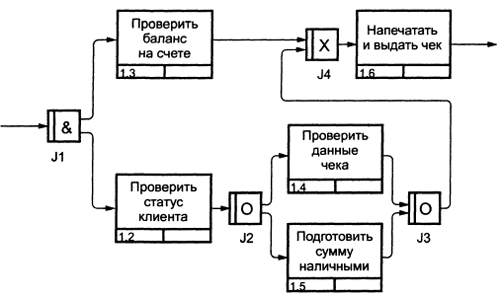


Рис. 0. Пример диаграммы в нотации IDEF3.

# Работа №4 «СОЗДАНИЕ DFD-МОДЕЛИ БИЗНЕС ПРОЦЕССА»

**Цель работы:** получить практические навыки в построении DFD-модели бизнес-процесса.

**Постановка задачи:**

С помощью методологии DFD декомпозировать 1 из функциональных блоков. Можно выбрать часть процесса, который моделировался на предыдущих лабораторных работах. При выборе учтите, что процесс обязательно должен предусматривать обработку информации, лучше, чтобы это была автоматизированная обработка с использованием одной или нескольких информационных систем.

**Знакомство с основами методологии DFD.**

Диаграммы потоков данных (Data flow diagramming, DFD) используются для описания документооборота и обработки информации. Подобно IDEF0, DFD представляет модельную систему как сеть связанных между собой работ. Их можно использовать как дополнение к модели IDEF0 для более наглядного отображения текущих операций документооборота в корпоративных системах обработки информации. DFD описывает:  процессы обработки информации (работы);  потоки данных (стрелки, arrow), которые могут моделировать и потоки материальных объектов (изделия, документы);  внешние ссылки (external references), которые обеспечивают интерфейс с внешними объектами, находящимися за границами моделируемой системы;  хранилища данных (data store). Как и в IDEF0, основными элементами DFD-диаграмм являются функциональные блоки, которые называются процессами или работами. Они преобразуют входы в выходы (чаще всего это преобразование входных данных в выходные). Блоки соединяются стрелками. В отличие от стрелок IDEF0, которые представляют собой жесткие ограничения на работу блоков, стрелки DFD показывают, как объекты (как правило, данные) двигаются от одной работы к другой. Это представление потоков совместно с хранилищами данных и внешними сущностями делает модели DFD более похожими на описание физических характеристик системы – движения объектов (data flow), хранения объектов (data stores), поставки и распространения объектов (external entities). При построении диаграмм потоков данных наиболее часто используют две нотации: Йордана и Гейна-Сарсона. Обе нотации имеют одинаковый по названиям и значению элементный состав, но имеют различное его графическое изображение.

**Элементы DFD диаграммы**

Выделяют 4 элемента в диаграмме:

1. Процесс.

Процессы, при которых идет изменение потока данных (обработка, трансформация и др. изменения). Процесс, как и в других диаграммах обычно прописывается с помощью глагола, например: “Отправка заполненной формы”.

1. Внешняя сущность.

Сущность (объект), которая получает или отправляете данные при взаимодействии с описанным процессом.

1. Хранилище данных.

Все хранилища данных или отдельные файлы, которые хранят исходные или выходные данные, а также все промежуточные хранилища.

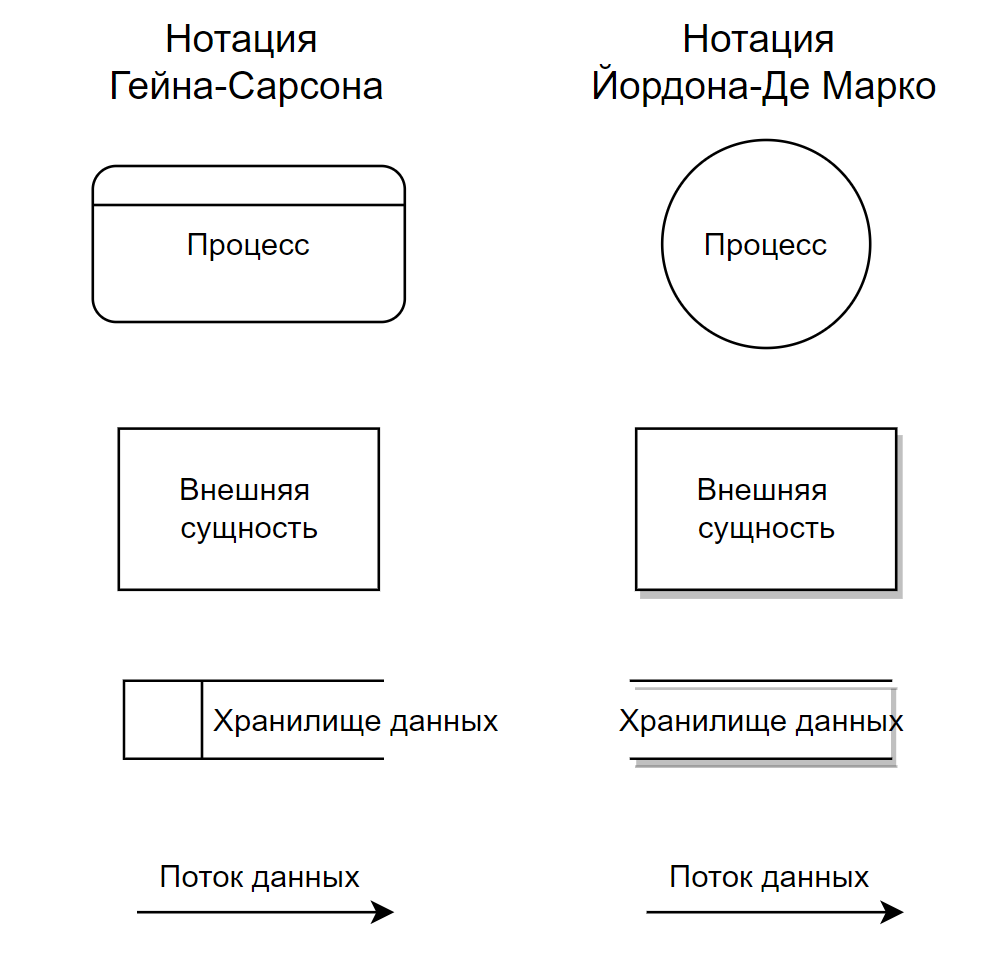
1. Поток данных.

Поток данных, который отображает направление и сами данные, которые перемещаются между внешними сущностями и хранилищами данных с помощью процессов.

Несколько правил построения диаграмм:

* Процесс должен иметь входной и выходной поток данных.
* Хранилища данных также должны иметь входные и выходные потоки данных.
* Данные с внешних сущностей должны обязательно проходить через процесс чтобы попасть в хранилище.

В DFD диаграммах также выделяют 2 разные нотации. Поэтому стоит обращать внимание на условные обозначения каждого элемента в зависимости от используемой нотации. Ниже представлена картинка сравнения элементов разных нотаций.



# Работа №5 «ПОСТРОЕНИЕ UML-МОДЕЛИ БИЗНЕС-ПРОЦЕССА»

**Цель работы:** получить практические навыки в построении прецедентной UML-модели бизнес-процесса.

**Порядок выполнения работы.**

Выберите бизнес-процесс, для которого будете формировать модель. Вы можете выбрать один из вариантов процессов, описанных в приложении, или предложить свой вариант. Можно выбрать один из процессов, для которого на предыдущих лабораторных работах строилась модель по одной из структурных методологий. Желательно, чтобы процесс имел различные версии, т.е. альтернативные потоки событий.

Для заданной предметной области:

* построить диаграмму классов;
* построить диаграмму последовательности;
* построить диаграмму взаимодействий (диаграмму коммуникаций);
* построить диаграмму пакетов;

**Знакомство с основами языка моделирования UML.**

Унифицированный язык моделирования UML (Unified Modeling Language) предназначен для описания, визуализации и документирования бизнес-систем на базе объектно-ориентированного подхода с целью последующего использования моделей бизнес-процессов для реализации их в виде программного обеспечения. Бурное развитие объектно-ориентированных языков программирования, сопровождающееся возрастанием сложности прикладных программ, вызвало потребность в создании объектно-ориентированного языка для формирования предварительной модели предметной области, для которой разрабатывается программа.

Такая модель необходима заказчикам, программистам и менеджерам проекта по созданию информационной системы для того, чтобы они могли выработать общий взгляд на цели и функции системы. И хотя модели предметной области, формируемые с помощью UML, предназначены, прежде всего, для последующей реализации в виде программного обеспечения, они имеют и самостоятельную ценность, т.к. позволяют наглядно отобразить функции и процессы бизнес-системы, объекты, участвующие в бизнеспроцессах, их отношения, а также динамику выполнения процессов. Начало работ над созданием унифицированного объектно-ориентированного языка моделирования относится к середине 1990-х годов. К тому времени уже было разработано более 50 различных языков объектно-ориентированного моделирования. Авторы наиболее распространенных языков – Г. Буч, Д. Румбах и А. Джекобсон, – собравшись «под крылом» компании Rational Software Corporation, начали работу над унифицированным методом. Ими был создан ряд версий унифицированного метода, который они назвали Unified Modeling Language (UML). В настоящее время большинством производителей информационных систем и такими комитетами по стандартам, как ANSI и OMG, язык UML был признан в качестве стандарта.

В технологии реинжиниринга бизнес-процессов, пожалуй, впервые UML стали применять не только и не столько для создания информационных систем (ИС), сколько для анализа и перепроектирования бизнеса. Вместо моделей процессов, реализуемых информационной системой, строятся модели бизнес-процессов, даже если они и не будут подвергнуты автоматизации, вместо объектов ИС (программных объектов) в моделях отражаются объекты бизнеса (исполнители, продукция, услуги и т.д.), вместо окружения ИС (пользователей ИС) моделируется окружение бизнеса (поставщики, партнеры, клиенты).

В рамках языка UML все представления о модели сложной системы фиксируются в виде специальных графических конструкций (схем, графов), получивших название диаграмм. Предполагается, что никакая единственная модель не может с достаточной степенью адекватности описывать различные аспекты сложной системы. Таким образом, модель сложной системы состоит из некоторого числа диаграмм, каждая из которых отражает некоторый аспект поведения или структуры системы.

Диаграммы UML подразделяют на два типа — это структурные диаграммы и диаграммы поведения.

Структурные диаграммы показывают статическую структуру системы и ее частей на разных уровнях абстракции и реализации, а также их взаимосвязь. Элементы в структурной диаграмме представляют значимые понятия системы и могут включать в себя абстрактные, реальные концепции и концепции реализации. Существует семь типов структурных диаграмм:

* Диаграмма составной структуры
* Диаграмма развертывания
* Диаграмма пакетов
* Диаграмма профилей
* Диаграмма классов
* Диаграмма объектов
* Диаграмма компонентов

Диаграммы поведения показывают динамическое поведение объектов в системе, которое можно описать, как серию изменений в системе с течением времени. А к диаграммам поведения относятся:

* Диаграмма деятельности
* Диаграмма прецедентов
* Диаграмма состояний
* Диаграмма последовательности
* Диаграмма коммуникаций
* Диаграмма обзора взаимодействия
* Временная диаграмма

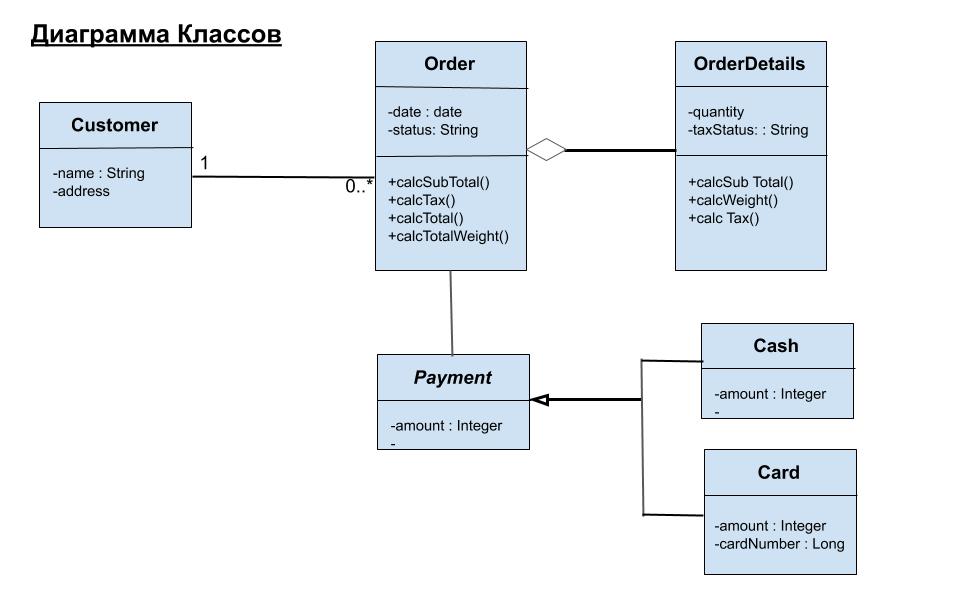
Диаграмма вариантов использования представляет собой наиболее общую концептуальную модель системы, которая является исходной для построения всех остальных диаграмм. Представление вариантов использования детализируется с помощью диаграмм состояний, деятельности, последовательности и кооперации. Диаграммы классов используются для представления логической структуры информационной системы, диаграммы компонентов и диаграммы развертывания – для представления физических компонентов информационной системы.

**Диаграмма классов**

Диаграмма классов — это центральная методика моделирования, которая используется практически во всех объектно-ориентированных методах. Эта диаграмма описывает типы объектов в системе и различные виды статических отношений, которые существуют между ними.

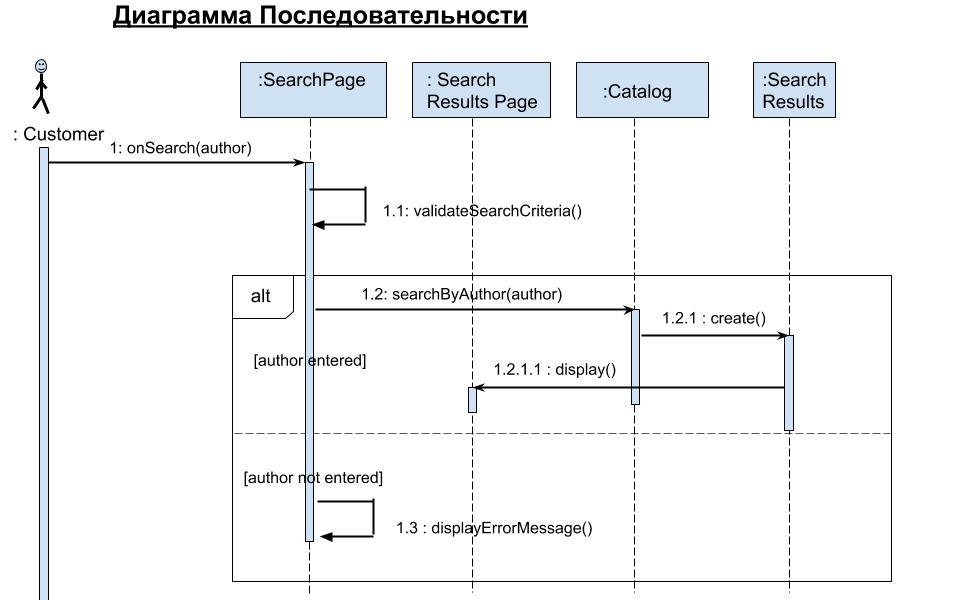
Три наиболее важных типа отношений в диаграммах классов (на самом деле их больше), это:

* Ассоциация, которая представляет отношения между экземплярами типов, к примеру, человек работает на компанию, у компании есть несколько офисов.
* Наследование, которое имеет непосредственное соответствие наследованию в Объектно-Ориентированном дизайне.
* Агрегация, которая представляет из себя форму композиции объектов в объектно-ориентированном дизайне.



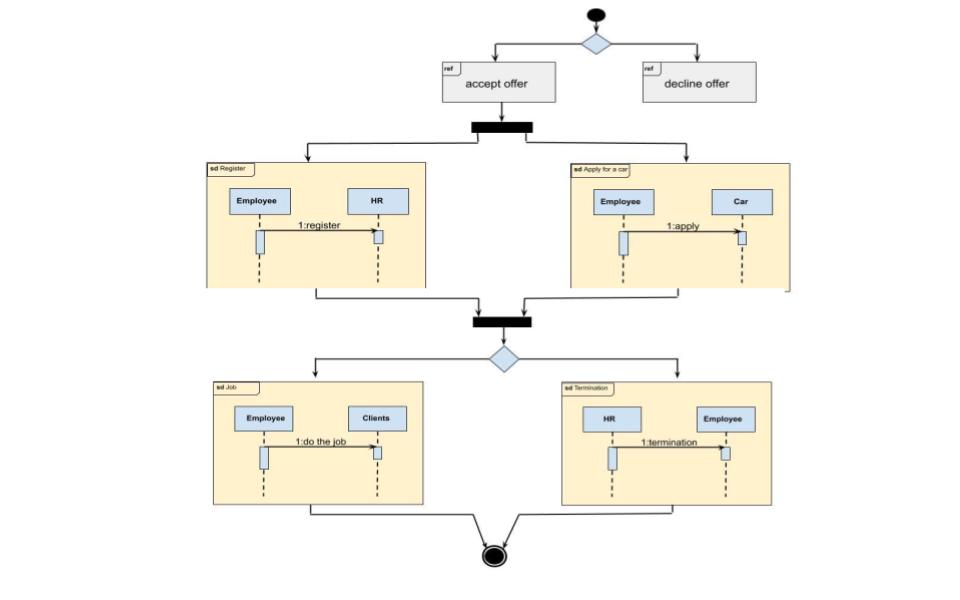
**Диаграмма последовательности**

Диаграмма последовательности моделирует взаимодействие объектов на основе временной последовательности. Она показывает, как одни объекты взаимодействуют с другими в конкретном прецеденте.



**Диаграмма обзора взаимодействия**

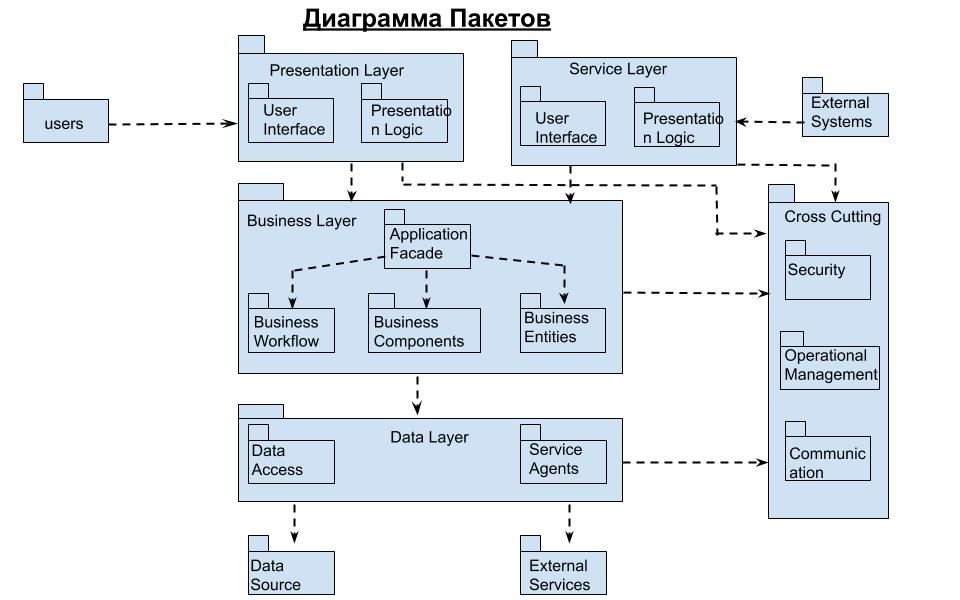
Диаграмма обзора взаимодействий фокусируется на обзоре потока управления взаимодействиями. Это вариант Диаграммы деятельности, где узлами являются взаимодействия или события взаимодействия. Диаграмма обзора взаимодействий описывает взаимодействия, в которых сообщения и линии жизни скрыты. Мы можем связать «реальные» диаграммы и добиться высокой степени навигации между диаграммами внутри диаграммы обзора взаимодействия.



**Диаграмма пакетов**

Диаграмма пакетов — это структурная схема UML, которая показывает пакеты и зависимости между ними.

Она позволяет отображать различные виды системы, например, легко смоделировать многоуровневое приложение.



# Варианты предметной области:

1. Моделирование бизнес-процессов розничного магазина;
2. Моделирование бизнес-процессов гостиницы;
3. Моделирование бизнес-процессов салона красоты;
4. Моделирование бизнес-процессов стоматологической клиники;
5. Моделирование бизнес-процессов ресторана;
6. Моделирование бизнес-процессов кафедры в университете;

# Работа 6 Анализ архитектуры клиент-серверной системы с позиции дальнейшей модификации и масштабирования. Анализ рисков проекта.

Задание:

Рассмотрите факторы, которые могут повлиять на масштабируемость системы, такие как архитектура, базы данных, алгоритмы, и другие.

Рассмотрите, какие изменения в архитектуре или инфраструктуре могут потребоваться для обеспечения требуемой масштабируемости.

По результатам анализа сделать мотивированный выбор архитектурной модели для проекта по выбранной тематике, описать возможности масштабирования проектируемой системы и связанные с этим риски.

Предложите стратегии масштабируемости, такие как использование облачных ресурсов, контейнеризация, кэширование, балансировка нагрузки и другие могут быть применены в вашем проекте.

Раскройте необходимые риски непосредственно в рамках вашего проекта.

Рассчитайте какое количество пользователей или запросов система должна обслуживать в вашем проекте?

Краткие теоретические сведения:

Когда речь идет об архитектурных моделях, существует несколько основных типов, каждый из которых определяет, как организованы компоненты системы и как они взаимодействуют друг с другом. Вот несколько ключевых архитектурных моделей:

1. Клиент-Сервер (Client-Server):

- В этой модели система разделена на две основные части: клиенты, которые обращаются за данными или услугами, и сервер, который предоставляет эти данные или услуги. Клиенты и серверы взаимодействуют через сеть.

- Примеры: веб-сервер и веб-браузер (HTTP), почтовые серверы и клиенты (SMTP, IMAP), игровые серверы и клиенты.

2. Peer-to-Peer (P2P):

- В этой модели нет центрального сервера. Каждый участник системы (пир) может действовать как клиент и сервер одновременно, обмениваясь данными или ресурсами непосредственно друг с другом.

- Примеры: файлообменные сети, такие как BitTorrent, VoIP-сети, децентрализованные криптовалюты, такие как Bitcoin.

3. Сервер в облаке (Cloud Server):

- Эта модель представляет собой размещение серверов и хранение данных на удаленных серверах в облаке (чаще всего в центральных дата-центрах), с которыми клиенты могут взаимодействовать через интернет.

- Примеры: облачные вычисления, хранение данных в облаке, веб-сервисы.

4. Микросервисы (Microservices):

- В этой модели приложение разбивается на небольшие, независимые сервисы, которые могут взаимодействовать друг с другом через API. Каждый микросервис выполняет определенную функцию.

- Примеры: архитектура микросервисов в веб-приложениях, где каждый микросервис может иметь свою собственную базу данных и логику.

5. Событийно-Ориентированная Архитектура (Event-Driven Architecture):

- В этой модели система строится вокруг потоков событий, где компоненты (или сервисы) реагируют на события и генерируют события для других компонентов.

- Примеры: системы обработки потока данных, системы реакции на события в реальном времени.

6. Серверная архитектура на стороне клиента (Client-Side Serverless Architecture):

- В этой модели бизнес-логика выполняется на стороне клиента, и сервер используется только для хранения и предоставления данных. Это связано с популярностью одностраничных приложений (SPA).

- Примеры: веб-приложения с клиентской бизнес-логикой и API для доступа к данным.

Каждая из этих архитектурных моделей имеет свои преимущества и недостатки, и выбор конкретной модели зависит от требований и целей проекта. Очень важно правильно выбрать архитектурную модель, чтобы она соответствовала специфике вашего приложения или системы.

**Примеры из реальной жизни:**

1. Looky:

- Архитектурная модель: Клиент-Сервер.

- Пояснение: Looky является примером типичного клиент-серверного приложения. Мобильные клиенты общаются с серверами Looky для загрузки и отображения фотографий и видео, а также для взаимодействия с другими пользователями.

2. Вконтакте:

- Архитектурная модель: Сервер в облаке (Cloud Server).

- Пояснение: Вконтакте использует облачные серверы, чтобы обеспечить доступность и масштабируемость для своих более чем двух миллиардов пользователей. Данные, изображения и видео хранятся и обрабатываются на серверах в центральных дата-центрах.

3. BitTorrent:

- Архитектурная модель: Peer-to-Peer (P2P).

- Пояснение: BitTorrent - это пример P2P-системы, где пользователи обмениваются файлами напрямую друг с другом без необходимости центрального сервера. Это позволяет эффективно распространять большие файлы и распределять нагрузку на участников сети.

4. Fortnite (многопользовательская онлайн игра):

- Архитектурная модель: Клиент-Сервер, но с использованием многих серверов.

- Пояснение: Fortnite - это многопользовательская игра, которая использует архитектуру клиент-сервер, но с множеством игровых серверов. Каждый игровой сервер может поддерживать несколько игроков. Это позволяет игрокам взаимодействовать в режиме реального времени в общем игровом мире.

5. Twitter:

- Архитектурная модель: Событийно-Ориентированная Архитектура.

- Пояснение: Twitter использует архитектуру событийной ориентации для передачи сообщений в реальном времени. Когда пользователи отправляют твиты, эти сообщения распространяются в реальном времени всем подписанным на автора пользователям, что требует быстрой и эффективной передачи событий.

6. Кинопоиск:

- Архитектурная модель: Микросервисы (Microservices).

- Пояснение: Кинопоиск использует архитектуру микросервисов для своей инфраструктуры стриминга. Каждая часть системы является независимым микросервисом, что облегчает масштабирование и обновление отдельных компонентов.

Каждое из этих приложений и игр выбрало архитектурную модель в соответствии с его уникальными требованиями и характеристиками. Например, социальные сети, требуют обработки огромного количества данных и взаимодействия с миллионами пользователей, что делает сервер в облаке идеальным выбором. В то время как игры, такие как Fortnite, нуждаются в быстрой передаче данных в реальном времени и многих игровых серверах для обеспечения высокой доступности и производительности.

Масштабируемость:

**Масштабируемость — это ключевой аспект проектирования клиент-серверных систем, особенно в игровой индустрии, где большое количество пользователей и высокая нагрузка могут стать вызовом.**

Масштабируемость означает способность системы расширяться или уменьшаться в зависимости от изменяющейся нагрузки и количества пользователей.

Какое количество пользователей или запросов система должна обслуживать в вашем проекте?

Рассмотрите различные виды масштабируемости, включая горизонтальную (scale out) и вертикальную (Scale up) масштабируемость.

Вертикальная масштабируемость (Scale Up):

Вертикальная масштабируемость заключается в увеличении вычислительной мощности существующих серверов или ресурсов, добавлении памяти, процессоров и других ресурсов к отдельным серверам.

Этот тип масштабируемости обеспечивает увеличение производительности и ресурсов на уровне отдельных узлов.

Преимущества:

* Относительно прост в реализации, может быть дешевле на первых этапах.

Недостатки:

* Существует предел физических ресурсов на сервере, что, в принципе, ограничивает вертикальную масштабируемость.
* С некоторого момент стоимость вертикального масштабирования может расти нелинейно, причем непропорционально дорого, что также ограничивает размер такого масштабирования.

Горизонтальная масштабируемость (Scale Out):

Горизонтальная масштабируемость включает в себя добавление новых серверов или узлов к системе, чтобы увеличить ее производительность и способность обработки нагрузки.

Этот тип масштабируемости подразумевает распределение нагрузки между множеством небольших серверов, работающих параллельно.

Преимущества:

* Позволяет легко увеличивать производительность и обеспечивать высокую доступность.

Недостатки:

* Требует сложной инфраструктуры и управления, а также поддержки балансировки нагрузки.

Риски масштабируемости:

Вот несколько распространенных рисков и способы их смягчения:

1. Риск масштабируемости:

- Потенциальный риск: Неправильный выбор архитектурной модели может привести к нехватке масштабируемости, особенно если приложение сталкивается с ростом количества пользователей.

- Способы уменьшения риска: Проведите анализ масштабируемости на ранних этапах проектирования, выберите модель, которая поддерживает вертикальную и горизонтальную масштабируемость, и учтите возможность использования облачных ресурсов.

2. Риск производительности:

- Потенциальный риск: Некорректная архитектурная модель может привести к недостаточной производительности приложения, особенно в условиях высокой нагрузки.

- Способы уменьшения риска: Проведите бенчмаркинг и оптимизацию на ранних этапах, используйте кэширование и асинхронные операции для улучшения производительности.

3. Риск безопасности:

- Потенциальный риск: Некорректный выбор архитектурной модели может создать уязвимости и угрозы безопасности для системы и данных пользователей.

- Способы уменьшения риска: Используйте передовые методы шифрования и аутентификации, обеспечьте контроль доступа и регулярно проводите аудит безопасности.

4. Риск сложности разработки и поддержки:

- Потенциальный риск: Сложные архитектурные модели могут привести к затратам времени и ресурсов на разработку, тестирование и поддержку.

- Способы уменьшения риска: Рассмотрите использование более простых и понятных архитектур, разрабатывайте документацию и обучайте команду.

5. Риск обновлений и масштабирования:

- Потенциальный риск: Некорректное выбор архитектурной модели может затруднить обновления и масштабирование приложения в будущем.

- Способы уменьшения риска: Изучите модели, которые обеспечивают легкость обновлений и масштабирования, такие как микросервисы или облачные решения.

6. Риск обеспечения качества и тестирования:

- Потенциальный риск: Сложная архитектурная модель может затруднить тестирование и обеспечение качества продукта.

- Способы уменьшения риска: Разработайте стратегию тестирования, включающую функциональное, интеграционное и производительностное тестирование, и используйте автоматизацию.

# Работа 7. Анализ архитектуры клиент-серверной системы с учетом трех типов архитектур: монолитной, модульной и микросервисной. Оценка их потенциала для модификации и масштабирования.

Здание: выполните анализ архитектуры клиент-серверной системы, в последовательности, указанной далее:

Далее предполагает использовать выбранную ранее студентом тему практик и рассматривать в рамках своего проекта, с учетом первый практик, где был IDEF0 AS-IS и TO-BE

1. Монолитная архитектура:

- Оцените, какая часть системы на данный момент построена на монолитной архитектуре.

- Рассмотрите структуру и зависимости между компонентами. Обратите внимание на единое местоположение всех функциональных блоков.

- Укажите ограничения и риски в контексте модификации и масштабирования.

2. Модульная архитектура:

- Проанализируйте наличие модульных компонентов в системе и их связи.

- Рассмотрите, какие части системы могут быть легко модифицированы или расширены без внесения значительных изменений в другие модули.

- Оцените гибкость модульной системы с точки зрения модификации и масштабирования.

3. Микросервисная архитектура:

- Идентифицируйте существующие микросервисы в системе и оцените, насколько они согласованы и независимы друг от друга.

- Оцените готовность микросервисной системы к масштабированию в условиях изменяющейся нагрузки.

- Рассмотрите потенциальные слабые места и риски, связанные с координацией и управлением микросервисами.

4. Изменения в архитектуре:

- Опишите, какие изменения в архитектуре или инфраструктуре могут потребоваться для обеспечения требуемой масштабируемости для каждого типа архитектуры.

- Укажите, какие архитектурные решения или технические изменения могут быть применены, чтобы обеспечить модификацию и масштабирование.

5. Стратегии масштабирования:

- Предложите стратегии масштабирования, такие как использование облачных ресурсов, контейнеризация, кэширование, балансировка нагрузки, автоматическое масштабирование и другие, которые могут быть применены в каждом типе архитектуры.

6. Риски:

- Идентифицируйте риски, связанные с масштабированием каждого типа архитектуры. Рассмотрите проблемы производительности, надежности, безопасности и управления данными.

- Опишите меры по снижению рисков и управлению ими.

7. Оценка нагрузки:

- Рассчитайте, какое количество пользователей или запросов система должна обслуживать для каждого типа архитектуры. Учтите текущие и будущие потребности.

8. Выбор архитектурной модели:

- Сделайте мотивированный выбор архитектурной модели для вашего проекта, учитывая требования к модификации и масштабированию.

Это необходимо для того, чтобы определить, какой тип архитектуры наилучшим образом соответствует требованиям вашего проекта и какие стратегии масштабирования и управления рисками следует использовать.

# Работа 8. Анализ применения шаблонов и паттернов проектирования в клиент-серверных системах

Цель задания: провести анализ и изучение применения шаблонов и паттернов проектирования в контексте клиент-серверных систем.

Задание:

1. Рассмотрите следующие шаблоны и паттерны:

- Шаблоны клиент-серверных систем (например, шаблоны для взаимодействия клиента и сервера).

- Структурные паттерны (например, "Адаптер" (Adapter) для интеграции разных интерфейсов, "Мост" (Bridge) для разделения абстракции и реализации).

- Паттерны поведения (например, "Состояние" (State) для управления состоянием системы, "Стратегия" (Strategy) для замены алгоритмов).

- Порождающие паттерны (например, "Фабричный метод" (Factory Method) для создания объектов, "Одиночка" (Singleton) для гарантированной единственной инстанциации).

2. Выбор конкретных шаблонов и паттернов: Выберите несколько конкретных шаблонов и паттернов, которые, наиболее подходят для применения в вашей клиент-серверной системе. Технически грамотно обоснуйте выбор этих паттернов.

3. Анализ применения: рассмотрите, как выбранные вами шаблоны и паттерны могут быть использованы для улучшения архитектуры и функциональности вашей системы. Опишите, как эти шаблоны и паттерны могут решать конкретные проблемы или улучшать производительность проектируемой клиент-серверной системы.

4. Проектирование и реализация: спроектируйте часть системы, используя выбранные шаблоны и паттерны. Результаты проектирования представьте в виде диаграмм и текстового описания, если потребуется добавьте глоссарий.

5. Отчет: подготовьте отчет, включающий ваш анализ применения шаблонов и паттернов, описания применения в выбранной системе и, результаты проектирования.

# Список рекомендуемой литературы

1. Гвоздева Т. В., Баллод Б. А.. Проектирование информационных систем. Стандартизация [Электронный ресурс]:учебное пособие. - Санкт-Петербург: Лань, 2019. - 252 с. – Режим доступа: https://e.lanbook.com/book/115515
2. Рочев К. В.. Информационные технологии. Анализ и проектирование информационных систем [Электронный ресурс]:учебное пособие. - Санкт-Петербург: Лань, 2019. - 128 с. – Режим доступа: https://e.lanbook.com/book/122181
3. Вейцман В. М.. Проектирование информационных систем [Электронный ресурс]:учебное пособие. - Санкт- Петербург: Лань, 2019. - 316 с. – Режим доступа: https://e.lanbook.com/book/122172
4. Остроух А. В., Суркова Н. Е.. Проектирование информационных систем [Электронный ресурс]:монография. - Санкт-Петербург: Лань, 2019. - 164 с. – Режим доступа: https://e.lanbook.com/book/118650
5. Информационно-правовой портал ГАРАНТ http:// www.garant.ru
6. Консультант Плюс http:// www.consultant.ru
7. Научная электронная библиотека eLIBRARY.ru
8. Российский технологический журнал
9. Тестирование и контроль программных систем: сайт. – URL: https://xreferat.com/33/2759-1-sushnost-i-osobennosti-ispol-zovaniya-instrumental-nogo-programmnogo-obespecheniya.html (дата обращения 15.03.2020). – Текст: электронный.
10. Контроль и корректировка кода: сайт. – URL: https://studfile.net/preview/2790134/page:3/ (дата обращения 15.03.2020). – Текст: электронный.
11. Object Constraints Language: сайт. – URL: https://ami.nstu.ru/~vms/lecture/lecture12/lecture12.htm (дата обращения 15.03.2020). – Текст: электронный.
12. Эксплуатация и сопровождение проекта: сайт. – URL: http://kgau.ru/istiki/umk/pis/l7.htm (дата обращения 15.03.2020). – Текст: электронный.
13. Типовые модели систем: сайт. – URL: http://kgau.ru/istiki/umk/pis/l37.htm (дата обращения 15.03.2020). – Текст: электронный.
14. Проектирование и разработка Java приложений и систем: сайт. – URL: https://dic.academic.ru/dic.nsf/ruwiki/608820 (дата обращения 15.03.2020). – Текст: электронный.
15. Современные CASE средства проектирования систем: сайт. – URL: http://window.edu.ru/resource/616/73616/files/kulyabov-korolkova\_formal-methods.pdf (дата обращения 15.03.2020). – Текст: электронный.
16. Реляционные СУБД: сайт. – URL: https://compress.ru/article.aspx?id=10082 (дата обращения 15.03.2020). – Текст: электронный.
17. Проектирование ER-диаграмм: сайт. – URL: http://citforum.ru/cfin/prcorpsys/infsistpr\_09.shtml (дата обращения 15.03.2020). – Текст: электронный.
18. Стандарты разработки: сайт. – URL: http://venec.ulstu.ru/lib/disk/2017/460.pdf (дата обращения 15.03.2020). – Текст: электронный.
19. Архитектура ЭВМ и систем: сайт. – URL: https://arxiv.org/ftp/arxiv/papers/1802/1802.06769.pdf (дата обращения 15.03.2020). – Текст: электронный.