Министерство науки и высшего образования
Российской Федерации
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Кузбасский государственный технический университет
имени Т. Ф. Горбачева»

Институт профессионального образования

Кафедра информатики и информационных систем

Составитель К. В. Ерошевич

Проектирование информационных систем

Методические материалы

Рекомендовано цикловой методической комиссией
Информационных систем и программирования
в качестве электронного издания
для использования в образовательном процессе

Кемерово 2026

Рецензент:
К. А. Кулиничев, преподаватель первой квалификационной категории кафедры информатики и информационных систем

Ерошевич Кирилл Владимирович
Проектирование информационных систем : методические материалы для обучающихся специальности СПО 09.02.11 «Разработка и управление программным обеспечением» / Кузбасский государственный технический университет имени Т. Ф. Горбачева, Кафедра информатики и информационных систем ; составитель К. В. Ерошевич. – Кемерово : КузГТУ, 2026. – 1 файл (917 Кб). – Текст : электронный.
.

Методические материалы по дисциплине Проектирование информационных систем: методические материалы для обучающихся специальности СПО 09.02.11 «Разработка и управление программным обеспечением» содержат указания для проведения лабораторных работ.

Лабораторная работа №1
«Выявление и первичный анализ требований»

Теоретическая часть

Понятие требования

В контексте разработки информационных систем, требование – это описание того, что система должна делать (функции), какими свойствами обладать (качество) и в каких условиях работать (ограничения). Это не просто пожелание, а формализованное условие, которое необходимо реализовать.

Зачем нужно выявление требований?

Ошибки, допущенные на этапе выявления требований, – самые дорогостоящие. Если неправильно понять задачу в начале, переделывать готовый продукт придется полностью. Цель этапа – построить общее понимание системы между заказчиком, пользователями и командой разработки.

Процесс выявления требований

Выявление (сбор) требований – это коммуникационная задача. Аналитик выступает в роли переводчика между бизнесом и IT.

Процесс включает:

1. Идентификацию стейкхолдеров (заинтересованных лиц): кто выиграет или проиграет от внедрения системы?
2. Сбор информации: Использование различных методов для извлечения знаний из голов стейкхолдеров.
3. Анализ: Отсев явно лишнего и уточнение противоречивого.

Основные методы выявления требований (детально):

1. Интервью: Беседа по заранее подготовленным вопросам. Бывает свободным (разговор на тему) и формализованным (строгая последовательность вопросов). Плюсы: прямой контакт, можно увидеть эмоции. Минусы: требует подготовки, занимает много времени.
2. Анкетирование: Письменный опрос большого количества людей. Эффективен, когда нужно собрать статистику или мнения удаленных сотрудников.
3. Анализ документов: Изучение существующих бумажных форм (накладные, счета), должностных инструкций, устаревших программ. Позволяет понять терминологию и реальные бизнес-процессы "как есть".

4. Наблюдение и погружение: Аналитик садится рядом с сотрудником и смотрит, как тот работает. Позволяет увидеть то, о чем сотрудник забывает рассказать на словах (например, использование стикеров-напоминалок или Excel-калькуляторов).
5. Мозговой штурм: Групповой метод генерации идей. Используется, когда нужно придумать что-то новое или найти решение спорной ситуации.

Результат первичного анализа:

На выходе мы получаем не структурированный документ, а "сырой" список требований (листики, стикеры, аудиозаписи) и глоссарий (словарь терминов предметной области, чтобы все говорили на одном языке).

Задание

Цель работы: Провести интервью с условным заказчиком (или проанализировать описание предметной области) и составить первичный список требований.

Описание предметной области (Вариант по умолчанию):

Небольшой книжный магазин работает в офлайн-режиме. Продавцы выписывают чеки от руки, а учет остатков книг ведется в тетради. Владелец магазина хочет заказать простую информационную систему для автоматизации работы. Он жалуется, что часто заканчиваются нужные книги, и он не может быстро посмотреть, сколько прибыли принес вчерашний день.

Ход работы:

1. Изучите описание предметной области.
2. Поставьте себя на место системного аналитика. Составьте список из 5–7 вопросов, которые вы задали бы владельцу магазина для уточнения деталей. Вопросы должны касаться разных аспектов: цели, функциональности, отчетности.
3. Основываясь на описании и (придуманных) ответах на свои вопросы, составьте список требований (не менее 10 пунктов). Пока просто запишите их "как есть", в виде простых предложений.
4. Составьте глоссарий проекта (дайте определения ключевым терминам: Книга, Склад, Чек, Продавец, Отчет, Остатки).

Контрольные вопросы

1. Что такое "требование" к информационной системе?
2. Перечислите основные методы выявления требований.
3. В чем преимущество метода "Наблюдение" перед "Интервью"?
4. С какой целью составляется глоссарий проекта на начальном этапе?

Лабораторная работа №2
«Классификация и формализация требований»

Теоретическая часть

Проблема "сырых" данных

Список, полученный в Лабораторной работе №1, похож на "кашу". В нем перемешаны глобальные цели ("система должна приносить прибыль") и мелкие детали ("кнопка должна быть синей"). Чтобы начать проектирование, этот список нужно упорядочить.

Классификация требований (Модель FURPS+)

Самый популярный способ разделить требования – это модель FURPS (и ее расширение FURPS+), где каждая буква обозначает категорию:

1. F (Functional) – Функциональные требования.
2. U (Usability) – Требования к удобству использования (Юзабилити).
3. R (Reliability) – Надежность.
4. P (Performance) – Производительность.
5. S (Supportability) – Поддерживаемость.

Формализация требований (Правила записи)

Мало разложить требования по полочкам, их нужно правильно записать, чтобы разработчик понял задачу однозначно, а тестировщик смог проверить результат. Для этого используется критерий SMART (адаптированный для аналитики):

1. Specific (Конкретное): Требование должно быть однозначным. Вместо "Быстрый поиск" нужно написать "Поиск книги по названию или автору".
2. Measurable (Измеримое): Должен быть критерий проверки. Вместо "Много книг" – > "Система поддерживает каталог до 10 000 наименований".
3. Achievable (Достижимое): Технически и финансово реализуемо в рамках проекта.
4. Relevant (Релевантное/Уместное): Требование нужно здесь и сейчас, решает конкретную бизнес-задачу.
5. Testable (Проверяемое): Должно быть понятно, как подтвердить, что требование выполнено (тест-кейс).

Инструмент описания: Use Case (Вариант использования)

Для описания сложных функциональных требований используют сценарии. Use Case описывает, как пользователь (Актер) взаимодействует с системой для достижения конкретной цели. Хороший сценарий описывает не только идеальный путь (happy path), но и альтернативные варианты (например, если книга не найдена).

Задание

Цель работы: Обработать список требований из Лабораторной работы №1, классифицировать их и составить User Story (пользовательские истории) или Use Case (варианты использования).

Ход работы:

1. Возьмите список требований, полученный в ЛР1.
2. Разделите все требования на категории: Функциональные (F), Нефункциональные (U, R, P) и Ограничения (Constraints).
3. Выберите 2–3 самых важных функциональных требования и опишите их в виде Use Case (сценария использования).
4. Проверьте одно из нефункциональных требований на соответствие критерию SMART. Если оно не соответствует, перепишите его в корректном виде.

Контрольные вопросы

1. Чем функциональные требования отличаются от нефункциональных? Приведите по 2 своих примера (не из лабораторной работы).
2. Что такое критерий SMART в анализе требований? Расшифруйте аббревиатуру.
3. Для чего нужны варианты использования (Use Case)?
4. К какой категории требований (по FURPS) относится фраза: "Система должна шифровать все персональные данные клиентов"?

Лабораторная работа №3
«Приоритизация требований и управление рисками»

Теоретическая часть

Зачем нужна приоритизация?

Ресурсы любой разработки (время, деньги, люди) всегда ограничены. Невозможно реализовать все требования сразу.

Приоритизация помогает ответить на вопросы:

1. Что сделать в первую очередь (MVP – Minimum Viable Product)?
2. От чего можно отказаться, если не хватит бюджета?
3. Что даст максимальную ценность бизнесу при минимальных затратах?

Некоторые методы приоритизации требований

Метод MoSCoW. Самый популярный метод в гибкой разработке (Agile). Все требования делятся на 4 категории (таблица 1):

Таблица 1

Категория	Значение	Описание
Must have	Обязательно	Без этих требований система не имеет смысла. Критически важные функции. Если их не сделать – проект провален
Should have	Желательно	Важные функции, но без них систему можно запустить. Обычно их добавляют в следующую очередь
Could have	Можно сделать	"Фишки", которые сделают жизнь удобнее, но легко заменяются обходными путями. Реализуются, если осталось время
Won't have	Не будем делать	Сознательный отказ от реализации сейчас. Эти требования переносятся в будущие версии или отбрасываются совсем

Матрица Value vs Effort "Ценность/Сложность"

Визуальный метод. Для каждого требования оценивается:

1. Бизнес-ценность (насколько это полезно, по шкале от 1 до 10)

2. Сложность реализации (трудозатраты, по шкале от 1 до 10)

Далее требования наносятся на матрицу (таблица 2):

Таблица 2

Ценность\Сложность	Низкая	Высокая
Низкая	Делаем в последнюю очередь	**ДЕЛАЕМ В ПЕРВУЮ ОЧЕРЕДЬ!** (Low Hanging Fruits)
Высокая	Не делаем	Делаем, но планируем ресурсы

Понятие риска в проектировании ИС

Риск – это потенциальное событие, которое может негативно повлиять на проект (сроки, бюджет, качество) или, в редких случаях, позитивно.

Управление рисками – это не пессимизм, а профессиональный подход: "Предупрежден – значит вооружен".

Классификация рисков:

1. Технические: сломается сервер, несовместимость версий, сложность интеграции.
2. Организационные: уволится ключевой разработчик, заказчик перестанет выходить на связь.
3. Рыночные: конкуренты выпустили аналогичный продукт, сменилось законодательство.
4. Оценочные: неправильно оценили сложность задачи (ошибка в сроках).

Процесс управления рисками:

1. Идентификация: Составить список того, что может пойти не так. (Методы: мозговой штурм, анализ прошлого опыта).
2. Анализ (Оценка): Оценить Вероятность (В) и Силу влияния (С) по шкале (например, от 1 до 5).
3. Планирование: Разработать стратегию реагирования.
4. Мониторинг и контроль: Следить за "красными флажками" (индикаторами, что риск активизировался).

Стратегии реагирования на риски:

1. Уклонение (Избегание): Сделать что-то, чтобы риск исчез (например, купить лицензионное программное оборудование (ПО), чтобы не блокировали за установку пиратского).

2. Передача (Transfer): Переложить риск на кого-то другого (застраховать сервер, нанять подрядчика на аутсорс).
3. Снижение (Mitigation): Уменьшить вероятность или влияние (регулярное резервное копирование данных (бэкапы), написание тестов).
4. Принятие (Acceptance): Согласиться с риском, если он дешевле, чем его предотвращение (просто иметь запасной план на случай неблагоприятных погодных условий, например, дождя).

Задание

Цель работы: Научиться расставлять приоритеты среди требований и выявлять потенциальные риски проекта.

Описание ситуации:

Вы продолжаете работу над системой для книжного магазина. У вас есть список требований (возьмите итоговый список из ЛР №2). Владелец магазина сообщил, что бюджет ограничен, а запустить систему нужно через 2 месяца. Не все можно успеть.

Ход работы:

Часть 1. Приоритизация требований (MoSCoW)

1. Возьмите список функциональных требований (из ЛР №2).
2. Представьте, что вы планируете первую версию продукта (MVP). Распределите все функциональные требования по категориям MoSCoW.
3. Оформите результат в виде таблицы или списка.

Часть 2. Матрица "Ценность/Сложность"

1. Выберите 5 любых требований (можно из разных категорий).
2. Оцените каждое по шкале от 1 до 5:
    - Ценность для бизнеса (насколько владелец магазина будет рад).
    - Сложность реализации (сколько времени/денег это съест).
3. Начертите матрицу 2×2 (как в теории) и разместите на ней ваши требования (можно просто написать номера или названия).
4. Сделайте вывод: какое требование нужно делать в первую очередь (высокая ценность + низкая сложность)?

Часть 3. Управление рисками

1. Проведите мозговой штурм и составьте список из 5–7 потенциальных рисков для проекта книжного магазина.
2. Для каждого риска оцените:
   - Вероятность (Высокая / Средняя / Низкая)
   - Влияние (Критическое / Среднее / Незначительное)
3. Заполните таблицу (таблица 3):

Таблица 3

№	Риск	Вероят-ность	Влияние	Стратегия реагирова-ния	План действий (что сделаем?)
1	Сбой электричес-тва во время продажи	Средняя	Критичес-кое	Снижение	Внедрить автосохране-ние чека и журналирова-ние действий
2	...	...	...	...	...

Контрольные вопросы

1. Для чего нужно приоритизировать требования?
2. Расшифруйте аббревиатуру MoSCoW. Что означает каждая категория?
3. В чем суть матрицы "Ценность/Сложность"?
4. Что такое "риск проекта"?
5. Перечислите четыре основные стратегии реагирования на риски. Приведите пример каждой стратегии для ИТ-проекта.
6. Почему категория "Must have" не должна составлять 100 % всех требований?

Лабораторная работа №4
«Создание глоссария и документирование ограничений»

Теоретическая часть

Зачем нужен глоссарий?

В любом проекте участвуют люди с разным бэкграундом: заказчики говорят на бизнес-языке, разработчики – на техническом, аналитики пытаются найти компромисс.

Глоссарий (или словарь проекта) – это документ, который фиксирует единую терминологию, чтобы все говорили на одном языке.

Проблемы, которые решает глоссарий:

1. Неоднозначность: Слово "Клиент" для продавца – это покупатель в магазине, для бухгалтера — контрагент в договоре, для программиста — таблица в базе данных.
2. Понимание требований: Если в требовании написано "Система должна удалять клиента", что значит "удалять"? Физически стирать из базы? Помечать как неактивного? Глоссарий дает ответ.
3. Онбординг новых сотрудников: Новый программист читает глоссарий и быстро входит в курс дела.

Структура глоссария

Обычно глоссарий оформляется в виде таблицы со следующими колонками (таблица 4):

Таблица 4

Термин	Определение	Синонимы	Примечание/Пример
Уникальное название понятия	Четкое, однозначное описание	Другие слова, которыми могут называть то же самое	Уточнения, примеры, ссылки

Требования к определениям:

1. Не должно быть круговых ссылок: Нельзя определять "Книгу" через "Издание", а потом "Издание" через "Книгу".
2. Не должно быть синонимов внутри определения: Если есть термин "Чек", в определении "Квитанция" не должна использоваться без пояснения, что это одно и то же.

3. Контекстуальность: Определение должно отражать смысл именно в рамках вашего проекта.

Понятие ограничений (Constraints)

В отличие от требований, которые говорят о том, что система должна делать, ограничения говорят о том, в каких рамках система должна существовать. Ограничения нельзя приоритизировать методом MoSCoW – их нужно соблюдать, иначе проект не будет принят или станет невозможным.

Документирование ограничений

Ограничения рекомендуется выносить в отдельный раздел документа требований или явно помечать тегом [ОГРАНИЧЕНИЕ], чтобы разработчики понимали, что это не обсуждается, а должно быть выполнено.

Задание

Цель работы: Систематизировать терминологию проекта и выявить все внешние и внутренние рамки, влияющие на разработку.

Часть 1. Расширение и формализация глоссария

1. Возьмите глоссарий, который вы начали составлять в ЛР №1.
2. Добавьте в него новые термины, которые появились в процессе работы над ЛР №2 и ЛР №3 (например: "Приоритет", "Риск", "MVP", "Функциональное требование" – но уже в контексте вашего магазина, если эти понятия стали частью системы).
3. Проанализируйте список требований и найдите в нем понятия, которые требуют уточнения. Например, если есть требование "Система должна формировать отчет", нужно определить, что такое "Отчет". Добавьте эти термины в глоссарий.
4. Оформите глоссарий в виде таблицы (таблица 5):

Таблица 5

Термин	Определение (в контексте книжного магазина)	Синонимы	Пример / Примечание
Книга	Товарная единица, имеющая название, автора, ISBN, цену и количество на складе	Товар, Издание, Литература	Физическая книга, электронные не учитываем
Чек	Электронный документ, фиксирующий факт продажи одной или нескольких книг	Транзакция, Продажа, Запись о продаже	Содержит дату, список книг, итоговую сумму
Отчет	Документ, формируемый системой на основе данных о продажах за период	Сводка, Выгрузка, Статистика	Например, отчет по прибыли за день
...	...	...	...

5. Проверьте, чтобы в определениях не было круглых ссылок (А = Б, Б = А).

Часть 2. Выявление и документирование ограничений

1. Изучите описание проекта книжного магазина и историю вашей работы с ним.
2. Подумайте, какие рамки существуют у этого проекта. Ограничения могут быть явными (написаны в задании) или скрытыми (логически вытекают из ситуации).
3. Заполните таблицу классификации ограничений (таблица 6). Придумайте недостающие ограничения самостоятельно, опираясь на типовые ситуации.

Таблица 6

№	Описание ограничения	Тип (Категория)	Влияние на проект	Источник (почему это ограничение возникло?)
1	Бюджет проекта – 500 000 рублей	Бизнес-ограничение	Нельзя нанять большую команду, придется делать простой MVP	Слова владельца магазина
2	Система должна работать на компьютерах продавцов с Windows 7	Техническое	Нельзя использовать современные технологии, не поддерживаемые этой ОС	В магазине старые компьютеры, денег на обновление нет
3	Хранение персональных данных покупателей (ФИО, телефон) должно быть безопасным	Законодательное	Нужно шифрование, усложняется разработка	Требование закона о персональных данных
4	Интерфейс должен быть крупным и простым, чтобы продавцы могли работать в нем, не отвлекаясь от покупателей	UX-ограничение	Минимум полей ввода, максимум кнопок	Наблюдение за работой продавцов

№	Описание ограничения	Тип (Категория)	Влияние на проект	Источник (почему это ограничение возникло?)
5	Система должна выгружать данные в существую-щую бухгалтерску ю программу "1С: Бухгалтерия"	Техническое	Потребуется разработка модуля экспорта или API	У компании уже есть бухгалтерия на 1С
6	(Придумайте свое)	...	...	

4. Сделайте вывод: какие из найденных ограничений являются самыми критичными и могут "убить" проект, если их не учесть?

Часть 3. Итоговый документ (Концепция)

Объедините результаты ЛР1-4 в единый черновик документа "Концепция системы" (Vision).

Структура может быть такой:

- Цель проекта (Зачем мы это делаем?).
- Глоссарий (Таблица из Части 1).
- Основные функциональные требования (Список из ЛР №2).
- Приоритеты (MoSCoW) (Из ЛР №3).
- Ограничения проекта (Таблица из Части 2).
- Основные риски (Из ЛР №3).

Контрольные вопросы

1. Для чего нужен глоссарий в проекте по разработке ПО?
2. Каким требованиям должно удовлетворять "хорошее определение" термина в глоссарии?
3. Чем ограничения (constraints) отличаются от нефункцио-нальных требований (non-functional requirements)?

(Подсказка: нефункциональные требования – это "качество" системы, а ограничения – это "рамки".)

4. Перечислите основные виды ограничений. Приведите пример каждого вида для проекта интернет-магазина.
5. Почему ограничения нельзя поместить в категорию "Won't have" (не будем делать) по методу MoSCoW?

Лабораторная работа №5
«Моделирование бизнес-процессов»

Теоретическая часть

Что такое бизнес-процесс?

Бизнес-процесс – это совокупность взаимосвязанных действий (операций), которые преобразуют входные ресурсы в выходной результат, ценный для клиента или компании.

Простыми словами: это ответ на вопрос "КТО и ЧТО делает, в какой последовательности, чтобы получить нужный результат?".

Зачем моделировать бизнес-процессы?

Прежде чем автоматизировать хаос, нужно понять, как компания работает сейчас. Моделирование помогает:

1. Увидеть узкие места: Где возникают задержки? Где делают лишнюю работу?
2. Достичь согласия: Заказчик и аналитик смотрят на одну картинку и спорят не о словах, а о схеме.
3. Подготовиться к автоматизации: Разработчикам нужно понимать логику переходов и состояний.
4. Обучение: На схемах учат новых сотрудников.

Нотации моделирования (языки описания)

BPMN (Business Process Model and Notation)

Самая популярная нотация для сложных процессов. Позволяет описывать процессы с участием нескольких исполнителей (ролей) – так называемые "пулы" и "дорожки".

Основные элементы:

1. События (круги): Начало, конец, промежуточное событие (например, получение счета).
2. Действия (прямоугольники): Задачи, которые выполняет человек или система.
3. Шлюзы (ромбы): Точки принятия решения ("да/нет", выбор одного из путей).
4. Потоки (стрелки): Показывают последовательность.
5. Пулы и дорожки: Разделяют ответственность (например, дорожка "Продавец", дорожка "Система").

IDEF0

Используется для функционального моделирования. Показывает не столько последовательность, сколько то, что входит в работу и что ее контролирует. Каждый блок – это функция. Взаимодействие функции с внешней средой и внутренними факторами показывают с помощью стрелок четырёх типов:

1. Вход: Что преобразуется (сырье, данные).
2. Выход: Результат.
3. Управление: Чем руководствуются (инструкции, законы).
4. Механизм: Кто делает (человек, станок, программа).

UML (Activity Diagrams)

Диаграммы деятельности в UML очень похожи на BPMN. Используются программистами для описания логики алгоритма или варианта использования.

Концепция AS-IS и TO-BE

1. AS-IS (Как есть): Модель текущего состояния. Она часто выявляет недостатки: долгие согласования, дублирование функций, потерю данных. Заказчик говорит "у нас все хорошо", а схема показывает "вот здесь у вас ручная работа и очереди".
2. TO-BE (Как будет): Модель будущего состояния после внедрения системы. Именно эту модель будет реализовывать разработчик.

Описание процесса текстом

Любую схему должен сопровождать текстовый регламент:

1. Цель процесса.
2. Владелец процесса (кто отвечает за результат).
3. Участники.
4. Входные данные.
5. Выходные данные (результат).
6. Основные шаги.

Задание

Цель работы: Построить модели бизнес-процессов книжного магазина в нотации AS-IS и TO-BE, проанализировать изменения после автоматизации.

Инструменты: Работу можно выполнять на бумаге (аккуратно от руки), в любом графическом редакторе (Draw.io, Visio, Lucidchart) или даже в Word с помощью фигур.

Часть 1. Моделирование процесса AS-IS (Как есть сейчас)

1. Изучите описание текущей работы магазина:

   Продавцы работают так: покупатель приносит книги на кассу. Продавец берет тетрадь учета остатков, проверяет, есть ли книги в наличии (если книга есть в тетради – значит, она есть на полке). Затем продавец вручную выписывает бумажный чек (пишет название книги, цену, считает сумму на калькуляторе). После продажи продавец идет в тетрадь учета и ручкой вычитает одну книгу из остатков. В конце дня продавец пересчитывает выручку по бумажным чекам и записывает итог в "Тетрадь отчетности" для владельца.
2. Постройте диаграмму процесса "Продажа книги в магазине" в нотации AS-IS.
3. Проанализируйте полученную схему. Найдите и выпишите не менее трех проблем текущего процесса (узкие места):

Часть 2. Моделирование процесса TO-BE (Как будет с системой)

1. Представьте, что система, которую вы проектировали в прошлых работах, уже внедрена.
2. Постройте диаграмму процесса "Продажа книги в магазине" в нотации TO-BE (с использованием ИС).

Часть 3. Сравнительный анализ

Заполните таблицу сравнения процессов (таблица 7):

Таблица 7

Критерий	Процесс AS-IS (Ручной)	Процесс TO-BE (Автоматизированный)
Время на одну продажу	(Оцените в минутах)	(Оцените в минутах)
Риск ошибки (где?)	Ошибки в подсчете, потеря тетради	Сбой программы, ошибочный ввод штрих-кода
Актуальность остатков	Только на момент последней ручной правки	В реальном времени
Участие человека	Полное	Контроль и ввод данных
...	...	...

Часть 4. Текстовое описание (Регламент)

Составьте краткое текстовое описание процесса TO-BE по структуре:

1. Название процесса: Розничная продажа книги.
2. Владелец процесса: Старший продавец.
3. Участники: Покупатель, Продавец, Информационная система (ИС).
4. Вход: Выбранная покупателем книга.
5. Выход: Оплаченный чек, обновленный остаток в БД.
6. Описание шагов: (кратко 3–4 предложения).

Контрольные вопросы

1. Что такое бизнес-процесс? Из каких элементов он состоит?
2. Для чего нужно моделировать бизнес-процессы перед разработкой ИС?
3. В чем разница между моделями AS-IS и TO-BE?
4. Какие нотации моделирования бизнес-процессов вы знаете? Чем они отличаются?
5. Что такое "узкое место" процесса? Как его обнаружить на схеме?
6. Зачем на схеме TO-BE разделять дорожки "Продавец" и "Система"?

Лабораторная работа №6
«Анализ заинтересованных сторон и пользователей»

Теоретическая часть

Кто такие заинтересованные стороны (стейкхолдеры)?

Стейкхолдер (Stakeholder) – это любое лицо, группа лиц или организация, которая оказывает влияние на проект или испытывает на себе влияние со стороны проекта.

В проектировании ИС игнорирование интересов какой-либо группы стейкхолдеров – прямая дорога к провалу. Даже если система технически идеальна, но не удобна продавцам или не нужна владельцу — ее не будут использовать.

Классификация стейкхолдеров (таблица 8):

Таблица 8

Категория	Кто это?	Интересы в проекте
Заказчик (Sponsor)	Тот, кто выделяет деньги и принимает результат	Получить систему в срок и в рамках бюджета. Система должна решать бизнес-задачи
Пользователи (Users)	Те, кто будет непосредственно работать в системе	Удобство, скорость работы, понятный интерфейс, решение их повседневных задач
Косвенные пользователи	Те, кто использует результаты работы системы, но не вводит данные	Бухгалтер (получает отчеты), руководитель (смотрит статистику)
Руководство	Начальники пользователей	Контроль подчиненных, прозрачность работы, дисциплина
Технические специалисты	Администраторы, служба поддержки	Надежность, безопасность, удобство администрирования, документация
Регуляторы	Государственные органы	Соблюдение законодательства (налоговая, Роскомнадзор и т. д.)

Матрица стейкхолдеров

Для анализа стейкхолдеров используется матрица "Власть / Интерес" (Power/Interest Grid) (таблица №9). Она помогает понять, какую стратегию общения выбрать с каждой группой.

Таблица 9

	Низкий интерес	Высокий интерес
Высокая власть	**Сектор D: Наблюдать (Monitor)** Держать в курсе, но не перегружать. Могут вмешаться, если что-то пойдет не так	**Сектор A: Активно управлять (Manage Closely)** Ключевые стейкхолдеры. Вовлекать в процесс, согласовывать изменения
Низкая власть	**Сектор C: Минимальные усилия (Minimal Effort)** Информировать по минимуму, только по важным событиям	**Сектор B: Информировать (Keep Informed)** Держать в курсе, прислушиваться к мнению. Это обычно пользователи

Понятие "Персона" (User Persona)

В анализе требований мало знать, что пользователь – это "продавец". Нужно понимать, какой это человек. Для этого создаются **Персоны** – архетипы реальных пользователей с их характеристиками.

Структура персоны:

1. Имя и фото (вымышленные, но реалистичные).
2. Демография: Возраст, пол, образование.
3. Роль в проекте: Продавец, администратор, бухгалтер.
4. Цели: Что хочет достичь с помощью системы?
5. Боли (Pains): Что не устраивает в текущей работе? Чего боится?
6. Потребности (Gains): Что облегчит его жизнь?
7. Уровень технической грамотности: Низкий / Средний / Продвинутый.

8. Сценарий использования: Короткая история использования системы.

Зачем нужны персоны?

1. Чтобы разработчики помнили, для кого они пишут код.
2. Чтобы тестировщики проверяли сценарии от лица конкретного человека.
3. Чтобы не делать интерфейс "под себя", а делать под реального пользователя.

Задание

Цель работы: Выявить всех стейкхолдеров проекта книжного магазина, определить стратегии работы с ними и создать детальные портреты (персоны) ключевых пользователей.

Часть 1. Карта стейкхолдеров

1. Вспомните или придумайте, кто может быть заинтересован в проекте автоматизации книжного магазина (или связан с ним).
2. Составьте полный список стейкхолдеров. Минимум 5–6 ролей.
3. Для каждого стейкхолдера определите: Власть (высокая / низкая), Интерес (высокий / низкий).
4. Начертите матрицу "Власть / Интерес" (4 квадранта) и разместите в ней всех стейкхолдеров.
5. Для каждого сектора матрицы предложите стратегию взаимодействия (что делаем с этой группой?).

Часть 2. Создание пользовательских персон

1. Выберите двух ключевых пользователей системы из списка стейкхолдеров.
2. Придумайте для каждого из них детальный портрет (персону). Заполните таблицу (таблица 10):

Таблица 10

Характеристика	Персона 1: Продавец	Персона 2: Владелец
Имя	Елена	Иван Петрович
Возраст	45 лет	52 года
Образование	Среднее специальное (библиотечное)	Высшее экономическое
Семейное положение	Замужем, двое детей	Женат, взрослые дети
Опыт работы с ПК	Низкий. Умеет включать/выключать, работает в 1–2 программах, боится новых кнопок	Средний. Уверенно пользуется Excel и банк-клиентом, но сложные интерфейсы не люби.
Цели в работе	Быстро обслужить покупателя и уйти домой вовремя. Меньше ошибок, чтобы не ругал начальник	Видеть реальную картину прибыли и остатков. Понимать, какие книги продаются лучше
Боли (Pains)	Боится запутаться в новой программе. Не хочет учиться долго. Боится, что программа "сломается и все пропадет"	Не доверяет автоматическому учету, хочет перепроверять. Раздражают сложные отчеты с кучей цифр
Потребности (Gains)	Крупные кнопки, минимум полей ввода, подсказки на экране. Чтобы программа работала быстро и не тормозила	Отчеты в виде графиков и понятных итогов. Возможность посмотреть продажи за любой день на телефоне
Сценарий использования	Утром включила кассу, пробила 50 чеков, в конце дня нажала "Закрыть смену" и пошла домой	Зашел в кабинет, посмотрел, что "Война и мир" продается плохо, снизил цену

3. Для каждой персоны придумайте цитату, отражающую их отношение к системе.

Часть 3. Анализ влияния на требования

1. Вернитесь к списку требований из ЛР №2 и ЛР №3.
2. Отметьте, на какие требования повлияла та или иная персона.
3. Сделайте вывод: как анализ пользователей помогает уточнять требования?

Контрольные вопросы

1. Кто такие стейкхолдеры проекта? Чем они отличаются от пользователей?
2. Для чего нужна матрица "Власть / Интерес"? Какие стратегии работы существуют для каждого квадранта?
3. Что такое "персона" (user persona) в проектировании?
4. Какие данные обязательно нужно включать в описание персоны?
5. Почему для проектирования интерфейса важно знать "боли" и "техническую грамотность" пользователя?
6. Может ли один человек быть в разных квадрантах матрицы стейкхолдеров? Приведите пример.

Лабораторная работа №7
«Моделирование прецедентов (Use Case)»

Теоретическая часть

Что такое Use Case (Прецедент, Вариант использования)?

Use Case (Прецедент) – это описание последовательности действий, которые система выполняет в ответ на внешнее событие, инициированное актором (пользователем или другой системой), приводящее к видимому для актора результату.

Простыми словами: Это история о том, как кто-то (актор) взаимодействует с системой, чтобы получить что-то ценное.

Из чего состоит Use Case?

1. Актор (Actor): Роль пользователя или внешней системы, взаимодействующей с системой (Продавец, Покупатель, Бухгалтерская система).
2. Цель: Что актор хочет достичь.
3. Основной сценарий (Main Success Scenario): Идеальный путь без ошибок.
4. Альтернативные сценарии (Alternative Flows): Ответвления от основного пути (ошибки, исключения, другие варианты).
5. Предусловия (Pre-conditions): Что должно быть истиной до начала.
6. Постусловия (Post-conditions): Что изменится в системе после успешного выполнения.

Зачем нужна диаграмма прецедентов (Use Case Diagram)?

В UML (Unified Modeling Language) есть специальная диаграмма для визуализации прецедентов. Она не показывает детальную последовательность шагов (для этого есть диаграммы активности или последовательности), а дает общую картину: КТО (актор) может делать ЧТО (прецедент) в системе.

Основные элементы диаграммы Use Case (таблица 11):

Таблица 11

Элемент	Обозначение	Описание
Актор (Actor)	Человечек	Внешняя сущность (человек или система), взаимодействующая с системой
Прецедент (Use Case)	Овал с названием	Конкретная функция или сервис, который система предоставляет актору
Границы системы (System Boundary)	Прямоугольник с названием системы	Показывает, что внутри – наша система, а снаружи – внешний мир
Отношение ассоциации	Линия	Связывает актора с прецедентом (кто что делает)
Отношение включения (<<include>>)	Пунктирная стрелка с <<include>>	Один прецедент ВСЕГДА включает другой (обязательная часть). *Например: Прецедент "Оформить заказ" ВСЕГДА включает "Проверить наличие товара"*
Отношение расширения (<<extend>>)	Пунктирная стрелка с <<extend>>	Один прецедент МОЖЕТ расширить другой в определенных условиях (необязательная часть). *Например: Прецедент "Оформить заказ" МОЖЕТ быть расширен прецедентом "Запросить подтверждение по SMS" (если сумма большая)*
Обобщение (Generalization)	Стрелка с пустым треугольником	Один актор или прецедент является частным случаем другого. *Например: "Продавец" – частный случай "Сотрудника"*

Правила построения хорошего Use Case:

1. Ценность для актора: У каждого прецедента должен быть конкретный результат, важный для актора.

2. Атомарность: Прецедент должен быть независимым и законченным. "Ввести пароль" – не прецедент, это шаг. "Авторизоваться в системе" – прецедент.
3. Именование: Начинается с глагола ("Оформить", "Найти", "Сформировать").

Задание

Цель работы: Построить диаграмму прецедентов для системы книжного магазина и детально описать ключевые сценарии использования.

Инструменты: Любой инструмент для рисования диаграмм (Draw.io, Lucidchart, Visio) или аккуратный рисунок от руки.

Часть 1. Определение акторов и прецедентов

1. Вспомните всех пользователей системы из ЛР №6 (продавец, владелец, бухгалтер и т. д.) – это ваши потенциальные акторы.
2. Проанализируйте функциональные требования из ЛР №2 и процессы ТО-ВЕ из ЛР №5. Составьте список всех возможных прецедентов (функций, которые система предоставляет акторам).
3. Составьте два списка: Список акторов (роли), Список прецедентов (функции).

Часть 2. Построение диаграммы Use Case

1. Нарисуйте прямоугольник (границы системы), подпишите его "Книжный магазин" (или "BookStore").
2. Поместите все прецеденты (овалы) внутри прямоугольника.
3. Разместите акторов (человечков) снаружи прямоугольника (слева или справа).
4. Соедините акторов с теми прецедентами, к которым они имеют доступ, с помощью линий (ассоциаций).
5. Добавьте отношения <<include>> и <<extend>> там, где это необходимо.
6. Добавьте обобщение, если нужно. Например: "Продавец" и "Администратор" могут быть обобщены до "Сотрудник".

Часть 3. Детальное описание сценариев (Текстовый шаблон)

1. Выберите один самый важный прецедент (например, "Продажа книги").

2. Опишите его по шаблону (это будет спецификация прецедента) (таблица 12):

Таблица 12

Поле	Описание
Название	Продажа книги
Акторы	Продавец, Покупатель (как косвенный)
Краткое описание	Позволяет продавцу оформить продажу одной или нескольких книг покупателю, сформировать чек и списать товар со склада
Предусловия	Продавец авторизован в системе; Кассовая смена открыта; Книги физически находятся у покупателя
Триггер	Покупатель подошел к кассе с книгами
Основной поток	1. Продавец начинает новую продажу (нажимает "Новый чек") 2. Система создает пустой чек с текущей датой и номером 3. Продавец сканирует штрих-код книги 4. Система находит книгу в БД, проверяет наличие и добавляет в чек с ценой 5. Повторяет шаги 3–4 для всех книг 6. Продавец объявляет итоговую сумму 7. Покупатель оплачивает (наличные/карта) 8. Продавец вносит тип оплаты и полученную сумму в систему 9. Система рассчитывает сдачу (если нужно), закрывает чек 10. Система уменьшает остатки книг в БД 11. Система печатает чек (или сохраняет электронный)

Поле	Описание
Альтернативные потоки	**Альтернатива А: Книга не найдена по штрих-коду** А1. Система выводит сообщение "Книга не найдена" А2. Продавец ищет книгу вручную по названию А3. Если книга найдена — продолжение с шага 4 **Альтернатива Б: Книги нет в наличии** Б1. Система выводит сообщение "Товара нет в наличии (остаток 0)" Б2. Продавец предлагает покупателю другую книгу или завершает продажу без этой позиции **Альтернатива В: Оплата картой (терминал)** В1. Вместо шага 7: Продавец выбирает "Оплата картой" В2. Система отправляет запрос на терминал В3. После подтверждения оплаты от терминала система закрывает чек
Постусловия	Чек сохранен в базе данных; Остатки книг уменьшены; Запись о продаже отражена в отчете

Часть 4. Выводы

Объясните, как построенная диаграмма Use Case и сценарии помогают:

1. Разработчикам понять, что программировать?
2. Тестировщикам понять, что проверять?
3. Заказчику подтвердить, что мы правильно поняли его потребности?

Контрольные вопросы

1. Что такое прецедент (Use Case)? Из каких элементов он состоит?

2. Чем актор отличается от обычного пользователя? Может ли актором быть не человек?
3. Что показывают отношения <<include>> и <<extend>> на диаграмме Use Case? Приведите примеры.
4. Зачем нужна диаграмма прецедентов, если есть текстовое описание требований?
5. Что такое "основной поток" и "альтернативный поток" в сценарии?
6. Чем предусловия отличаются от постусловий?

Лабораторная работа №8
«Детальное моделирование поведения системы»

Теоретическая часть

От прецедентов к детальному поведению

В Лабораторной работе №7 мы создали диаграмму прецедентов (Use Case), которая показывает, КТО может делать ЧТО в системе. Однако для разработки этого недостаточно. Программистам нужно понять, КАК ИМЕННО будет работать система внутри: в какой последовательности вызываются функции, как объекты взаимодействуют друг с другом, какие состояния они проходят.

Для детального моделирования поведения в UML используются два основных типа диаграмм:

1. Диаграмма последовательности (Sequence Diagram) – показывает взаимодействие объектов во времени.
2. Диаграмма состояний (State Machine Diagram) – показывает жизненный цикл одного объекта.

Диаграмма последовательности (Sequence Diagram)

Назначение: Описывает, как объекты (акторы, компоненты системы) обмениваются сообщениями во времени. Это идеальный инструмент для детализации конкретного сценария Use Case.

Основные элементы (таблица 13):

Таблица 13

Элемент	Обозначение	Описание
Линия жизни (Lifeline)	Вертикальная пунктирная линия от прямоугольника с именем объекта	Представляет объект во времени
Актор (Actor)	Прямоугольник с человечком или просто прямоугольник <<actor>>	Внешний инициатор взаимодействия
Сообщение (Message)	Стрелка от одной линии жизни к другой	Вызов метода, отправка сигнала
Сообщение возврата (Return Message)	Пунктирная стрелка	Возврат результата (можно не рисовать, если очевидно)
Фокус управления (Activation Bar)	Узкий прямоугольник на линии жизни	Период, когда объект выполняет действие
Альтернативы (Alt)	Прямоугольник с пунктирной линией внутри, разделенный на секции	Ветвление (if/else)
Цикл (Loop)	Прямоугольник с надписью loop	Повторение действий
Опционально (Opt)	Прямоугольник с надписью opt	Действие, которое может выполняться, а может и нет

Диаграмма состояний (State Machine Diagram)

Назначение: Показывает, через какие состояния проходит объект в течение своего жизненного цикла и что вызывает переход из одного состояния в другое. Обычно строится для одного важного класса (объекта), например: "Заказ", "Чек", "Пользователь".

Основные элементы (таблица 14):

Таблица 14

Элемент	Обозначение	Описание
Состояние (State)	Прямоугольник со скругленными углами	Ситуация в жизни объекта (например, "Чек открыт", "Чек оплачен")
Начальное состояние (Initial State)	Закрашенный кружок	С чего начинается жизнь объекта
Конечное состояние (Final State)	Закрашенный кружок в ободке	Где объект "умирает" или завершает цикл
Переход (Transition)	Стрелка	Движение из одного состояния в другое
Событие (Event)	Надпись на стрелке	Что вызвало переход (например, "оплата получена")
Условие (Guard)	В квадратных скобках	Условие, при котором переход возможен

Зачем нужны эти диаграммы?

Для разработчиков: По диаграмме последовательности легко написать код, так как видна последовательность вызовов методов.

Для аналитиков: Диаграмма состояний помогает выявить пропущенные сценарии (например, "А что будет с чеком, если покупатель ушел, а книги уже отсканированы?").

Для тестировщиков: Понимание состояний помогает писать тесты на все возможные переходы.

Задание

Цель работы: Детализировать поведение системы для ключевого прецедента с помощью диаграммы последовательности и описать жизненный цикл основного объекта.

Часть 1. Построение диаграммы последовательности (Sequence Diagram)

1. Возьмите прецедент "Продажа книги", который вы детально описывали в ЛР №7 (текстовый сценарий).
2. Определите список объектов (участников) взаимодействия.

3. Постройте диаграмму последовательности для основного потока (Happy Path) прецедента "Продажа книги".
4. Добавьте на диаграмму фокус управления (активационные прямоугольники) на линиях жизни, чтобы показать, когда объект активен.
5. Добавьте фрагмент loop для циклического сканирования нескольких книг.

Часть 2. Построение диаграммы состояний (State Machine Diagram)

1. Выберите ключевой объект системы, у которого есть ярко выраженный жизненный цикл. Лучший кандидат – "Чек" (Receipt).
2. Подумайте, через какие состояния может проходить чек от момента создания до "смерти".
3. Определите события, которые вызывают переход между состояниями.
4. Нарисуйте диаграмму состояний.
5. Добавьте условия (в квадратных скобках), где это необходимо.

Часть 3. Анализ и выводы

1. Сравните текстовое описание Use Case (ЛР №7) и диаграмму последовательности. Что нового дает диаграмма?
2. Посмотрите на диаграмму состояний. Выявите, какие состояния и переходы не были очевидны из простого списка требований.
3. Ответьте на вопрос: "Зачем нужно моделировать состояния объекта до того, как программист сел писать код?"

Контрольные вопросы

1. Чем диаграмма последовательности отличается от диаграммы прецедентов?
2. Какие элементы используются в диаграмме последовательности? Что такое "линия жизни" и "фокус управления"?
3. Для каких объектов обычно строят диаграмму состояний?
4. Что такое "состояние" объекта? Приведите примеры состояний для объекта "Заказ" в интернет-магазине.
5. Как на диаграмме состояний обозначаются начало и конец жизненного цикла?

6. Какую диаграмму вы будете строить, если нужно показать программисту точную последовательность вызовов методов? Почему?

Лабораторная работа №9
«Проектирование объектной модели (Диаграммы классов)»

Теоретическая часть

Что такое объектная модель?

После того как мы описали поведение системы (как она работает), нужно описать ее структуру (из чего она состоит). **Объектная модель** – это представление системы в виде набора классов, их атрибутов, методов и взаимосвязей.

Диаграмма классов (Class Diagram) – основной инструмент объектного моделирования в UML. Это "чертеж" системы, показывающий разработчикам, какие сущности нужно реализовать в коде и как они связаны друг с другом.

Основные элементы диаграммы классов

Класс (Class)

Изображается в виде прямоугольника, разделенного на три секции (таблица 15):

Таблица 15

Секция	Содержит
Верхняя	Имя класса
Средняя	Атрибуты (свойства)
Нижняя	Методы (операции)

Модификаторы доступа (видимость):

- + public (доступен всем);
- - private (доступен только внутри класса);
- # protected (доступен классу и наследникам);
- ~ package (доступен внутри пакета).

Отношения между классами представлены в таблице 16.

Таблица 16

Тип отношения	Обозначение	Описание
Ассоциация (Association)	Простая линия	Классы знают друг о друге, могут быть связаны
Направленная ассоциация	Стрелка	Один класс знает о другом, но не наоборот
Агрегация (Aggregation)	Ромб на конце (пустой)	Часть и целое. Часть может существовать отдельно от целого (слабая связь)
Композиция (Composition)	Ромб на конце (закрашенный)	Часть и целое. Часть НЕ может существовать отдельно от целого (сильная связь). Если целое удаляется, удаляются и части
Наследование (Generalization)	Стрелка с пустым треугольником	Класс-родитель и класс-потомок. Потомок наследует все свойства и методы родителя
Реализация (Realization)	Пунктирная стрелка с пустым треугольником	Класс реализует интерфейс
Зависимость (Dependency)	Пунктирная стрелка	Класс временно использует другой (например, в параметре метода)

Множественность (Multiplicity)

Показывает, сколько экземпляров одного класса может быть связано с одним экземпляром другого класса. Указывается на концах линий (таблица 17).

Таблица 17

Обозначение	Значение
1	Ровно один
0..1	Ноль или один
*	Много (ноль и более)
1..*	Один и более
0..*	Ноль и более

Зачем нужна диаграмма классов?

1. Для разработчиков: Это готовая схема базы данных и структуры классов в коде (ORM-модели).
2. Для аналитиков: Позволяет проверить, все ли сущности учтены и правильно ли они связаны.
3. Для документирования: Фиксирует архитектуру системы.

Задание

Цель работы: Построить объектную модель системы книжного магазина в виде диаграммы классов UML.

Часть 1. Выявление классов

1. Проанализируйте все материалы предыдущих лабораторных работ (требования, сценарии Use Case, описание процессов).
2. Составьте список потенциальных классов (сущностей) системы. Подумайте, какие объекты существуют в предметной области книжного магазина.
3. Отберите 5–7 самых важных классов для первой версии системы (MVP).

Часть 2. Определение атрибутов и методов

1. Для каждого класса определите его атрибуты (свойства). Подумайте, какими данными должен обладать объект.
2. Для каждого класса определите основные методы (что объект умеет делать).
3. Не забудьте указать типы данных для атрибутов (String, Integer, Double, Date, Boolean) и модификаторы доступа (обычно атрибуты - private, методы + public).

Часть 3. Построение диаграммы классов

1. Нарисуйте все выбранные классы в виде прямоугольников с тремя секциями.

2. Заполните секции (имя, атрибуты, методы).
3. Определите и нарисуйте связи между классами. Используйте правильные типы отношений и множественность.
4. Добавьте множественность на концах связей.
5. Проверьте, чтобы все ключевые сущности из Use Case были отражены в диаграмме.

Часть 4. Анализ и выводы

1. Объясните, почему вы выбрали именно такие типы связей (почему композиция, а не агрегация для Чек-ПозицияЧека)?
2. Подумайте, какие классы будут соответствовать таблицам в базе данных?
3. Как диаграмма классов помогает разработчикам начать писать код?

Контрольные вопросы

1. Что показывает диаграмма классов? Из каких основных частей состоит класс на UML-диаграмме?
2. Чем отличается агрегация от композиции? Приведите примеры из реальной жизни.
3. Что такое множественность (multiplicity)? Какие значения она может принимать?
4. Как на диаграмме классов обозначается наследование?
5. Зачем нужно указывать модификаторы доступа (+, -, #) на диаграмме классов?
6. Предположим, у вас есть классы "Студент" и "Группа". Какая между ними связь? Какая множественность?

Лабораторная работа №10
«Проектирование и визуализация модели данных»

Теоретическая часть

От объектной модели к модели данных

В Лабораторной работе №9 мы создали диаграмму классов, которая описывает объекты системы с точки зрения программирования. Теперь нам нужно спроектировать, как эти данные будут храниться – то есть создать модель данных (физическую или логическую модель базы данных).

Модель данных (Data Model) – это абстрактное представление того, какие данные хранятся в системе, как они организованы и как связаны между собой.

Уровни моделей данных приведены в таблице 18.

Таблица 18

Уровень	Назначение	Пример
Концептуальная модель	Высокоуровневое описание сущностей и связей (без атрибутов и типов). Для обсуждения с заказчиком.	"Книги", "Авторы", "Продажи"
Логическая модель	Детальное описание сущностей, атрибутов, типов данных, связей. Независит от конкретной СУБД.	Таблицы, колонки, первичные ключи, внешние ключи (в нотации IDEF1X)
Физическая модель	Реализация для конкретной СУБД (MySQL, PostgreSQL, Oracle). Учитывает индексы, типы данных конкретной БД, представления.	CREATE TABLE books (id INT PRIMARY KEY, title VARCHAR(255))

Основные понятия реляционных баз данных представлены в таблице 19.

Таблица 19

Термин	Описание
Таблица (Relation)	Хранит данные об одной сущности (например, "Книги")
Кортеж (Row, Запись)	Одна строка таблицы (одна конкретная книга)
Атрибут (Column, Поле)	Свойство сущности (название книги, цена)
Первичный ключ (Primary Key, PK)	Уникальный идентификатор записи (например, ID книги). Не может быть NULL
Внешний ключ (Foreign Key, FK)	Ссылка на первичный ключ другой таблицы (связь между таблицами).
Индекс (Index)	Механизм ускорения поиска (как оглавление в книге)

Нормализация данных

Нормализация – это процесс организации данных для устранения избыточности и аномалий (когда при изменении одной записи приходится менять много мест).

Основные нормальные формы:

1. 1НФ (Первая нормальная форма): Все атрибуты атомарны (неделимы). В ячейке не может быть списка значений.
2. 2НФ (Вторая нормальная форма): Находится в 1НФ + все неключевые атрибуты зависят от полного первичного ключа (актуально для составных ключей).
3. 3НФ (Третья нормальная форма): Находится в 2НФ + нет транзитивных зависимостей (неключевые атрибуты не зависят от других неключевых атрибутов).

Нотации моделирования данных

Для визуализации моделей используются специальные нотации (способы рисования):

1. IDEF1X: Стандарт для реляционных моделей. Таблицы рисуются прямоугольниками, связи – линиями с точками на концах.

2. IE (Information Engineering): Нотация с "вороньими лапками" (crow's foot) для обозначения множественности.
3. UML: Можно использовать и UML, но для баз данных привычнее IDEF1X или IE.

Типы связей между таблицами:

1. Один-ко-многим (1:N): Одна запись в таблице А соответствует многим записям в таблице Б. Пример: Один автор может написать много книг.
2. Многие-ко-многим (M:N): Много записей в А соответствует многим записям в Б. Реализуется через промежуточную (связующую) таблицу. Пример: Книги и Авторы (книгу могут написать несколько авторов, автор может написать несколько книг).
3. Один-к-одному (1:1): Одной записи в А соответствует одна запись в Б. Используется редко, обычно для разделения часто и редко используемых данных.

ER-диаграмма (Entity-Relationship Diagram)

Это итоговая диаграмма "сущность-связь", которая показывает структуру базы данных. Она является результатом проектирования и основой для создания SQL-скриптов.

Задание

Цель работы: Разработать логическую модель данных для системы книжного магазина и представить ее в виде ER-диаграммы.

Часть 1. Трансформация классов в таблицы

1. Возьмите диаграмму классов из ЛР №9.
2. Преобразуйте каждый класс в таблицу базы данных. Заполните таблицу соответствия (таблица 20):

Таблица 20

Класс (объектная модель)	Таблица (модель данных)	Примечание
Книга	Books	Основная таблица товаров
Автор	Authors	Справочник авторов
Чек	Receipts	Заголовок документа продажи
ПозицияЧека	ReceiptItems	Строки документа (табличная часть)
Продавец	Employees (или Users)	Сотрудники
...	...	...

Часть 2. Определение атрибутов и типов данных

1. Для каждой таблицы определите набор полей (атрибутов), взяв за основу атрибуты классов из ЛР9.
2. Добавьте необходимые служебные поля.
3. Для каждого поля укажите: Тип данных (INT, VARCHAR(n), DECIMAL, DATE, BOOLEAN, TEXT), Ограничения (NOT NULL, UNIQUE, DEFAULT).
4. Заполните таблицу спецификации для каждой таблицы (пример для Books таблица 21):

Таблица 21

Поле	Тип	Ограничения	Описание
id	INT	PRIMARY KEY, AUTO_INCREMENT	Уникальный идентифика-тор книги
isbn	VARCHAR(20)	UNIQUE	Международ-ный стандартный книжный номер
title	VARCHAR(255)	NOT NULL	Название книги
price	DECIMAL(10, 2)	NOT NULL, DEFAULT 0.00	Цена в рублях

Поле	Тип	Ограничения	Описание
published_year	INT		Год издания
quantity_in_stock	INT	NOT NULL, DEFAULT 0	Текущий остаток на складе
created_at	DATETIME	NOT NULL	Дата добавления в каталог

Часть 3. Определение связей

1. Для каждой пары связанных таблиц определите тип связи (1:N, M:N, 1:1).
2. Определите, где должен находиться внешний ключ.

Часть 4. Построение ER-диаграммы

1. Используя любой инструмент (Draw.io, Lucidchart, Visio, DataGrip, MySQL Workbench), нарисуйте ER-диаграмму.
2. Изобразите все таблицы в виде прямоугольников с перечнем полей.
3. Выделите первичные ключи (жирным или значком ключа) и внешние ключи (курсивом или значком ключа).
4. Нарисуйте связи между таблицами линиями. На концах линий укажите множественность (используйте нотацию "воронья лапка" или цифры 1, *).
5. Подпишите связи, если это необходимо для понимания.

Часть 5. Проверка нормализации

1. Проверьте ваши таблицы на соответствие 3-й нормальной форме (3НФ).
2. Найдите хотя бы одно нарушение нормализации в первоначальном проекте (если оно есть) и исправьте его.
3. Опишите, какие аномалии могут возникнуть, если оставить данные ненормализованными.

Контрольные вопросы

1. Чем логическая модель данных отличается от физической?
2. Что такое первичный ключ (PK) и внешний ключ (FK)? Для чего они нужны?

3. Какие типы связей между таблицами существуют? Как реализуется связь "многие-ко-многим" в реляционной базе данных?
4. Что такое нормализация данных? Назовите первые три нормальные формы.
5. Почему плохо хранить список авторов в одной ячейке таблицы "Книги" (через запятую)? Какие проблемы это вызовет?
6. Для чего нужно указывать типы данных и ограничения (NOT NULL, UNIQUE) при проектировании таблиц?

Лабораторная работа №11
«Нормализация базы данных»

Теоретическая часть

Что такое нормализация?

Нормализация – это процесс организации данных в реляционной базе данных, направленный на устранение избыточности (дублирования) и обеспечение целостности данных при их изменении. Это пошаговый процесс приведения структуры базы данных к определенным нормальным формам.

Зачем нужна нормализация?

Представьте таблицу, где в одной строке хранится и книга, и ее автор, и издательство. Если издательство сменит адрес, вам придется обновлять тысячи строк. Если автор напишет новую книгу, вы добавите новую строку и снова продублируете адрес издательства. Нормализация решает эти проблемы.

Проблемы ненормализованной БД (Аномалии):

1. Аномалии вставки (Insert Anomaly): Нельзя добавить данные, если нет полной информации. Пример: Нельзя добавить нового автора, если он еще не написал книгу.
2. Аномалии обновления (Update Anomaly): При изменении данных приходится менять много строк. Пример: При смене адреса издательства нужно обновить все книги этого издательства.
3. Аномалии удаления (Delete Anomaly): При удалении данных удаляется и полезная информация. Пример: При удалении последней книги издательства теряется информация об издательстве.

Нормальные формы (ключевые понятия):

1. Первая нормальная форма (1НФ). Таблица находится в 1НФ, если все атрибуты (поля) содержат атомарные (неделимые) значения и нет повторяющихся групп.
2. Вторая нормальная форма (2НФ). Таблица находится в 2НФ, если: Она находится в 1НФ. Каждый неключевой атрибут полностью зависит от первичного ключа (для таблиц с составным ключом).
3. Третья нормальная форма (3НФ). Таблица находится в 3НФ, если: Она находится в 2НФ. Отсутствуют транзитивные

зависимости (неключевые атрибуты не зависят от других неключевых атрибутов).

4. Нормальная форма Бойса-Кодда (НФБК). Более строгая версия 3НФ. Таблица в НФБК, если любой детерминант (атрибут, от которого зависят другие) является потенциальным ключом.
5. Четвертая нормальная форма (4НФ). Устраняет многозначные зависимости (когда один атрибут определяет несколько независимых множеств значений).

Денормализация

Иногда, для повышения производительности (ускорения запросов), разработчики сознательно отказываются от полной нормализации и добавляют избыточность (дублируют данные). Это называется денормализацией. Но это исключение, а не правило.

Процесс нормализации на практике:

1. Начинаем с одной большой таблицы (универсального отношения).
2. Последовательно приводим к 1НФ, 2НФ, 3НФ.
3. В большинстве бизнес-приложений достаточно 3НФ.

Задание

Цель работы: Выполнить нормализацию базы данных книжного магазина от ненормализованной формы до 3-й нормальной формы (3НФ).

Часть 1. Исходная ненормализованная таблица

Представьте, что сначала заказчик предоставил данные в виде одной таблицы (например, Excel-файла), с которой работает магазин: Таблица: Книжный_Учет (таблица 22).

Таблица 22

Номер_чека	Дата	Продавец	Книги	Сумма_чека	Издательство	Адрес_издательства	Авторы
1	01.03.2024	Иванова Мария	Война и мир (2 шт.); Анна Каренина (1 шт.)	2500.00	Эксмо	Москва, ул. Правды, 1	Толстой Л.Н.
2	01.03.2024	Петров Алексей	Преступление и наказание (1 шт.)	800.00	АСТ	Москва, ул. Звездная, 5	Достоевский Ф.М.
3	02.03.2024	Иванова Мария	Война и мир (1 шт.); Идиот (1 шт.)	1800.00	Эксмо; АСТ	Москва, ул. Правды, 1; Москва, ул. Звездная, 5	Толстой Л.Н.; Достоевский Ф.М.
4	02.03.2024	Сидорова Елена	Мастер и Маргарита (2 шт.)	1600.00	Азбука	СПб, ул. Ленина, 10	Булгаков М.А.

Проблемы текущей структуры (очевидные нарушения):

1. Поле Книги содержит несколько значений (названия и количество) через точку с запятой.
2. Поле Авторы содержит нескольких авторов.
3. Поле Издательство содержит несколько издательств (в строке 3).

4. Дублируется информация об издательствах (адреса повторяются).

Часть 2. Приведение к 1НФ (Первая нормальная форма)

1. Устраните повторяющиеся группы. Сделайте так, чтобы в каждой ячейке было только одно значение.
2. Разбейте сложные поля на атомарные компоненты:
   - Разделите поле Книги на отдельные позиции (одна строка = одна книга в чеке).
   - Выделите количество в отдельное поле.
   - Разделите поле Авторы (пока оставьте как есть, но подготовьтесь к дальнейшей нормализации (одна ячейка – один автор)).
   - При наличии нескольких значений в поле Издательство разделить их на отдельные записи.

Часть 3. Приведение к 2НФ (Вторая нормальная форма)

1. Определите составной первичный ключ для таблицы в 1НФ. Что уникально идентифицирует строку? (Возможно, Номер_чека + Название_книги).
2. Выявите атрибуты, которые зависят только от части ключа.
3. Разделите таблицу на несколько, устраняя частичные зависимости.
4. Заполните полученные таблицы данными из исходной.

Часть 4. Приведение к 3НФ (Третья нормальная форма)

1. Выявите транзитивные зависимости (когда одно неключевое поле зависит от другого неключевого).
2. Устраните транзитивные зависимости и атомизируйте данные.
3. Определите первичные ключи для всех новых таблиц (обычно искусственные ID).

Часть 5. Итоговая структура (3НФ)

Опишите итоговую структуру базы данных в виде перечня таблиц и полей (или нарисуйте небольшую ER-диаграмму).

Часть 6. Анализ результатов

1. Сравните исходную таблицу и итоговую структуру. Какие проблемы решены?
2. Опишите, как теперь будут выполняться операции:
   - Смена адреса издательства (сколько таблиц нужно обновить?)
   - Добавление нового автора без книги

- Удаление чека (потеряется ли информация об авторе?)

3. Есть ли в вашей структуре потенциальные кандидаты на денормализацию? (Например, сумма чека дублируется – это допустимая денормализация для скорости).

Контрольные вопросы

1. Что такое нормализация базы данных? Какие проблемы она решает?
2. Перечислите аномалии, возникающие в ненормализованных базах данных.
3. Опишите требования к первой нормальной форме (1НФ). Приведите пример нарушения.
4. Опишите требования ко второй нормальной форме (2НФ). Чем она отличается от 1НФ?
5. Опишите требования к третьей нормальной форме (3НФ). Что такое транзитивная зависимость?
6. Достаточно ли обычно 3НФ для большинства бизнес-приложений? Почему?
7. Что такое денормализация и когда она применяется?

Лабораторная работа №12
«Проектирование пользовательского интерфейса»

Теоретическая часть

Что такое пользовательский интерфейс (UI)?

Пользовательский интерфейс (User Interface, UI) – это совокупность средств и методов, с помощью которых пользователь взаимодействует с системой. Это то, что пользователь видит на экране, и то, как он управляет программой.

Что такое пользовательский опыт (UX)?

Пользовательский опыт (User Experience, UX) – это ощущения и впечатления, которые возникают у пользователя при взаимодействии с интерфейсом. Хороший UX означает, что интерфейс интуитивно понятен, эффективен и доставляет удовольствие от работы.

Простая формула: *UI – это как выглядит, UX – это как работает.*

Зачем проектировать интерфейс?

1. Снижение ошибок: Понятный интерфейс уменьшает вероятность того, что пользователь нажмет не ту кнопку.
2. Скорость обучения: Чем проще интерфейс, тем быстрее новый сотрудник начнет работать.
3. Удовлетворенность: Пользователи не любят сложные и нелогичные программы.
4. Экономия: Исправлять ошибки проектирования интерфейса после написания кода очень дорого.

Этапы проектирования интерфейса:

1. Анализ пользователей. Используем персоны из ЛР №6. Понимаем, кто будет работать с системой (продавец с низким уровнем ПК или владелец с планшетом).
2. Создание карты сайта (Site Map). Показывает структуру экранов и навигацию между ними. Какие разделы есть в системе и как между ними переходить.
3. Варианты использования (Use Case). Берем сценарии из ЛР №7. Понимаем, какие задачи пользователь решает на каждом экране.
4. Создание прототипов

- Низкая детализация (Low-fidelity, Lo-Fi): Бумажные наброски, схемы (wireframes). Показывают только структуру и расположение элементов, без цветов и деталей. Быстро и дешево.
- Высокая детализация (High-fidelity, Hi-Fi): Цветные макеты, приближенные к реальному виду. Показывают шрифты, иконки, отступы.

5. Юзабилити-тестирование. Проверка прототипов на реальных пользователях (или коллегах). Наблюдаем, как они выполняют задачи, и исправляем проблемы.

Инструменты прототипирования:

1. Бумага и карандаш (самый быстрый способ);
2. Figma (профессиональный инструмент);
3. Draw.io / Lucidchart (для простых схем);
4. Balsamiq (специализированный инструмент для Lo-Fi прототипов).

Задание

Цель работы: Разработать прототипы пользовательского интерфейса для ключевых сценариев работы в системе книжного магазина.

Часть 1. Анализ и подготовка

1. Вспомните персоны пользователей из ЛР №6 (Продавец Елена и Владелец Иван Петрович).
2. Выпишите ключевые задачи, которые эти пользователи решают в системе (на основе Use Case из ЛР №7).
3. Определите основные экраны, которые понадобятся в системе.

Часть 2. Создание карты навигации (Site Map)

1. Нарисуйте схему, показывающую, как экраны связаны между собой.
2. Используйте прямоугольники для экранов и стрелки для переходов.

Часть 3. Создание прототипа низкой детализации (Lo-Fi Wireframes)

1. Выберите два ключевых экрана.
2. Для каждого экрана нарисуйте прототип низкой детализации. Можно на бумаге (сфотографировать и вставить в отчет) или в любом редакторе (Draw.io, Figma).

3. На прототипе обязательно покажите:
    - Расположение основных элементов: Заголовки, поля ввода, кнопки, списки.
    - Подписи к элементам: Что это за поле?
    - Основной сценарий: Стрелками или цифрами покажите, в каком порядке пользователь взаимодействует с экраном.

Контрольные вопросы

1. Чем отличаются UI и UX? Приведите примеры.
2. Что такое прототип низкой детализации (Lo-Fi) и для чего он нужен?
3. Перечислите не менее 5 эвристик Якоба Нильсена (принципов юзабилити).
4. Почему важно проектировать интерфейс до начала программирования?
5. Что такое карта сайта (Site Map) и как она помогает в проектировании?
6. Какие элементы обязательно должны быть на экране продажи для кассира? Почему?

Лабораторная работа №13
«Проработка шаблонов и спецификаций»

Теоретическая часть

Что такое шаблоны проектирования?

Шаблон проектирования (Design Pattern) – это типовое решение часто встречающейся проблемы при проектировании программного обеспечения. Это не готовый код, а описание того, как решить проблему, которое можно использовать в разных ситуациях.

Аналогия: Шаблон – это не готовый дом, а проверенный архитектурный план, по которому можно построить много разных домов.

Зачем нужны шаблоны?

1. Использование опыта: "Не нужно изобретать велосипед – используют проверенные решения".
2. Единый язык: Разработчики говорят "Фабрика" или "Синглтон", и все понимают, о чем речь.
3. Снижение сложности: Правильные шаблоны упрощают архитектуру и делают код понятнее.
4. Облегчение поддержки: Код, построенный на шаблонах, легче модифицировать.

Что такое спецификация?

Спецификация (Specification) в контексте проектирования ИС – это детальное описание того, как должен работать конкретный компонент или функция. Это мостик между анализом (ЧТО делать) и разработкой (КАК делать).

Виды спецификаций:

1. Функциональная спецификация: Описание логики работы, входных и выходных данных, алгоритмов.
2. Техническая спецификация: Описание архитектуры, технологий, интерфейсов взаимодействия.
3. Спецификация экрана: Описание элементов пользовательского интерфейса (что происходит при нажатии каждой кнопки).
4. Спецификация API: Описание методов, которые предоставляет система для интеграции.

Основные группы шаблонов проектирования (GoF):

1. Порождающие шаблоны (Creational). Отвечают за создание объектов, делая систему независимой от способа создания объектов.
2. Структурные шаблоны (Structural). Определяют различные сложные структуры классов, упрощая отношения между объектами.
3. Поведенческие шаблоны (Behavioral). Определяют алгоритмы и взаимодействия между объектами.

Задание

Цель работы: Применить шаблоны проектирования к системе книжного магазина и разработать детальную спецификацию ключевой функции.

Часть 1. Идентификация и применение шаблонов

1. Проанализируйте архитектуру вашей системы (по диаграмме классов из ЛР №9).
2. Определите, какие шаблоны проектирования могут быть полезны в следующих ситуациях (таблица 23):

Таблица 23

Ситуация	Какой шаблон применить?	Почему?
В системе должен быть только один экземпляр класса "Касса" (или "Магазин")	**Singleton**	Чтобы не создать случайно вторую кассу с другими настройками
Система должна уведомлять владельца, когда остаток книги становится меньше 3	**Observer**	Склад (наблюдаемый объект) уведомляет подписчиков (владельца) об изменении
Нужно подключаться к разным внешним системам (1С старая и 1С новая) с разными API	**Adapter**	Адаптеры приведут разные интерфейсы к единому виду

Ситуация	Какой шаблон применить?	Почему?
Расчет скидки может меняться или добавляться новая стратегия (сезонная скидка, скидка постоянному клиенту)	**Strategy**	Инкапсулируем алгоритмы скидок и делаем их взаимозаменяемыми

3. Заполните таблицу, предложив свои варианты применения шаблонов для книжного магазина.

Часть 2. Разработка детальной спецификации

1. Выберите одну ключевую функцию системы (например, "Оформление продажи" или "Возврат товара").
2. Разработайте для нее полную спецификацию по шаблону, приведенному в теоретической части.

Часть 3. Спецификация элемента интерфейса (детализация)

1. Выберите один сложный элемент интерфейса из вашего прототипа (например, кнопка "Оплатить" или таблица позиций чека).
2. Составьте для него спецификацию поведения.

Часть 4. Шаблон проектирования в действии

1. Выберите один из шаблонов, которые вы определили в Части 1.
2. Опишите, как конкретно этот шаблон будет реализован в коде вашего проекта.
3. Нарисуйте небольшую диаграмму классов, демонстрирующую применение шаблона.

Контрольные вопросы

1. Что такое шаблон проектирования? Для чего они нужны?
2. Назовите три основные группы шаблонов GoF ("банды четырех") и их назначение.
3. Приведите пример порождающего, структурного и поведенческого шаблона.
4. Что такое спецификация? Чем она отличается от требования?

5. Какие разделы должна содержать хорошая спецификация функции?
6. Почему важно описывать альтернативные потоки и бизнес-правила в спецификации?

Лабораторная работа №14
«Сценарное моделирование и анализ»

Теоретическая часть

Что такое сценарное моделирование?

Сценарное моделирование (Scenario Modeling) – это метод анализа поведения системы, при котором мы рассматриваем различные варианты развития событий (сценарии) использования системы. Это не просто описание того, как система работает в идеальном случае, а исследование всех возможных путей, включая ошибки, исключения и редко встречающиеся ситуации.

Цели сценарного моделирования:

1. Выявление полноты требований: Не упустили ли мы важные ситуации?
2. Проверка логики: Корректно ли система реагирует на нестандартные ситуации?
3. Подготовка к тестированию: Сценарии становятся основой для тест-кейсов.
4. Согласование с заказчиком: Показать заказчику не только "как хорошо работает система", но и "что будет, если пойдет не так".

Виды сценариев приведены в таблице 24.

Таблица 24

Тип сценария	Описание	Пример
Основной сценарий (Happy Path)	Идеальный путь, когда все идет по плану, ошибок нет	Покупатель выбрал книгу, оплатил, получил чек
Альтернативный сценарий (Alternative Flow)	Другие способы достижения цели, но в рамках успеха	Оплата картой вместо наличных
Сценарий исключения (Exception Flow)	Ситуации, когда что-то пошло не так	Книги нет в наличии, терминал не работает, покупатель передумал
Сценарий ошибки (Error Flow)	Критические сбои, ведущие к невозможности завершить операцию	Сервер упал, база данных недоступна

Методы сценарного моделирования:

1. POS (Потоковый анализ). Анализ потоков данных и управления в системе. Используются диаграммы активности (activity diagrams) или блок-схемы.
2. Таблица решений (Decision Table). Используется, когда результат зависит от комбинации нескольких условий. Компактно представляет логику выбора.
3. Конечные автоматы (State Machines). Используются, когда поведение системы зависит от состояния, в котором она находится.
4. Диаграммы последовательности (Sequence Diagrams). Показывают взаимодействие объектов во времени для конкретного сценария.

Задание

Цель работы: Разработать полный набор сценариев для ключевого прецедента системы и провести анализ покрытия требований.

Часть 1. Выявление сценариев

1. Возьмите прецедент "Продажа книги" (или другой ключевой прецедент из ЛР №7).
2. Используя метод мозгового штурма, придумайте минимум 7 различных сценариев использования этого прецедента.
3. Сгруппируйте их по типам

Часть 2. Детальное описание сценария

1. Выберите один сложный сценарий из списка (желательно не Happy Path, например, сценарий исключения).
2. Опишите его детально по шаблону расширенного Use Case.

Часть 3. Построение диаграммы деятельности (Activity Diagram)

Для того же прецедента "Продажа книги" постройте диаграмму деятельности (activity diagram), которая покажет все возможные пути (ветвления).

Контрольные вопросы

1. Что такое сценарное моделирование? Зачем оно нужно?
2. Чем отличается основной сценарий от альтернативного и сценария исключения?

3. Что такое матрица покрытия требований и как она строится?
4. Как сценарное моделирование связано с тестированием программного обеспечения?
5. Какие методы сценарного моделирования вы знаете?
6. Почему важно моделировать не только успешные, но и ошибочные сценарии?

Лабораторная работа №15
«Функциональное проектирование системы»

Теоретическая часть

Что такое функциональное проектирование?

Функциональное проектирование – это этап, на котором определяется, КАК ИМЕННО система будет реализовывать требуемую функциональность. Если предыдущие этапы отвечали на вопрос "ЧТО система должна делать?", то функциональное проектирование отвечает на вопрос "КАК это будет работать с точки зрения функций и компонентов?".

Это мост между анализом требований и технической реализацией (программированием).

Основные задачи функционального проектирования:

1. Декомпозиция системы на функциональные модули (разбиение большой задачи на маленькие подзадачи).
2. Определение потоков данных между модулями.
3. Спецификация функций каждого модуля.
4. Определение интерфейсов взаимодействия между модулями.
5. Проектирование архитектуры (высокоуровневая структура системы).

Функциональная архитектура

Функциональная архитектура показывает, из каких крупных блоков (модулей, компонентов) состоит система и как они связаны.

Диаграмма компонентов (Component Diagram)

В UML для описания физической структуры системы используется диаграмма компонентов.

Она показывает:

1. Компоненты (физические модули, библиотеки, файлы).
2. Интерфейсы, которые компоненты предоставляют и требуют.
3. Зависимости между компонентами.

Элементы диаграммы и их обозначения с описанием приведены в таблице 25.

Таблица 25

Элемент	Обозначение	Описание
Компонент	Прямоугольник с значком компонента (или <<component>>)	Модуль системы
Интерфейс	Кружок (или прямоугольник с <<interface>>)	Набор методов, которые компонент предоставляет другим
Зависимость	Пунктирная стрелка	Один компонент использует другой
Порт	Квадратик на границе компонента	Точка взаимодействия

Потоки данных (Data Flow Diagrams - DFD)

DFD – это классический метод функционального моделирования, показывающий, как данные перемещаются между процессами (функциями), внешними сущностями и хранилищами данных.

Traceability Matrix (Матрица трассировки)

Важный инструмент функционального проектирования, который связывает требования с функциональными модулями и проверяет, что все требования реализованы.

Задание

Цель работы: Выполнить функциональное проектирование системы книжного магазина: определить модули, их взаимодействие, построить диаграмму компонентов и контекстную DFD.

Часть 1. Функциональная декомпозиция

1. Проанализируйте все требования и сценарии, разработанные в предыдущих лабораторных работах.
2. Разбейте систему на крупные функциональные модули. Обоснуйте, почему вы выбрали именно такое разбиение.
3. Для каждого модуля заполните краткую спецификацию.

Часть 2. Построение диаграммы компонентов

Используя выбранные модули, постройте диаграмму компонентов UML.

Часть 3. Построение контекстной диаграммы DFD (уровень 0)

1. Постройте контекстную диаграмму (Data Flow Diagram уровня 0), которая показывает систему как единое целое и ее взаимодействие с внешними сущностями.
2. Определите внешние сущности.
3. Нарисуйте один центральный процесс "Книжный магазин" (круг).
4. Покажите потоки данных между внешними сущностями и системой.

Часть 4. Матрица трассировки (Traceability Matrix)

1. Составьте матрицу, связывающую функциональные требования (из ЛР №2) с модулями, которые вы определили.
2. Проанализируйте матрицу.

Контрольные вопросы

1. Что такое функциональное проектирование? Чем оно отличается от анализа требований?
2. Что такое функциональная декомпозиция? Зачем она нужна?
3. Какие элементы показывают на диаграмме компонентов UML?
4. Что такое контекстная диаграмма DFD и для чего она используется?
5. Что такое матрица трассировки (traceability matrix)? Какую проблему она решает?
6. Чем отличаются внешние сущности от процессов в DFD?
7. Почему важно определять зависимости между модулями на этапе проектирования?

Лабораторная работа №16
«Трассировка требований»

Теоретическая часть

Что такое трассировка требований?

Трассировка требований (Requirements Traceability) – это процесс установления и документирования связей между требованиями и другими артефактами проекта на протяжении всего жизненного цикла разработки. Это "прослеживаемость" требований: от источника (почему появилось) до реализации (как реализовано) и проверки (как протестировано).

Зачем нужна трассировка?

1. Полнота реализации: Гарантия, что все требования были реализованы (нет "забытых" требований).
2. Анализ влияния (Impact Analysis): Понимание, какие компоненты системы затронет изменение конкретного требования.
3. Верификация и валидация: Подтверждение, что система делает то, что нужно (валидация), и делает это правильно (верификация).
4. Поддержка тестирования: Понимание, какие тесты проверяют какие требования.
5. Управление изменениями: При изменении требования легко найти все связанные артефакты (код, тесты, документацию).

Направления трассировки (таблица 26):

Таблица 26

Направление	Описание	Пример
Прямая трассировка (Forward)	От требований к артефактам реализации (проектирование, код, тесты)	Требование FR-01 реализовано в модуле "Продажи" и проверяется тестом TC-01
Обратная трассировка (Backward)	От артефактов к источникам требований (почему это сделано?)	Модуль "Продажи" существует, потому что есть требование FR-01 от заказчика
Горизонтальная трассировка	Связи между требованиями разных уровней (бизнес-требования -> функциональные требования)	Бизнес-цель "Увеличить скорость обслуживания" -> функциональное требование "Сканирование штрих-кода"

Типы связей в трассировке (таблица 27):

Таблица 27

Тип связи	Обозначение	Описание
Порождает (Derives from)	<-	Один артефакт является источником для другого
Реализует (Implements)	->	Требование реализовано в компоненте
Проверяет (Verifies)	->	Тест-кейс проверяет требование
Зависит от (Depends on)	<->	Один артефакт зависит от другого
Блокируется (Blocks)	-/	Один артефакт блокирует выполнение другого

Матрица трассировки требований (Requirements Traceability Matrix - RTM)

Основной инструмент трассировки – это матрица, которая связывает требования с другими элементами проекта.

Простая матрица (требования -> компоненты) (таблица 28):

Таблица 28

ID требования	Описание	Версия	Модуль "Продажи"	Модуль "Каталог"	Тест-кейс	Статус
FR-01	Продажа книги	1.0	+		TC-01	Реализовано
FR-02	Поиск книги	1.0		+	TC-02	В разработке

Уровни трассировки:

1. Трассировка от бизнес-целей к требованиям.
2. Трассировка от требований к архитектуре.
3. Трассировка от требований к тестам.

Инструменты для трассировки:

1. Excel / Google Sheets (для небольших проектов).
2. JIRA / YouTrack (с плагинами для трассировки).
3. IBM DOORS / Jama Connect (специализированные инструменты управления требованиями).
4. Enterprise Architect (моделирование с трассировкой).

Процесс трассировки в проекте:

1. Идентификация: Каждому требованию присваивается уникальный ID.
2. Установление связей: В процессе проектирования и разработки создаются связи между требованиями и артефактами.
3. Поддержание актуальности: При изменении требований обновляются связи.
4. Анализ: Периодически проверяется полнота покрытия.

Задание

Цель работы: Разработать полную матрицу трассировки требований для системы книжного магазина, установив связи между требованиями, модулями, сценариями и тестами.

Часть 1. Подготовка исходных данных

1. Соберите все требования, разработанные в предыдущих лабораторных работах (из ЛР №2, ЛР №3). Убедитесь, что у каждого требования есть уникальный идентификатор.
2. Подготовьте список модулей из ЛР №15.
3. Подготовьте список ключевых сценариев из ЛР №14.

4. Подготовьте список тестов (придумайте хотя бы по одному тесту на каждое требование).

Часть 2. Построение базовой матрицы трассировки (Требования -> Модули)

1. Создайте матрицу в Excel или Google Sheets (или в виде таблицы в отчете).
2. По вертикали – требования (с ID и описанием).
3. По горизонтали – модули.
4. Отметьте знаком "+", в каком модуле реализуется каждое требование.

Часть 3. Расширенная матрица (Требования -> Сценарии -> Тесты)

Добавьте в матрицу связь с разработанными сценариями и тестами.

Часть 4. Анализ полноты покрытия

1. Проанализируйте полученную матрицу.
2. Выявите пробелы (gap-анализ).

Часть 5. Матрица трассировки "Источник -> Требование" (Обратная трассировка)

1. Вспомните или придумайте источники требований (из ЛР №1).
2. Постройте матрицу, показывающую, из какого источника появилось каждое требование.

Часть 6. Анализ влияния (Impact Analysis)

1. Выберите одно требование (например, FR-01 "Продажа книги").
2. Представьте, что заказчик попросил изменить это требование: теперь при продаже нужно указывать не только книгу, но и подарочную упаковку (опционально).
3. Используя матрицу трассировки, определите:
    - Какие модули будут затронуты изменением? (Продажи, Склад, возможно Отчеты)
    - Какие сценарии нужно будет изменить?
    - Какие тест-кейсы потребуют пересмотра?
4. Оформите результат анализа в виде краткого заключения.

Контрольные вопросы

1. Что такое трассировка требований? Для чего она нужна?
2. Какие направления трассировки существуют (прямая, обратная, горизонтальная)?
3. Что такое матрица трассировки требований (RTM)? Из каких элементов она состоит?
4. Как трассировка помогает при управлении изменениями (impact analysis)?
5. Почему важно связывать требования не только с модулями, но и с тестами?
6. Что такое "пробел" (gap) в трассировке и как его обнаружить?
7. Какие инструменты можно использовать для ведения трассировки?

Лабораторная работа №17
«Управление изменениями требований»

Теоретическая часть

Почему изменения неизбежны?

Изменения требований – это не отклонение от нормы, а естественная часть любого ИТ-проекта. Причины изменений могут быть разными:

1. Заказчик уточнил свои потребности ("я думал, это работает иначе").
2. Изменилось законодательство (новые требования к отчетности).
3. Появились новые технологии (можно сделать лучше / дешевле).
4. Конкуренты выпустили новый функционал.
5. Изменились бизнес-процессы компании.

Закон Легмана: *"Требования к программному обеспечению всегда изменяются в процессе разработки"*.

Что такое управление изменениями?

Управление изменениями (Change Management) – это формальный процесс, который позволяет контролировать поступление, анализ, одобрение и реализацию изменений в требованиях проекта.

Цели управления изменениями:

1. Контроль хаоса: Чтобы изменения не вносились бессистемно ("а давайте еще вот эту кнопочку").
2. Оценка влияния: Понимание, как изменение повлияет на сроки, бюджет и архитектуру.
3. Принятие обоснованных решений: Заказчик должен понимать цену изменения (в деньгах или времени).
4. Сохранение целостности: Чтобы изменение в одном месте не "сломало" другое.

Change Control Board (CCB) – Совет по управлению изменениями

Это группа людей, которые принимают решения об изменениях. В состав обычно входят:

1. Заказчик (или его представитель).
2. Менеджер проекта.
3. Системный аналитик.

4. Технический архитектор (иногда).

Анализ влияния (Impact Analysis)

Ключевой этап. При поступлении изменения нужно ответить на вопросы:

1. Что изменится? Какие требования, модули, документы нужно править?
2. Сколько это стоит? Трудозатраты в человеко-часах.
3. Какие риски? Что может сломаться?
4. Что не изменится? Какие части системы останутся нетронутыми.

Базовые версии (Baseline)

Baseline – это зафиксированное состояние требований на определенный момент времени (например, после утверждения технического задания (ТЗ)). Изменения после установки baseline должны проходить через формальный процесс.

Задание

Цель работы: Освоить процесс управления изменениями: от получения запроса до принятия решения и документирования.

Часть 1. Поступление запроса на изменение

Представьте, что проект книжного магазина находится в стадии разработки (после утверждения baseline 1.0). Поступают следующие запросы на изменения:

Запрос А (от владельца магазина):

"Я понял, что мне нужно видеть не только отчет по продажам, но и отчет по самым популярным книгам. Хочу, чтобы система автоматически строила рейтинг продаж и показывала топ-10 книг за неделю. Это очень важно для закупок!"

Запрос Б (от продавца Елены):

"Когда я пробиваю книгу, мне каждый раз нужно вручную выбирать способ оплаты. А у нас 90 % покупателей платят картой. Сделайте, чтобы по умолчанию стояла оплата картой, а наличные можно было выбрать отдельно. Это сэкономит мне кучу времени!"

Запрос В (от бухгалтера):

"Налоговая требует новый формат фискальных чеков с 1 числа следующего месяца. Наша старая форма больше не подходит. Нужно срочно менять формат вывода на печать."

Запрос Г (от системного администратора):

"Я заметил, что база данных иногда падает при пиковых нагрузках. Надо бы добавить кеширование популярных запросов, иначе в черную пятницу все ляжет. Это не функциональное требование, но без него система ненадежна."

Часть 2. Регистрация запросов

1. Зарегистрируйте каждый запрос в журнале изменений. Присвойте каждому уникальный ID (например, CR-01, CR-02, ...).
2. Заполните начальные поля (ID, дата, автор, краткое описание, категория, приоритет (по вашей оценке)).

Часть 3. Анализ влияния (Impact Analysis)

1. Для каждого запроса проведите анализ влияния. Используйте матрицу трассировки из ЛР №16, чтобы понять, какие модули и требования будут затронуты.
2. Заполните таблицу анализа.

Часть 4. Принятие решения

1. Представьте, что бюджет проекта ограничен, а до релиза осталось 2 недели. Вам как Change Control Board (вы и ваши коллеги) нужно принять решение по каждому запросу.
2. Заполните поле "Решение" для каждого запроса (Утвердить / Отклонить / Отложить до следующей версии). Обоснуйте решение.

Часть 5. Документирование изменений

1. Выберите один утвержденный запрос (например, CR-02 или CR-03).
2. Оформите полный документ Change Request.

Часть 6. Обновление документации

1. Представьте, что изменение CR-02 утверждено и реализовано.
2. Опишите, какие документы проекта нужно обновить.

Контрольные вопросы

1. Почему изменения требований неизбежны в процессе разработки?
2. Что такое Change Request (CR)? Из каких основных разделов он состоит?

3. Опишите процесс управления изменениями (этапы).
4. Что такое анализ влияния (Impact Analysis) и зачем он нужен?
5. Кто входит в Change Control Board (CCB) и какие решения они принимают?
6. Что такое baseline и как он связан с управлением изменениями?
7. Почему важно документально фиксировать все запросы на изменения, даже отклоненные?

Лабораторная работа №18
«Контроль качества требований»

Теоретическая часть

Что такое качество требований?

Качество требований – это степень, в которой требования правильно и полно описывают поведение системы, являясь при этом понятными, непротиворечивыми и пригодными для использования в разработке. Плохие требования – главная причина провала проектов.

Последствия плохих требований:

1. Переделки на поздних этапах (в 100 раз дороже, чем исправление на этапе анализа).
2. Непонимание между заказчиком и разработчиками.
3. Пропущенные функции или лишний функционал.
4. Срыв сроков и бюджета.

Методы контроля качества требований:

1. Инспекция требований (Requirements Inspection). Структурированная проверка требований группой экспертов.
2. Прототипирование. Создание прототипов для проверки понимания требований заказчиком.
3. Разработка тест-кейсов. Попытка написать тесты на основе требований выявляет неполноту и неоднозначность.
4. Атомарный анализ. Проверка, можно ли требование разбить на несколько.
5. Измерение качества (метрики).

Задание

Цель работы: Провести контроль качества набора требований, выявить дефекты и предложить исправления.

Часть 1. Анализ требований на наличие дефектов

1. Дан следующий набор требований к системе книжного магазина (некоторые из них содержат дефекты). Проанализируйте каждое требование (таблица 29).

Таблица 29

ID	Требование
R-01	Система должна быть удобной и быстрой
R-02	При продаже книги система должна проверять наличие на складе
R-03	Система должна формировать различные отчеты
R-04	Время отклика системы не должно превышать 2 секунд при работе 5 одновременных пользователей
R-05	При ошибке подключения к базе данных система должна показать сообщение
R-06	Система должна поддерживать не менее 1000 книг в каталоге
R-07	Интерфейс должен быть интуитивно понятным
R-08	Продавец может отменить чек до его закрытия
R-09	Система должна работать на любом оборудовании
R-10	Пароль пользователя должен храниться в безопасности

2. Для каждого требования определите, какие критерии качества нарушены.

Часть 2. Исправление требований

1. Выберите 5 требований с дефектами из Части 1.
2. Перепишите их в качественной форме, исправив все нарушения.

Часть 3. Чек-лист проверки

1. Составьте собственный чек-лист для проверки качества требований (минимум 10 пунктов).
2. Используйте критерии из теории и добавьте свои.

Часть 4. Инспекция требований (ролевая игра)

1. Разделитесь на группы (или представьте роли).
2. Возьмите 3 требования из предыдущих работ (или из Части 1) и проведите их инспекцию.
3. Заполните протокол инспекции.

Контрольные вопросы

1. Почему качество требований критически важно для успеха проекта?

2. Перечислите основные характеристики качественных требований (не менее 5).
3. Что такое "однозначность" требования? Приведите пример неоднозначного требования.
4. Чем "проверяемость" отличается от "выполнимости"?
5. Какие методы контроля качества требований существуют?
6. Что такое инспекция требований? Кто в ней участвует?
7. Какие метрики можно использовать для измерения качества требований?

Лабораторная работа №19
«Формирование структуры технического задания»

Теоретическая часть

Что такое техническое задание (ТЗ)?

Техническое задание (ТЗ) – это основной документ, который определяет требования к разрабатываемой системе, устанавливает цели проекта, описывает функциональные и нефункциональные характеристики, а также условия разработки и приемки. ТЗ является юридически значимым документом, на основе которого заключаются договоры и принимается результат работ.

Назначение ТЗ:

1. Договорная база: Фиксирует договоренности между заказчиком и разработчиком.
2. Единое понимание: Обеспечивает одинаковое понимание системы всеми участниками проекта.
3. План работ: Служит основой для планирования и оценки сроков.
4. Приемка: Является критерием для приемки готовой системы.

Виды ТЗ (по ГОСТ):

1. ГОСТ 34.602-2020 – Информационные технологии. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы.
2. ГОСТ 19.201-78 – Техническое задание на программное обеспечение (для программных продуктов).
3. Современные шаблоны – адаптированные под Agile, включающие только необходимые разделы.

Структура ТЗ по ГОСТ 34.602-2020 (таблица 30):

Таблица 30

№	Наименование раздела	Ключевое содержание
1	Общие сведения	Полное и условное наименование системы, шифр темы, реквизиты заказчика и разработчика, основания для разработки (договор, приказ), плановые сроки начала и окончания работ, порядок финансирования

№	Наименование раздела	Ключевое содержание
2	Цели и назначение создания системы	Цели: ожидаемые результаты внедрения (в числовых показателях) и критерии их достижения. Назначение: виды деятельности, подлежащие автоматизации (управление, учет, проектирование и т. п.)
3	Характеристика объекта автоматизации	Краткое описание объекта (предприятия, цеха, процесса), его структура, условия эксплуатации, параметры окружающей среды и другие сведения, влияющие на проектирование системы
4	Требования к автоматизированной системе (самый объемный раздел)	4.1. Требования к структуре АС в целом – состав подсистем, их назначение, взаимодействие между собой и со смежными системами, режимы работы, перспективы развития. 4.2. Требования к функциям – детальный перечень всех функций и задач с указанием периодичности, алгоритмов и результатов. 4.3. Требования к видам обеспечения – требования к математическому, информационному, программному, техническому, метрологическому, организационному и методическому обеспечению. 4.4. Общие технические требования – надежность, безопасность (электрическая, пожарная, экологическая), эргономика, защита информации, патентная чистота и стандартизация

№	Наименование раздела	Ключевое содержание
5	Состав и содержание работ по созданию системы	Перечень всех этапов работ (обследование, проектирование, кодирование, отладка, опытная эксплуатация) с указанием сроков, исполнителей и содержания каждого этапа
6	Порядок разработки автоматизированной системы (новый раздел)	Порядок взаимодействия заказчика и разработчика, правила отчетности на каждом этапе, сроки согласования документов и процедура внесения изменений
7	Порядок контроля и приемки системы	Виды испытаний (предварительные, приемочные, межведомственные), их состав и методики, перечень документов, оформляемых по результатам, а также статус приемочной комиссии
8	Требования по подготовке объекта к вводу системы в действие	Мероприятия, которые должен выполнить заказчик: ремонт помещений, прокладка сетей, обучение персонала, подготовка исходных данных для загрузки в систему
9	Требования к документированию	Полный перечень эксплуатационной документации, подлежащей передаче заказчику (руководство пользователя, администратора, формуляры, инструкции)
10	Источники разработки	Список нормативных актов, государственных стандартов, отчетов о НИР, технических регламентов и других материалов, которыми руководствовались при составлении ТЗ

Современная структура ТЗ (для коммерческих проектов) (таблица 31):

Таблица 31

Раздел	Содержание
1. Введение	Краткое описание проекта, термины и определения
2. Основание для разработки	Документы, на основе которых ведется разработка
3. Назначение и цели	Бизнес-цели, решаемые проблемы
4. Общее описание системы	Контекст, пользователи, ограничения
5. Функциональные требования	Детальное описание функций (с Use Case или User Stories)
6. Нефункциональные требования	Производительность, безопасность, надежность, удобство
7. Требования к интерфейсу	Ссылки на прототипы, требования к UI/UX
8. Требования к данным	Структура данных, модель БД
9. Требования к интеграции	Взаимодействие с внешними системами
10. Состав и содержание работ	Этапы, сроки, результаты
11. Порядок приемки	Критерии приемки, процедура тестирования
12. Требования к документации	Перечень документов
13. Приложения	Глоссарий, прототипы, схемы

Требования к хорошему ТЗ:

1. Полнота: Должно охватывать все аспекты системы.
2. Непротиворечивость: Разделы не должны противоречить друг другу.
3. Однозначность: Исключены двойные толкования.
4. Проверяемость: Каждое требование можно проверить.
5. Реалистичность: Достижимо в рамках бюджета и сроков.

Процесс разработки ТЗ:

1. Сбор информации: Интервью, анализ документов, изучение бизнес-процессов.
2. Анализ и моделирование: Построение моделей, прототипов.
3. Написание ТЗ: Структурирование информации по разделам.
4. Согласование: Обсуждение с заказчиком, внесение правок.
5. Утверждение: Подписание документа.

Задание

Цель работы: Разработать структуру и основные разделы технического задания на систему автоматизации книжного магазина, используя результаты предыдущих лабораторных работ.

Часть 1. Сбор материалов для ТЗ

Просмотрите результаты всех предыдущих лабораторных работ и соберите материалы, которые войдут в ТЗ.

Часть 2. Разработка структуры ТЗ

1. Определите, какие разделы будут включены в ваше ТЗ. Используйте современную структуру (более гибкую) или классический ГОСТ.
2. Создайте оглавление (план) ТЗ.

Часть 3. Наполнение разделов ТЗ

Выберите 4 ключевых раздела и наполните их содержанием, используя материалы предыдущих работ.

Часть 4. Оформление приложений

1. Определите, какие материалы лучше вынести в приложения.
2. Оформите описание одного приложения (например, "Прототипы интерфейса" со ссылкой на ЛР №12).

Контрольные вопросы

1. Что такое техническое задание (ТЗ) и для чего оно нужно?
2. Какие разделы обязательно должны быть в ТЗ по ГОСТ 34.602-2020?
3. Чем отличается современное ТЗ от классического ГОСТовского?

4. Почему ТЗ является юридически значимым документом?
5. Какие материалы из предыдущих этапов проектирования входят в ТЗ?
6. Что должно содержаться в разделе "Требования к системе"?
7. Зачем в ТЗ нужен раздел "Порядок контроля и приемки"?

Лабораторная работа №20
«Подготовка итоговой презентации проекта»

Теоретическая часть

Зачем нужна итоговая презентация?

Итоговая презентация проекта – это финальный документ, который демонстрирует результаты всей проделанной работы. Она может быть представлена разным аудиториям:

1. Заказчику: Для демонстрации того, что будет разработано.
2. Руководству: Для защиты проекта и выделения ресурсов.
3. Команде разработки: Для синхронизации понимания.
4. Учебной комиссии: Для защиты курсовой работы/диплома.

Цели презентации:

1. Донести основную идею: Показать, какую проблему решает система.
2. Продемонстрировать результаты: Представить ключевые артефакты проектирования.
3. Обосновать решения: Объяснить, почему выбрана именно такая архитектура и функциональность.
4. Показать готовность: Убедить аудиторию, что проект проработан достаточно для начала разработки.

Структура итоговой презентации (рекомендуемая) (таблица 32):

Таблица 32

Слайд	Содержание	Комментарий
1. Титульный слайд	Название проекта, команда, дата	Первое впечатление
2. Проблематика и цели	Какая проблема решается, цели проекта	Зачем это нужно?
3. Заинтересованные стороны	Кто будет использовать систему (персоны)	Ключевые пользователи
4. Обзор функциональности	Ключевые функции, приоритеты (MoSCoW)	Что система делает?

Слайд	Содержание	Комментарий
6. Варианты использования (Use Case)	Диаграмма прецедентов	Кто что может делать
7. Ключевые сценарии	Основной и альтернативные сценарии	Детали работы
8. Модель данных (ER-диаграмма)	Структура базы данных	Как хранятся данные
9. Архитектура системы	Компоненты и их взаимодействие	Из чего состоит система
10. Прототипы интерфейса	Ключевые экраны	Как это выглядит
11. Ограничения и риски	Технические и бизнес-ограничения, основные риски	Что важно учесть
12. Этапы разработки	План работ, сроки, этапы	Когда будет готово
13. Критерии приемки	Как будем принимать работу	Что считается успехом
14. Заключение и выводы	Итоги, следующие шаги	Резюме
15. Вопросы и ответы	Контакты, Q&A	Завершение

Задание

Цель работы: Разработать итоговую презентацию проекта автоматизации книжного магазина, обобщающую результаты всех предыдущих лабораторных работ.

Часть 1. Отбор материалов

1. Просмотрите все лабораторные работы №1–19 и выберите ключевые артефакты для включения в презентацию.
2. Составьте список того, что обязательно должно быть в презентации.

Часть 2. Создание структуры презентации

1. Определите последовательность слайдов (используйте рекомендуемую структуру или создайте свою).

2. Для каждого слайда определите основную мысль и какие материалы будут использованы.

Часть 3. Создание слайдов

1. Используя любой инструмент (PowerPoint, Google Slides, Canva, Keynote), создайте презентацию из 15–18 слайдов.
2. Для каждого слайда: Минимум текста (только тезисы), Максимум визуализации (диаграммы, схемы, скриншоты), Единый стиль оформления.

Часть 4. Подготовка текста выступления

1. Для каждого слайда напишите краткий текст выступления (что вы будете говорить).
2. Рассчитайте время: 15–18 слайдов = примерно 10–12 минут выступления.

Часть 5. Подготовка к вопросам

Предусмотрите возможные вопросы и подготовьте ответы.

Контрольные вопросы

1. Какова цель итоговой презентации проекта?
2. Какие разделы обязательно должны быть в презентации?
3. В чем разница между презентацией для заказчика и для разработчиков?
4. Что такое "правило 10/20/30" Гая Кавасаки?
5. Почему на слайдах должно быть минимум текста?
6. Какие артефакты из предыдущих лабораторных работ обязательно нужно включить в презентацию?
7. Как подготовиться к вопросам после презентации?

Лабораторная работа №21
«Комплексное моделирование системы»

Теоретическая часть

Что такое комплексное моделирование?

Комплексное моделирование – это создание целостной модели системы, объединяющей все аспекты: функциональные, структурные, поведенческие и информационные. Это не отдельные диаграммы, а их согласованная совокупность, где каждый элемент связан с другими и дополняет общую картину.

Зачем нужно комплексное моделирование?

1. Целостность: Обеспечивает непротиворечивость всех частей проекта.
2. Полнота: Позволяет увидеть, не упущены ли важные аспекты.
3. Связность: Показывает, как функциональные требования реализуются в архитектуре и данных.
4. Понимание: Дает полное представление о системе всем участникам проекта.

Единая модель системы (System Model)

В идеале все модели должны быть согласованы:

1. Акторы из Use Case соответствуют персонам и ролям в модели данных.
2. Сценарии Use Case реализуются через методы классов в объектной модели.
3. Атрибуты классов становятся полями таблиц в модели данных.
4. Функциональные требования покрываются вариантами использования.
5. Ограничения влияют на архитектуру и выбор технологий.

Методология комплексного моделирования:

1. Определение границ: Что входит в систему, что остается вовне.
2. Построение контекстной модели: Внешние сущности и основные потоки.
3. Функциональная декомпозиция: Разбиение на подсистемы/модули.

4. Поведенческое моделирование: Сценарии, состояния, взаимодействия.
5. Структурное моделирование: Классы, данные, компоненты.
6. Верификация и согласование: Проверка связей между моделями.

Инструменты комплексного моделирования:

1. Enterprise Architect – профессиональный инструмент с поддержкой трассировки.
2. IBM Rational Rhapsody – моделирование сложных систем.
3. Visual Paradigm – UML-инструмент с поддержкой трассировки.
4. Draw.io / Lucidchart – для небольших проектов (ручная проверка связей).

Задание

Цель работы: Выполнить комплексное моделирование системы книжного магазина, установив и проверив связи между всеми разработанными моделями.

Часть 1. Инвентаризация моделей

1. Составьте полный список всех моделей и артефактов, разработанных в предыдущих лабораторных работах.

Часть 2. Построение карты связей

1. Создайте диаграмму, показывающую, как связаны между собой основные артефакты.
2. Используйте прямоугольники для артефактов и стрелки для связей с подписями.
3. Для каждой связи кратко опишите характер отношения (например, "реализует", "детализирует", "соответствует", "использует").

Часть 3. Проверка горизонтальной целостности

1. Проверьте согласованность моделей одного уровня.
2. Если найдены несоответствия, предложите способы их устранения.

Контрольные вопросы

1. Что такое комплексное моделирование системы? Зачем оно нужно?
2. Какие виды целостности моделей существуют?
3. Как связаны диаграмма классов и модель данных?
4. Как проверить, что все функциональные требования реализованы в моделях?
5. Что такое сквозной пример (трассировка) и как он помогает в комплексном моделировании?
6. Какие инструменты поддерживают комплексное моделирование?
7. Почему важно проверять согласованность моделей перед началом разработки?

Лабораторная работа №22
«Анализ полноты и согласованности моделей»

Теоретическая часть

Что такое анализ полноты и согласованности?

Анализ полноты (Completeness Analysis) – проверка того, что все необходимые элементы системы описаны и ничто важное не упущено.

Анализ согласованности (Consistency Analysis) – проверка того, что различные модели и артефакты не противоречат друг другу.

Эти два вида анализа являются финальной проверкой качества проектирования перед началом разработки.

Виды полноты (таблица 33)**:**

Таблица 33

Тип полноты	Что проверяется	Пример нарушения
Функциональная	Все ли функции, требуемые заказчиком, описаны	Требование есть, но нет Use Case
Структурная	Все ли сущности предметной области описаны	В БД нет таблицы для хранения авторов
Поведенческая	Все ли сценарии использования описаны	Нет описания, что делать при ошибке оплаты
Информационная	Все ли данные, необходимые для работы, описаны	В форме нет поля для ввода цены
Ролевая	Все ли пользователи учтены	Нет описания прав для администратора

Виды согласованности (таблица 34):

Таблица 34

Тип согласованности	Что проверяется	Пример нарушения
Синтаксическая	Соответствие правилам нотации	На диаграмме классов стрелка ведет в никуда
Семантическая	Соответствие смыслов	В одном месте "чек" — документ, в другом — строка
Поведенческая	Непротиворечивость логики	По Use Case книга списывается, по модели данных — нет
Временная	Непротиворечивость во времени	Событие происходит до того, как создан объект
Межмодельная	Согласованность разных моделей	В Use Case актор "Продавец", в модели данных нет таблицы "Продавцы"

Методы анализа:

1. Матричное трассирование. Использование матриц связей для проверки, что каждый элемент одного типа связан с элементами другого типа.
2. Кросс-референсные списки. Перекрестные ссылки между артефактами (например, список требований с указанием реализующих их классов).
3. Формальные методы проверки. Использование математических методов для проверки логической непротиворечивости (редко в бизнес-проектах).
4. Экспертная оценка. Проверка опытными аналитиками и архитекторами.
5. Прототипирование и тестирование. Создание прототипов и написание тестов для выявления неполноты.

Задание

Цель работы: Провести комплексный анализ полноты и согласованности всех разработанных моделей системы книжного магазина, выявить несоответствия и предложить исправления.

Часть 1. Подготовка материалов

1. Соберите все артефакты, созданные в предыдущих лабораторных работах (или хотя бы их список с краткими характеристиками).
2. Сгруппируйте их по категориям (таблица 35):

Таблица 35

Категория	Артефакты
Требования	Список функциональных требований (ЛР №2), нефункциональные требования (ЛР №2), приоритеты (ЛР №3)
Пользователи	Персоны (ЛР №6), стейкхолдеры (ЛР №6), акторы Use Case (ЛР №7)
Функции	Use Case диаграмма (ЛР №7), сценарии (ЛР №14)
Поведение	Диаграммы последовательности (ЛР №8), диаграммы состояний (ЛР №8)
Структура	Диаграмма классов (ЛР №9), модель данных (ЛР №10, ЛР №11)
Архитектура	Диаграмма компонентов (ЛР №15), функциональные модули (ЛР №15)
Интерфейс	Прототипы (ЛР №12), спецификации экранов
Ограничения	Ограничения (ЛР №4), риски (ЛР №3)

Часть 2. Анализ полноты

1. Проверьте полноту по каждому направлению, используя чек-лист. Заполните таблицу.
2. Подсчитайте долю полноты (процент выполненных критериев).

Контрольные вопросы

1. Чем отличается анализ полноты от анализа согласованности?
2. Какие виды полноты вы знаете? Приведите примеры.
3. Какие виды согласованности существуют?

4. Что такое матрица согласованности и как она строится?
5. Как определить критичность найденного несоответствия?
6. Почему важно проводить анализ полноты и согласованности перед началом разработки?
7. Какие метрики можно использовать для оценки качества моделей?

Лабораторная работа №23
«Подготовка финального отчета»

Теоретическая часть

Что такое финальный отчет?

Финальный отчет – это итоговый документ, который обобщает все результаты работы над проектом. Он представляет собой полное описание процесса проектирования информационной системы, включая все созданные артефакты, анализ, выводы и рекомендации.

Назначение финального отчета:

1. Документирование результатов: Фиксация всех принятых решений и созданных моделей.
2. Передача знаний: Возможность для новых участников проекта быстро войти в курс дела.
3. Основа для разработки: Руководство для команды разработчиков.
4. Отчетность: Демонстрация выполненной работы заказчику или руководителю.
5. Архивирование: Сохранение информации для будущих проектов.

Структура финального отчета (рекомендуемая) дана в таблице 36:

Таблица 36

Раздел	Содержание	Источники
1. Введение	Цель работы, краткое описание проекта, область применения	ЛР №1
2. Анализ предметной области	Описание текущих бизнес-процессов (AS-IS), выявленные проблемы	ЛР №1, ЛР №5
3. Цели и задачи проекта	Бизнес-цели, функциональные и нефункциональные цели	ЛР№2
4. Заинтересованные стороны и пользователи	Стейкхолдеры, персоны, их потребности	ЛР №6

Раздел	Содержание	Источники
6. Моделирование бизнес-процессов	Модели AS-IS и TO-BE, анализ изменений	ЛР №5
7. Функциональное моделирование	Диаграммы Use Case, спецификации сценариев	ЛР №7, ЛР №14
8. Поведенческое моделирование	Диаграммы последовательности, диаграммы состояний	ЛР №8
9. Структурное моделирование	Диаграмма классов, описание объектной модели	ЛР №9
10. Моделирование данных	Логическая и физическая модели данных, ER-диаграмма, нормализация	ЛР №10, ЛР №11
11. Архитектура системы	Диаграмма компонентов, описание модулей, взаимодействие	ЛР №15
12. Проектирование интерфейса	Прототипы, описание ключевых экранов, требования к UI/UX	ЛР №12
13. Управление требованиями	Приоритизация, трассировка, управление изменениями	ЛР №3, ЛР №16, ЛР №17
14. Обеспечение качества	Контроль качества требований, анализ полноты и согласованности	ЛР №18, ЛР №22
15. Риски и ограничения	Анализ рисков, стратегии реагирования, ограничения проекта	ЛР №3, ЛР №4
16. План реализации	Этапы разработки, сроки, ресурсы	ЛР №19
17. Заключение	Основные результаты, выводы, рекомендации	-
18. Список литературы	Использованные источники, стандарты	-
19. Приложения	Глоссарий, полные спецификации, все диаграммы	ЛР №1, ЛР №4, и др.

Требования к оформлению отчета:

Общие требования:

1. Объем: 30–50 страниц (в зависимости от сложности проекта).
2. Шрифт: Times New Roman, 14 pt (для основного текста)
3. Межстрочный интервал: 1.5.
4. Поля: левое 30 мм, правое 15 мм, верхнее / нижнее 20 мм.
5. Нумерация страниц: внизу по центру.
6. Заголовки разделов: жирный шрифт, 16 pt.

Требования к иллюстрациям:

1. Все рисунки и таблицы должны иметь подписи (Рисунок 1 – Название, Таблица 1 – Название).
2. Ссылки на рисунки и таблицы в тексте.
3. Диаграммы должны быть читаемыми (не менее 300 dpi для растровых изображений).

Требования к стилю:

1. Формальный, деловой стиль.
2. Однозначность формулировок.
3. Отсутствие разговорных выражений.
4. Использование терминов из глоссария.

Задание

Цель работы: Подготовить финальный отчет по проекту автоматизации книжного магазина, объединяющий результаты всех лабораторных работ.

Часть 1. Сбор и систематизация материалов

1. Соберите все лабораторные работы №1–22.
2. Создайте папку с материалами, структурированную по разделам будущего отчета.

Часть 2. Разработка структуры отчета

1. Создайте документ в текстовом редакторе (Word, Google Docs, LaTeX).
2. Оформите титульный лист.
3. Создайте автоматическое оглавление.

Часть 3. Наполнение разделов.

Контрольные вопросы

1. Какова цель финального отчета по проекту?
2. Какие разделы обязательно должны присутствовать в финальном отчете?
3. Какие требования предъявляются к оформлению отчета (шрифты, поля, интервалы)?
4. Как правильно оформлять рисунки и таблицы в отчете?
5. Что должно содержаться во введении и заключении отчета?
6. Какие материалы целесообразно выносить в приложения?
7. Почему важно проверять отчет на наличие ошибок перед сдачей?

Список литературы

1. Вигерс, К. Разработка требований к программному обеспечению / К. Вигерс, Дж. Битти ; перевод с английского. – 3-е изд., доп. – Москва : Русская редакция, 2019. – 736 с. – ISBN 978-5-7502-0486-7.
2. Фаулер, М. UML. Основы / М. Фаулер ; перевод с английского. – 3-е изд. – Санкт-Петербург : Символ-Плюс, 2018. – 192 с. – ISBN 978-5-93286-221-4.
3. ГОСТ 34.602-2020. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. – Москва : Стандартинформ, 2021. – 24 с.
4. Коберн, А. Современные методы описания функциональных требований к системам / А. Коберн ; перевод с английского. – Москва : Лори, 2019. – 343 с. – ISBN 978-5-85582-423-6.
5. Ларман, К. Применение UML и шаблонов проектирования / К. Ларман ; перевод с английского. – 3-е изд. – Москва : Вильямс, 2020. – 736 с. – ISBN 978-5-907144-16-7.

Оглавление