

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Б1.В.ДВ.01.01

(индекс дисциплины)

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Инженерия требований и архитектура программных систем

(наименование дисциплины)

по направлению подготовки
09.03.04 Программная инженерия

направленность (профиль)
Программная инженерия с применением ИИ-технологий

Форма обучения: заочная

Год набора: 2024

Общая трудоемкость: 7 ЗЕ

Распределение часов дисциплины по семестрам

Курс Вид занятий	3	Итого
	зачет	
Лекции	6	6
Лабораторные		
Практические		
Руководство: курсовые работы (проекты) / РГР		
Промежуточная аттестация	0,25	0,25
Контактная работа	6,25	6,25
Самостоятельная работа	242	242
Контроль	3,75	3,75
Итого	252	252

Рабочую программу составил(и):

старший преподаватель института цифровых технологий, Рогова Н.Н.

(должность, ученое звание, степень, Фамилия И.О.)

Рецензирование рабочей программы дисциплины:



Отсутствует



Рецензент

(должность, ученое звание, степень, Фамилия И.О.)

Рабочая программа дисциплины составлена на основании ФГОС ВО и учебного плана
направления подготовки

09.03.04 Программная инженерия

Срок действия рабочей программы дисциплины до «31» августа 2031 г.

УТВЕРЖДЕНО

На заседании института цифровых технологий

(протокол заседания № 1 от «05» сентября 2025 г.).

1. Цель освоения дисциплины

Цель освоения дисциплины – формирование у обучающихся теоретических представлений о методологии инженерии требований и принципах построения архитектуры программных систем: их структурной организации, типовых архитектурных стилях и шаблонах проектирования, о роли и месте архитектурных решений в жизненном цикле программного обеспечения, методах обеспечения атрибутов качества и способах моделирования взаимодействия компонентов; и формирование практических навыков выявления и специфицирования требований, проектирования статических и динамических моделей системы, а также разработки и документирования архитектуры программных комплексов.

2. Место дисциплины в структуре ОПОП ВО

Дисциплины и практики, на освоении которых базируется данная дисциплина: «Управление требованиями к программному обеспечению».

Дисциплины и практики, для которых освоение данной дисциплины необходимо как предшествующее: «Управление проектами разработки программного обеспечения».

3. Планируемые результаты обучения

Формируемые и контролируемые компетенции (код и наименование)	Индикаторы достижения компетенций (код и наименование)	Планируемые результаты обучения
ПК-5 Способен проектировать архитектуру программного обеспечения и взаимодействие его компонентов	ПК-5 Способен проектировать архитектуру программного обеспечения и взаимодействие его компонентов	Знать: архитектурные стили (монолит, микросервисы, сервис-ориентированная архитектура). Уметь: анализировать преимущества и недостатки разных архитектур. Владеть: навыками анализа существующих архитектурных решений.
	ПК-5.2. Умеет проектировать архитектуру программного обеспечения и описывать взаимодействие его компонентов	Знать: шаблоны архитектурных решений. Уметь: выбирать и проектировать архитектуру, адекватную требованиям проекта. Владеть: навыками создания архитектурных диаграмм (компонентов, развертывания, последовательности).
	ПК-5.3. Владеет навыками проектирования архитектуры программного обеспечения	Знать: принципы масштабируемости, отказоустойчивости и безопасности на архитектурном уровне. Уметь: проектировать API и протоколы взаимодействия между компонентами. Владеть: практическими навыками проектирования системы с использованием выбранного архитектурного стиля.

4. Структура и содержание дисциплины

Модуль (раздел)	Вид учебной работы	Наименование тем занятий (учебной работы)	Курс	Объем, ч.	Баллы	Интерактив, ч.	Формы текущего контроля (наименование оценочного средства)
Модуль 1. Инженерия требований	СР	Роль инженерии требований и архитектуры в ЖЦ ПО.	3	2	6	—	Тестовые задания 1-193
	Лек 1	Методы выявления требований.	3	2		—	
	СР	Моделирование функциональных требований	3	2		—	
	СР	Нефункциональные требования и качество системы.	3	2		—	
	СР	Самостоятельное изучение методических рекомендаций при подготовке к практическим работам.	3	90		—	
	СР	ПР 1. Анализ предметной области, заинтересованных сторон и границ программной системы	3	4	8	—	Отчет по практической работе 1
	СР	ПР 2. Выявление и спецификация функциональных требований к системе	3	6	8	—	Отчет по практической работе 2
Модуль 2. Архитектура программных систем	Лек 2	Базовые понятия архитектуры ПО.	3	2	6	—	Тестовые задания 194-450
	Лек 3	Модульность и компоненты.	3	2		—	

Модуль (раздел)	Вид учебной работы	Наименование тем занятий (учебной работы)	Курс	Объем, ч.	Баллы	Интерактив, ч.	Формы текущего контроля (наименование оценочного средства)
	СР	Моделирование архитектуры. Диаграммы нотации UML.	3	2		—	
	СР	Моделирование архитектуры. Модель C4.	3	2		—	
	СР	Монолитная и слоистая архитектура.	3	2		—	
	СР	Распределенные системы и SOA.	3	2		—	
	СР	Событийно-ориентированная архитектура (EDA).	3	2		—	
	СР	Архитектура данных. Паттерны управления транзакциями в распределенных системах.	3	2		—	
	СР	Проектирование API. Стили API.	3	2		—	
	СР	Динамическое моделирование. Описание поведения системы во времени.	3	2		—	
	СР	Документирование архитектуры. Подходы к документации. Записи архитектурных решений.	3	2		—	
	СР	Оценка и эволюция архитектуры. Методы оценки (ATAM).	3	2		—	

Модуль (раздел)	Вид учебной работы	Наименование тем занятий (учебной работы)	Курс	Объем, ч.	Баллы	Интерактив, ч.	Формы текущего контроля (наименование оценочного средства)
	СР	Самостоятельное изучение методических рекомендаций при подготовке к практическим работам.	3	94		—	
	СР	ПР 3 Моделирование функциональных требований с использованием UML	3	4	8	—	Отчет по практической работе 3
	СР	ПР 4 Нефункциональные требования и архитектурные драйверы программной системы	3	6	8	—	Отчет по практической работе 4
	СР	ПР 5 Проектирование статической архитектуры: С4-модель, модульность и архитектурные стили	3	6	8	—	Отчет по практической работе 5
	СР	ПР 6 Динамическое моделирование, архитектура данных, API и оценка архитектуры	3	6	8	—	Отчет по практической работе 6
	ПА	Промежуточная аттестация	3	0,25		—	
	Контроль	Зачет	3	3,75	40	—	Тестовые задания 1-450
Итого:				252	100		

Схема расчета итогового балла: по накопительному рейтингу
Текущий рейтинг + Результат итогового теста

5. Образовательные технологии

В рамках учебного курса предусмотрены следующие образовательные технологии:

- технология дистанционного обучения: лекции, практические занятия, самостоятельная работа, реализуемые с применением информационно-телекоммуникационных сетей при опосредованном (на расстоянии) взаимодействии обучающихся и преподавателя.

6. Методические указания по освоению дисциплины

6.1. Рекомендации по подготовке к практическим занятиям

Обучающимся следует:

- при подготовке к практическим занятиям следует обязательно использовать не только лекции, учебную литературу, но и другие источники;
- в начале занятий задать преподавателю вопросы по материалу, вызвавшему затруднения в его понимании и освоении при решении задач, заданных для самостоятельного решения;
- на занятии доводить каждую задачу до окончательного решения, демонстрировать понимание проведенных расчетов (анализов, ситуаций), в случае затруднений обращаться к преподавателю.

Для того чтобы практические занятия приносили максимальную пользу, необходимо помнить, что решение задач проводится по рассмотренному на лекциях материалу и связаны, как правило, с детальным разбором отдельных вопросов лекционного курса. Следует подчеркнуть, что только после усвоения лекционного материала с определенной точки зрения (а именно с той, с которой он излагается на лекциях) он будет закрепляться обучающимся на практических занятиях как в результате обсуждения и анализа лекционного материала, так и с помощью решения проблемных ситуаций, задач. При этих условиях обучающийся не только хорошо усвоит материал, но и научится применять его на практике, а также получит дополнительный стимул (и это очень важно) для активной проработки лекции.

При самостоятельном решении задач нужно обосновывать каждый этап решения, исходя из теоретических положений курса. Если обучающийся видит несколько путей решения проблемы (задачи), то нужно сравнить их и выбрать самый рациональный. Полезно до начала вычислений составить краткий план решения проблемы (задачи). Решение проблемных задач или примеров следует излагать подробно, вычисления располагать в строгом порядке, отделяя вспомогательные вычисления от основных. Решения при необходимости нужно сопровождать комментариями, схемами, чертежами и рисунками.

Следует помнить, что решение каждой учебной задачи должно доводиться до окончательного логического ответа, которого требует условие, и по возможности с выводом. Полученный ответ следует проверить способами, вытекающими из существа данной задачи. Полезно также (если возможно) решать несколькими способами и сравнить полученные результаты. Решение задач данного типа нужно продолжать до приобретения твердых навыков в их решении.

6.2. Рекомендации по подготовке к зачету

Подготовка к зачету способствует закреплению, углублению и обобщению знаний, получаемых, в процессе обучения, а также применению их к решению практических задач. Готовясь к зачету, обучающийся ликвидирует имеющиеся пробелы в знаниях, углубляет, систематизирует и упорядочивает свои знания. На экзамене обучающийся демонстрирует то, что он приобрел в процессе обучения по конкретной учебной дисциплине.

Необходимо ориентировать обучающихся на систематическую подготовку к занятиям в течение семестра, что позволит использовать время экзаменационной сессии для систематизации знаний.

7. Оценочные средства

7.1. Паспорт оценочных средств

Курс	Код контролируемой компетенции (или ее части)	Наименование оценочного средства
3	ПК-5	Тестовые задания 1-450 Вопросы к зачету 1-80 Отчеты по практическим работам 1-6

7.2. Типовые задания или иные материалы, необходимые для текущего контроля

7.2.1. Типовые тестовые материалы

(наименование оценочного средства)

Типовые примеры заданий

Задание 1

Что является основной целью инженерии требований в жизненном цикле ПО?

Выберите один из 4 вариантов ответа:

1. Написание исходного кода
2. **Определение, документирование и поддержание требований к программному обеспечению**
3. Тестирование готовой системы
4. Развертывание системы у заказчика

Задание 2

Какая из перечисленных фаз жизненного цикла ПО напрямую зависит от качества выполненных работ по инженерии требований?

Выберите один из 4 вариантов ответа:

1. Кодирование
2. **Все последующие фазы (проектирование, кодирование, тестирование и т.д.)**
3. Сопровождение
4. Внедрение

Задание 3

Какова роль архитектуры программных систем по отношению к требованиям?

Выберите один из 4 вариантов ответа:

1. Архитектура не зависит от требований
2. **Архитектура является мостом между высокоуровневыми требованиями и конкретной реализацией**
3. Архитектура определяется только нефункциональными требованиями
4. Архитектура создается после полного завершения кодирования

Задание 4

Что такое "требование" согласно определению IEEE?

Выберите один из 4 вариантов ответа:

1. Любое пожелание заказчика
2. **Условие или возможность, необходимые пользователю для решения задачи или достижения цели**
3. Техническое задание на разработку
4. Функция, которую должна выполнять система

Задание 5

Каковы основные последствия некачественно выполненных работ по инженерии требований?

Выберите несколько из 4 вариантов ответа:

1. **Увеличение стоимости разработки**
2. **Увеличение сроков разработки**
3. **Низкое качество конечного продукта**
4. Ускорение процесса тестирования

Задание 6

Что из перечисленного входит в основные задачи инженерии требований?

Выберите несколько из 4 вариантов ответа:

1. **Выявление требований**
2. **Анализ требований**
3. **Документирование требований**
4. Написание пользовательской документации

Задание 7

В каком стандарте ISO/IEC рассматриваются процессы инженерии требований?

Выберите один из 4 вариантов ответа:

1. ISO 9001
2. **ISO/IEC/IEEE 29148**
3. ISO 14001
4. ISO/IEC 12207

Задание 8

Что такое "Жизненный цикл программного обеспечения" (ЖЦ ПО)?

Выберите один из 4 вариантов ответа:

1. Период времени от начала продаж до снятия с поддержки
2. **Последовательность этапов существования ПО от концепции до вывода из эксплуатации**
3. Время работы программы без сбоев
4. Процесс разработки кода

Задание 9

Какой из перечисленных процессов относится к инженерии требований?

Выберите один из 4 вариантов ответа:

1. **Управление требованиями**
2. Написание кода
3. Юнит-тестирование
4. Развертывание на production

Задание 10

Какая из перечисленных моделей ЖЦ ПО наиболее гибко реагирует на изменение требований?

Выберите один из 4 вариантов ответа:

1. Каскадная модель
2. V-образная модель
3. **Итеративная модель**
4. Спиральная модель

Критерии оценки за пройденный тест:

100 баллов выставляется обучающемуся, если он ответил правильно на все вопросы рандомной выборки 30 тестовых заданий;

0-99 баллов выставляется обучающемуся в зависимости от количества верных ответов

7.2.2. Пример практической работы

Практическая работа 1. Анализ предметной области, заинтересованных сторон и границ программной системы

Цель работы: изучить методы анализа предметной области и определения контекста программной системы; закрепить понятия «заинтересованная сторона», «границы системы», «окружение системы» и «контекст использования». Освоить приёмы систематизации информации о целях проекта и ожиданиях участников, научиться строить контекстную диаграмму и матрицу заинтересованных сторон. Развить навыки аналитического мышления, позволяющие на ранних этапах жизненного цикла ПО выявлять ключевые интересы, ограничения и риски.

Задание

1. Изучите методические указания по работе. Ознакомьтесь с основными понятиями: предметная область, программная система, заинтересованные стороны, границы системы, контекст системы, окружение. Обратите внимание на то, как выбор границ и участников проекта влияет на последующую работу с требованиями и архитектурой.
2. Выберите предметную область и кратко опишите её. Можно использовать предложенный преподавателем пример (например, «Система бронирования аудиторий университета») либо согласованный вами вариант. Подготовьте текстовое описание (0,5–1 страница), в котором отразите:
 - текущую ситуацию и проблемы, которые предполагается решать с помощью системы;
 - основные категории пользователей;
 - ожидаемый результат от внедрения системы.
3. Выделите заинтересованные стороны проекта. Сформируйте таблицу, в которую включите не менее 8 заинтересованных сторон разных типов (пользователи, заказчик, поддержка, администрация, внешние системы и др.). Рекомендуемая структура таблицы:

Идентификатор	Роль/ заинтересованная сторона	Краткое описание	Интересы и ожидания	Влияние на проект (низкое/ среднее/ высокое)	Критерии успеха

4. Сформулируйте цели системы и проекта. Опишите не менее 6–8 целей на уровне бизнеса и организации (не функций интерфейса). Оформите их в виде таблицы:

ID	Цель	Формулировка цели	Связанные заинтересованные стороны	Показатели достижения	Приоритет (низкое/ среднее/ высокое)

5. Определите границы и контекст системы.
Перечислите внешние системы, организации и устройства, с которыми будет взаимодействовать разрабатываемое ПО.
Для каждого элемента опишите:
- тип взаимодействия (обмен данными, получение отчётов, авторизация, внешние сервисы и др.);
 - направление обмена (входящий/исходящий/двусторонний);
 - вид передаваемой информации.
6. Постройте контекстную диаграмму системы. Используя любой инструмент для диаграмм (например, PlantUML, Diagrams.net), создайте диаграмму уровня контекста (по аналогии с C4 Level 1):
- в центре разместите разрабатываемую систему;
 - по периметру – внешних факторов и системы;
 - подпишите связи, указав тип взаимодействия и основные потоки данных.
- Сохраните диаграмму в формате изображения (PNG, PDF) для включения в отчёт.
7. Постройте матрицу «Влияние–заинтересованность» для сторон проекта.
По результатам шага 3 оцените:
- степень заинтересованности каждой стороны в проекте (низкая/средняя/высокая);
 - степень влияния на принятие решений (низкое/среднее/высокое).
- На основе этого:
- нарисуйте матрицу 2×2 с осями «Влияние» и «Заинтересованность»;
 - разместите в ней заинтересованные стороны (можно обозначать их идентификаторами из таблицы). Дополнительно составьте короткие рекомендации по взаимодействию с каждой группой (наблюдать, информировать, вовлекать, активно управлять).
8. Подготовьте отчёт по работе.
В отчёт включите:
- текстовое описание предметной области;
 - таблицу заинтересованных сторон;
 - таблицу целей;
 - таблицу внешних взаимодействий;
 - контекстную диаграмму;
 - матрицу «Влияние–заинтересованность» и краткие выводы (0,5–1 страница) о том, как выявленные заинтересованные стороны и контекст повлияют на дальнейшую разработку требований и архитектуры.
 - Выводы

Требования к оформлению

Отчет должен содержать подробное описание (включая иллюстрации). Отчёт по практическому занятию выполняется на страницах формата А4 в электронном виде.

При оформлении отчёта используется сквозная нумерация страниц, считая титульный лист первой страницей. Номер страницы на титульном листе не ставится. Номера страницы ставятся по центру сверху.

При оформлении отчёта соблюдать следующие требования:

- Для заголовков: полужирный шрифт, 14 пт, центрированный.
- Для основного текста: нежирный шрифт, 14 пт, выравнивание по ширине.
- Во всех случаях тип шрифта – Times New Roman, отступ абзаца 1.25 см, полуторный межстрочный интервал.
- Поля: левое – 2 см, правое, верхнее и нижнее – 1 см.

Процедура оценивания

Оценка выполненной практической работы проводится по следующим критериям:

1. Наличие всей существенной информации по работе
2. Точность и полнота предоставляемых сведений
3. Непротиворечивость приводимой информации
4. Правильность интерпретаций и выводов, которые сделаны по результатам работы
5. Степень достижения обучающимся поставленной цели
6. Обоснованность применяемого решения
7. Грамотность (содержательная) используемых формулировок

Критерии оценки за отчеты по практическим работам:

Формы текущего контроля	Критерии и нормы оценки
Отчеты по практическим работам 1, 2, 3, 4, 5, 6	8 баллов – задание выполнено в полном объеме без замечаний 7 баллов – задание выполнено в полном объеме, присутствуют замечания 6 баллов – задание выполнено в объеме 70%, замечаний нет. 5 баллов – задание выполнено в объеме 70%, присутствуют замечания. 4 балла – задание выполнено в объеме 50%, замечаний нет. 3 балла – задание выполнено в объеме 50%, присутствуют замечания. 2 балла - задание выполнено в объеме менее 50%, замечаний нет. 1 балл – задание выполнено в объеме менее 50%, присутствуют замечания. 0 баллов – задание не выполнено.

7.3. Оценочные средства для промежуточной аттестации по итогам освоения дисциплины

7.3.1. Вопросы к промежуточной аттестации

Курс _____ 3 _____

№	Вопросы к зачету
1.	Что такое инженерия требований и архитектура программной системы? Каково место этих дисциплин в жизненном цикле ПО? Как они взаимосвязаны между собой?
2.	Что понимается под «архитектурой программной системы» в современных стандартах? Какие элементы (структуры, поведение, решения, ограничения) обычно включает архитектура?
3.	Как различаются уровни «требования – архитектура – реализация»? Какие последствия возникают, если пропустить или формально выполнить архитектурный уровень?
4.	Какие преимущества даёт раннее проектирование архитектуры с точки зрения стоимости изменений и рисков проекта? В каких случаях можно ограничиться минимальной архитектурой?
5.	Какие типичные ошибки при старте проекта (отсутствие архитектурного видения, неявные требования, переусложнение) приводят к архитектурным проблемам в дальнейшем?

№	Вопросы к зачету
6.	Кто такие стейкхолдеры программной системы? Какие основные группы стейкхолдеров можно выделить (пользователи, заказчики, сопровождение и др.)? Как их интересы влияют на архитектуру?
7.	Что такое контекст системы и границы системы? Какие объекты и внешние системы обычно отображаются на контекстной диаграмме? Как определение границ помогает избежать «расползания» архитектуры
8.	Что такое бизнес-цели и бизнес-требования к системе? Как они трансформируются в системные и архитектурные требования? Почему архитектура должна быть связана с бизнес-целями?
9.	Как выявлять и документировать ожидания разных стейкхолдеров, чтобы они были учтены в архитектуре? Какие конфликты интересов стейкхолдеров наиболее типичны?
10.	Как изменение бизнес-целей (например, выход на новый рынок) может повлиять на уже существующую архитектуру? Какие свойства архитектуры позволяют легче переживать такие изменения?
11.	Какие методы извлечения требований вы знаете (интервью, воркшопы, наблюдение, анализ документов, анкетирование, прототипирование)? В чём особенности каждого из них?
12.	Что такое интервью при сборе требований? Какие виды интервью (структурированное, полуструктурированное, неструктурированное) применяются и когда? Как с помощью интервью выявлять архитектурные драйверы?
13.	Как прототипирование помогает в сборе и уточнении требований? В каких случаях прототип может существенно повлиять на архитектурные решения?
14.	В чём преимущества и недостатки анализа существующих документов и систем как метода извлечения требований? Какие архитектурные ограничения можно выявить таким способом?
15.	Почему важно комбинировать несколько методов извлечения требований? Какие риски возникают при опоре только на один метод (например, только на интервью с заказчиком)?
16.	Что такое функциональные требования к системе? Как функциональные требования связаны с поведением компонентов архитектуры?
17.	Что такое вариант использования (use case)? Какова типичная структура описания use case (актор, основной сценарий, альтернативы, исключения)? Как use cases помогают предложить архитектурную декомпозицию?
18.	Что такое пользовательские истории (user stories)? Как формулируется «As a ... I want ... so that ...»? Чем user stories отличаются от подробных use cases с точки зрения архитектурного проектирования?
19.	Для чего используется диаграмма вариантов использования UML? Какие элементы на ней отражаются (акторы, варианты использования, связи)? Как по такой диаграмме можно выделить подсистемы и границы компонентов?
20.	Как моделирование бизнес-процессов (например, с помощью BPMN) помогает выявлять требования к оркестрации компонентов и интеграции систем?
21.	Что такое нефункциональные требования (требования к качеству системы)? Какие основные группы нефункциональных требований вы можете перечислить? Почему именно они часто определяют архитектурные решения?
22.	Что такое атрибуты качества (производительность, масштабируемость, надёжность, безопасность и др.)? Как они формализуются и измеряются?
23.	Что такое сценарии качества? Какова типичная структура такого сценария? Приведите пример сценария качества для производительности и объясните, как он влияет на архитектуру?
24.	Какие типичные конфликты могут возникать между различными нефункциональными требованиями (например, безопасность vs. удобство, производительность vs. надёжность)? Как эти конфликты решаются на архитектурном уровне?
25.	Как требования к масштабируемости, доступности и отказоустойчивости отражаются на архитектуре взаимодействия компонентов и архитектуре развертывания?

№	Вопросы к зачету
26.	Что такое документ SRS (Software Requirements Specification)? Какова его типичная структура согласно стандартам? Как SRS используется при разработке архитектуры?
27.	Какие требования предъявляются к качеству формулировок требований (полнота, непротиворечивость, однозначность, проверяемость и др.)? Как несоблюдение этих требований влияет на архитектуру?
28.	Чем классический документ SRS отличается от product backlog в Agile-подходе? Какие преимущества и недостатки имеет каждый формат для архитектурного проектирования?
29.	Какие разделы или артефакты документации требований наиболее важны именно для архитектора (ограничения, сценарии качества, интерфейсные требования и др.)? Почему?
30.	Какие типичные ошибки встречаются при документировании требований (избыточная детализация, отсутствие приоритетов, смешение уровней абстракции)? Как их избежать?
31.	Что такое архитектура программной системы? Чем она отличается от детального проектирования и реализации? Какие решения считаются архитектурными
32.	Какие принципы архитектурного проектирования (модульность, слабое зацепление, высокая связанность внутри модуля, разделение ответственности) вы знаете? Как они влияют на качество системы?
33.	Что такое программный компонент на архитектурном уровне? Чем он отличается от модуля или класса? Какие свойства компонента (интерфейсы, независимость развёртывания) важны при проектировании?
34.	Какие критерии можно использовать для декомпозиции системы на компоненты (по функциям, по доменам, по изменчивости, по владению данными)? Как выбор критериев влияет на гибкость архитектуры?
35.	Какие типичные архитектурные риски возникают при неправильной декомпозиции системы (избыточная связанность, дублирование логики, узкие места по производительности)? Как их избежать?
36.	Что такое архитектурный стиль? Какие распространённые архитектурные стили вы знаете (слоистая архитектура, клиент–сервер, микросервисы, SOA, событийно-ориентированная архитектура)?
37.	В чём суть слоистой архитектуры? Какие типичные слои выделяются в бизнес-приложениях? Какие преимущества и недостатки имеет слоистый подход с точки зрения развития и тестирования системы?
38.	Чем микросервисная архитектура отличается от монолитной? Как различается подход к разбиению системы и организации взаимодействия компонентов в этих стилях? В каких случаях микросервисы оправданы?
39.	Что такое архитектурные паттерны (например, MVC, Repository, Dependency Injection)? Чем они отличаются от паттернов на уровне классов (GoF-паттерны)? Как паттерны помогают обеспечить требуемые атрибуты качества?
40.	Как выбор архитектурного стиля и паттернов влияет на организацию взаимодействия компонентов и на архитектуру развёртывания? Приведите пример такого влияния.
41.	Что такое архитектурные представления (views) и точки зрения (viewpoints)? Зачем для описания архитектуры нужны разные представления одной и той же системы?
42.	В чём суть модели 4+1 (Kruchten)? Какие четыре основных вида представлений она включает и что играет роль «+1»? Как эта модель помогает связать архитектуру с пользовательскими сценариями?
43.	Что такое модель C4 для визуализации архитектуры? Какие уровни (контекст, контейнеры, компоненты, код) она выделяет? Для каких аудиторий полезен каждый уровень?
44.	Как применяются диаграммы компонентов и диаграммы развёртывания UML при описании архитектуры? Какие аспекты системы они отражают?
45.	Что такое Architecture Decision Record (ADR)? Какие сведения в нём фиксируются (контекст, решение, обоснование, последствия)? Как ADR помогают управлять эволюцией архитектуры?

№	Вопросы к зачету
46.	Что такое архитектурные тактики в контексте атрибутов качества? Приведите примеры тактик для повышения производительности, масштабируемости и надёжности?
47.	Как архитектурные решения влияют на сопровождаемость и расширяемость системы? Какие приёмы (чёткое разделение ответственности, инверсия зависимостей, использование интерфейсов) здесь важны.
48.	В чём заключается метод АТАМ (Architecture Tradeoff Analysis Method)? Какие основные шаги он включает? Какие результаты даёт применение АТАМ в проекте?
49.	Как можно сравнивать альтернативные варианты архитектуры по сценариям качества? Какие критерии (затраты, риски, сложность) учитываются при выборе архитектурного решения?
50.	Что такое архитектурный технический долг? Какие причины приводят к его накоплению? Как технический долг влияет на качество и стоимость дальнейшего развития системы?
51.	Что такое компонент и интерфейс компонента? Как интерфейс отделяет реализацию компонента от его использования? Почему это важно для слабого зацепления?
52.	Что такое контрактное программирование (design by contract)? Какие элементы контракта (предусловия, постусловия, инварианты) можно задавать для интерфейсов компонентов?
53.	Чем синхронное взаимодействие компонентов отличается от асинхронного с точки зрения архитектуры? Как выбор синхронного или асинхронного взаимодействия влияет на производительность и надёжность системы?
54.	Как проектирование API компонентов (форматы данных, версии, протоколы) влияет на гибкость и расширяемость архитектуры? Какие ошибки в проектировании API чаще всего мешают развитию системы?
55.	Как обеспечить совместимость и эволюцию интерфейсов компонентов во времени (версионирование, обратная совместимость, адаптеры)? Какие подходы позволяют уменьшить влияние изменений API на остальные компоненты?
56.	Какие основные интеграционные паттерны вы знаете (точка-к-точке, шина сообщений, брокер сообщений, pub/sub)? В чём их отличия и области применения?
57.	Что такое очереди сообщений и брокеры сообщений (RabbitMQ, Kafka и др.)? Как они помогают реализовать асинхронное взаимодействие и повысить отказоустойчивость системы?
58.	В чём особенности REST-подхода к интеграции по сравнению с классическим RPC и gRPC? Как выбор между REST, RPC и gRPC влияет на производительность и эволюцию интерфейсов?
59.	Что такое событийно-ориентированная архитектура (Event-Driven Architecture)? Какие роли выполняют издатели и подписчики? В каких случаях событийная интеграция особенно эффективна?
60.	Что такое саги (sagas) в распределённых системах? Как с их помощью достигается согласованность данных без распределённых транзакций? В каких сценариях этот паттерн особенно полезен?
61.	Что такое архитектура развертывания программной системы? Какие элементы (узлы, среды, сети, компоненты) обычно показываются на схеме развертывания?
62.	Чем отличается логическая архитектура от архитектуры развертывания? Какие решения принимаются на каждом из этих уровней? Почему важно согласовывать их между собой?
63.	Как использование контейнеров (Docker) и оркестраторов (Kubernetes) влияет на архитектуру приложений? Какие новые требования к компонентам и их взаимодействию при этом появляются?
64.	Что такое концепция 12-факторного приложения? Какие из 12 факторов наиболее важны с архитектурной точки зрения (конфигурация, логи, горизонтальное масштабирование и др.)?
65.	Что такое наблюдаемость (observability) системы? Как требования к логированию, метрикам и распределённой трассировке отражаются на архитектуре компонентов и их взаимодействии?

№	Вопросы к зачету
66.	Какие основные подходы к эволюции архитектуры вы знаете (архитектурный рефакторинг, перепроектирование, постепенная миграция)? В каких случаях целесообразен каждый из них?
67.	Какие признаки указывают на то, что архитектура системы устарела или плохо соответствует текущим требованиям (трудность изменений, низкая производительность, частые сбои)?
68.	Какие стратегии миграции от монолитной архитектуры к микросервисной применяются на практике (strangler-pattern, выделение доменов и т.п.)? Каковы их преимущества и риски?
69.	Как управлять архитектурным техническим долгом в долгосрочном проекте? Какие практики (регулярные архитектурные ревью, плановый рефакторинг, приоритизация долгов) помогают держать долг под контролем?
70.	Какие подходы и инструменты позволяют поддерживать архитектурную документацию в актуальном состоянии (ADR-репозиторий, «живые» диаграммы, генерация из кода)? Почему важно, чтобы архитектурное описание не отставало от реализации?
71.	Что такое API (Application Programming Interface) в контексте программной архитектуры? Какие основные элементы входят в описание API (ресурсы, операции, форматы данных, протоколы)? Как качество проектирования API влияет на взаимодействие компонентов и эволюцию системы?
72.	Какие принципы хорошего проектирования API вы знаете (простота, единообразие, предсказуемость, минимизация связности)? Как обеспечить устойчивость API к изменениям во времени? Какие ошибки при проектировании API особенно опасны для архитектуры?
73.	В чём особенности REST-стиля при проектировании API? Какие принципы REST (идентификация ресурсов, stateless-взаимодействие, использование стандартных HTTP-методов) важны для архитектуры? В каких случаях REST-API является предпочтительным выбором?
74.	Чем RPC-подход (включая gRPC) к проектированию API отличается от REST-подхода? В каких случаях RPC-стиль более эффективен, а в каких — менее удобен? Как выбор между REST и RPC влияет на производительность, связность и тестируемость системы?
75.	Какие ещё стили и форматы API используются в современных системах (SOAP, GraphQL, event-driven API и др.)? В чём их преимущества и ограничения по сравнению с REST и RPC? Как архитектурные требования (производительность, гибкость выборки данных, безопасность) влияют на выбор стиля API?
76.	Что такое динамическое моделирование программной системы? Чем оно отличается от статического моделирования архитектуры? Какие вопросы о поведении системы во времени помогает ответить динамическое моделирование?
77.	Какие диаграммы UML относятся к средствам динамического моделирования (диаграмма последовательности, коммуникации, состояний, деятельности)? Какие аспекты поведения системы отражает каждая из этих диаграмм? В каких ситуациях предпочтительно использовать тот или иной вид диаграммы?
78.	Что такое диаграмма последовательности UML? Какие элементы на ней используются (объекты/компоненты, lifeline, сообщения, активации)? Как с помощью диаграммы последовательности описывается взаимодействие компонентов во времени в рамках конкретного сценария?
79.	Что такое диаграмма состояний UML? Какие элементы (состояния, переходы, события, действия) на ней отображаются? В каких случаях использование диаграмм состояний особенно полезно для описания поведения компонентов или доменных объектов?
80.	Как динамическое моделирование (диаграммы последовательности, состояний, деятельности) помогает проверять и уточнять архитектурные решения? Как несоответствие между статическими архитектурными моделями и динамическими диаграммами может указывать на проблемы во взаимодействии компонентов?

7.3.2. Критерии и нормы оценки

Курс	Форма проведения промежуточной аттестации	Критерии и нормы оценки	
3	Зачет (по накопительному рейтингу)	«зачтено»	рейтинговый балл 55-100
		«не зачтено»	рейтинговый балл 0-54

8. Учебно-методическое и информационное обеспечение дисциплины

8.1. Обязательная литература

№ п/п	Авторы, составители	Заглавие (заголовок)	Тип (учебник, учебное пособие, учебно-методическое пособие, практикум, др.)	Год издания	Количество в научной библиотеке / Наименование ЭБС
1.	Хабаров А. Н.	Хабаров А. Н. Разработка программных приложений : учебник / А. Н. Хабаров, А. Н. Ермакова. — Ставрополь : АГРУС, 2025. — 208 с.	учебник	2025	ЭБС «IPRbooks»
2.	Бурькова Е.В.	Бурькова Е. В. Проектирование микропроцессорных систем : учебное пособие / Е. В. Бурькова. — Оренбург : Оренбургский государственный университет, ЭБС АСВ, 2025. — 114 с. — ISBN 978-5-7410-3410-1	учебное пособие	2025	ЭБС «IPRbooks»
3.	Лыгина Н. И.	Лыгина Н. И. Разработка требований к программному продукту : учебное пособие / Н. И. Лыгина, О. В. Лауферман. — Новосибирск : НГТУ, 2023. — 76 с. — ISBN 978-5-7782-4987-5.	учебное пособие	2023	ЭБС «IPRbooks»
4.	Вичугова А. А	Вичугова А. А. Инструментальные средства информационных систем : учебное пособие / А. А. Вичугова. — 2-е изд. — Москва : Ай Пи Ар Медиа, 2024. — 135 с. — ISBN 978-5-4497-1248-6.	учебное пособие	2024	ЭБС «IPRbooks»
5.	Щенёва Ю. Б.	Щенёва Ю. Б. Проектирование и разработка клиент-серверных приложений. Ч.1 : учебное пособие / Ю. Б. Щенёва. — Рязань : Рязанский государственный радиотехнический университет, 2024. — 124 с.	учебное пособие	2024	ЭБС «IPRbooks»

8.2. Дополнительная литература

№ п/п	Авторы, составители	Заглавие (заголовок)	Тип (учебник, учебное пособие, учебно- методическое пособие, практикум, др.)	Год издания	Количество в научной библиотеке / Наименование ЭБС
1.	Курячий Г. В.	Онокой, Л. С. Проектирование информационных систем : учебное пособие / Л. С. Онокой, О. А. Морозова, Т. Е. Точилкина. — Москва : Прометей, 2024. — 352 с. — ISBN 978-5-00172-780-4.	учебное пособие	2024	ЭБС «IPRbooks»
2.	Тузовский А. Ф.	Тузовский, А. Ф. Проектирование и разработка web-приложений : учебное пособие / А. Ф. Тузовский. — 2-е изд. — Москва : Ай Пи Ар Медиа, 2024. — 218 с. — ISBN 978-5-4497-1293-6.	учебное пособие	2024	ЭБС «IPRbooks»

8.3. Перечень профессиональных баз данных и информационных справочных систем

№ пп	Наименование	Ссылка
1	Springer Nature (Полнотекстовая коллекция журналов)	https://www.springernature.com/gp/products
2	Springer eBooks (Полнотекстовая коллекция электронных книг издательства Springer Nature)	https://link.springer.com/
3	«Кодекс»	https://kodeks.ru/
4	ELIBRARY.RU (электронная библиотека научных публикаций)	http://elibrary.ru
5	"Гарант"	https://www.garant.ru/
6	"КонсультантПлюс"	https://www.consultant.ru/
7	Техэксперт	https://cntd.ru/

8.4. Перечень программного обеспечения

№ п/п	Наименование ПО	Реквизиты договора (дата, номер, срок действия)
1	WinPro 10 RUS Upgrd OLP NL Acdmc	Договор № 757 от 04.07.2018, срок действия - бессрочно; Контракт № 1653 от 14.12.2018, срок действия – бессрочно
2	Office Stdandard 2013 Russian OLP NL AcademicEdition	Контракт № 690 от 19.05.2015, срок действия - бессрочно
3	Diagrams.net Desktop (ранее Draw.io)	Лицензия: Apache License 2.0 (свободное и бесплатное ПО с открытым исходным кодом)
4	Visual Studio Code 1.x	Лицензия: MIT License (свободное и бесплатное ПО с открытым исходным кодом).
5	Postman	Лицензия: Postman EULA (бесплатная версия для образовательных целей)

8.5. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

№ п/п	Наименование оборудованных учебных кабинетов, лабораторий, мастерских и др. объектов для проведения практических и лабораторных занятий, помещений для самостоятельной работы обучающихся (номер аудитории)	Перечень основного оборудования
1	Компьютерный класс. Учебная аудитория для проведения занятий лекционного типа. Учебная аудитория для проведения занятий семинарского типа. Учебная аудитория для проведения лабораторных работ. Учебная аудитория для курсового проектирования (выполнения курсовых работ). Учебная аудитория для проведения групповых и индивидуальных консультаций. Учебная аудитория для проведения занятий текущего контроля и промежуточной аттестации. (УЛК-401).	Компьютер (монитор 19", системный блок Pentium (R) Dual-Core E5500 2,8 GHz / 4 Gb / 500 Gb), столы ученические, столы компьютерные, стол преподавательский, стулья, доска аудиторная (меловая).
2	Помещение для самостоятельной работы обучающихся (УЛК-105).	Стол, стулья, стеллажи (в т.ч. выставочные) с книгами, компьютеры, мобильные рабочие места.
3	Помещение для самостоятельной работы обучающихся (УЛК-406).	Стол компьютерный, стулья, микрокомпьютеры raspberry pi 32 bit.