МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский технологический университет «МИСИС»
Новотроицкий филиал

Рабочая программа дисциплины (модуля)

Программная инженерия

Закреплена за подразделением — Кафедра математики и естествознания (Новотроицкий филиал)

Направление подготовки — 09.03.03 Прикладная информатика

Профиль — Прикладная информатика в технических системах

Квалификация **Бакалавр**
Форма обучения **очная**
Общая трудоемкость **7 ЗЕТ**

Часов по учебному плану 252
в том числе:
аудиторные занятия 157
самостоятельная работа 68
часов на контроль 27

Формы контроля в семестрах:
экзамен 7
зачет 6
курсовая работа 7

Распределение часов дисциплины по семестрам

Семестр (<Курс>.<Семестр на курсе>)	**6 (3.2)**		**7 (4.1)**		Итого	
Недель	10		19			
Вид занятий	УП	РП	УП	РП	УП	РП
Лекции	18	18	17	17	35	35
Лабораторные	27	27	34	34	61	61
Практические	27	27	34	34	61	61
Итого ауд.	72	72	85	85	157	157
Контактная работа	72	72	85	85	157	157
Сам. работа	36	36	32	32	68	68
В том числе сам. работа в рамках ФОС		22		32		
Часы на контроль			27	27	27	27
Итого	108	108	144	144	252	252

Программу составил(и):
к.п.н, Доцент, Абдулвелеева Р.Р.

Рабочая программа
Программная инженерия

Разработана в соответствии с ОС ВО:

Самостоятельно устанавливаемый образовательный стандарт высшего образования - бакалавриат Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский технологический университет «МИСИС» по направлению подготовки 09.03.03 Прикладная информатика (приказ от 05.03.2020 г. № 95 о.в.)

Составлена на основании учебного плана:

09.03.03 Прикладная информатика, 09.03.03_25_Прикладная информатика_ПрПИвТС.plx Прикладная информатика в технических системах, утвержденного Ученым советом ФГАОУ ВО НИТУ "МИСиС" в составе соответствующей ОПОП ВО 25.12.2024, протокол № 58

Утверждена в составе ОПОП ВО:

09.03.03 Прикладная информатика, Прикладная информатика в технических системах, утвержденной Ученым советом ФГАОУ ВО НИТУ "МИСиС" 25.12.2024, протокол № 58

Рабочая программа одобрена на заседании
Кафедра математики и естествознания (Новотроицкий филиал)

Протокол от 12.03.2025 г., №3

Руководитель подразделения Швалева А. В.

1. ЦЕЛИ ОСВОЕНИЯ	
1.1	Цели освоения дисциплины: формирование у обучающихся представления о современных процессах проектирования, разработки, тестирования и эксплуатации программного продукта и о взаимосвязи всех аспектов программной инженерии.
1.2	Задачи:
1.3	- изучить понятийный аппарат дисциплины, основные теоретические положения и методы;
1.4	- сформировать умения и навыки применения теоретических знаний для решения профессиональных задач.

2. МЕСТО В СТРУКТУРЕ ОБРАЗОВАТЕЛЬНОЙ ПРОГРАММЫ	
Блок ОП:	Б1.В
2.1	**Требования к предварительной подготовке обучающегося:**
2.1.1	Технологии программирования
2.1.2	Экономика
2.1.3	Языки и среды разработки интернет-приложений
2.1.4	Вычислительные системы, сети и телекоммуникации
2.1.5	Языки программирования
2.1.6	Информационные системы и технологии
2.1.7	Компьютерная графика
2.1.8	Алгоритмизация и программирование
2.2	**Дисциплины (модули) и практики, для которых освоение данной дисциплины (модуля) необходимо как предшествующее:**
2.2.1	Моделирование металлургических процессов с использованием современных программных продуктов
2.2.2	Подготовка к процедуре защиты и защита выпускной квалификационной работы
2.2.3	Теоретическая механика
2.2.4	Электротехника, электроника и схемотехника

3. РЕЗУЛЬТАТЫ ОБУЧЕНИЯ ПО ДИСЦИПЛИНЕ, СООТНЕСЕННЫЕ С ФОРМИРУЕМЫМИ КОМПЕТЕНЦИЯМИ
УК-2: Способен собирать и интерпретировать данные и определять круг задач в рамках поставленной цели, выбирать оптимальные способы решения, исходя из действующих правовых норм, имеющихся ресурсов и ограничений, умение обосновывать принятые решения
Знать:
УК-2-З1 Основы проектирования, конструирования информационных систем
ОПК-7: Способен выбирать и применять методики проектирования и актуальные инструментальные средства, проектировать и разрабатывать алгоритмы и программы, пригодные для практического применения
Знать:
ОПК-7-З1 основные технологии создания и внедрения информационных систем, стандарты управления жизненным циклом информационной системы

4. СТРУКТУРА И СОДЕРЖАНИЕ

Код занятия	Наименование разделов и тем /вид занятия/	Семестр / Курс	Часов	Формируемые индикаторы компетенций	Литература и эл. ресурсы	Примечание	КМ	Выполняемые работы
	Раздел 1. Основные понятия и место программной инженерии							

1.1	Рынок программного обеспечения. Обзор технологий программирования (структурное, модульное, объектно-ориентированное, компонентное программирование). Программный продукт и его основные характеристики. Составляющие стоимости ПО. Предмет программной инженерии. Основные понятия. /Лек/	6	6		Л1.1 Л1.2Л2.1 Л2.3Л3.2 Э1 Э2 Э3 Э4		КМ1	
1.2	Выбор и обоснование темы проекта /Лаб/	6	4		Л1.1 Л1.2Л2.1Л3.2 Э1 Э2			Р2
1.3	Представление выбранной темы проекта на предмет актуальности, постановки цели и задач проекта. /Пр/	6	12		Л1.1 Л1.2Л2.1 Э1		КМ1	Р2
	Раздел 2. Жизненный цикл ПО: процессы, модели, стандарты							
2.1	Программный процесс и модель программного процесса. Методы программной инженерии. Роль стандартов в программной инженерии. Основные стандарты программной инженерии. Жизненный цикл программного продукта. Процесс, действие, задача жизненного цикла. Фазы (этапы) жизненного цикла. Основные процессы жизненного цикла ПО (ISO12207 и ISO 15504). Вспомогательные процессы жизненного цикла ПО (ISO12207 и ISO 15504). Организационные процессы жизненного цикла ПО (ISO12207 и ISO 15504). /Лек/	6	8		Л1.1 Л1.2Л2.1 Л2.2 Э1 Э2 Э3 Э4		КМ2	
2.2	Самостоятельное изучение учебного материала в LMS /Ср/	6	4		Л1.1 Л1.2Л2.1 Л2.2 Э1 Э2 Э3 Э4		КМ1	Р3
2.3	Модели жизненного цикла программной системы. Спецификация требований. /Пр/	6	8		Л1.1 Л1.2Л2.1 Л2.2Л3.1 Э1 Э2 Э3 Э4		КМ2	
2.4	Методология и стандарты создания программного обеспечения: Выбор темы проекта. /Лаб/	6	2		Л1.1 Л1.2Л2.1 Л2.2Л3.1 Л3.3 Э1 Э2 Э3 Э4			Р2,Р3,Р4
	Раздел 3. Анализ предметной области и требований к ПО							

3.1	Описание и анализ предметной области. Постановка задачи. /Лаб/	6	11		Л1.1 Л1.2Л2.1 Л2.2Л3.1 Л3.2 Л3.3 Э1 Э2 Э3 Э4			Р4,Р5
3.2	Анализ предметной области и требования к ПО. Разработка компонентов модели данных приложения. /Пр/	6	7		Л1.1 Л1.2Л2.1 Л2.2Л3.1 Л3.2 Л3.3 Э1 Э2 Э3 Э4		КМ2	Р3,Р4
3.3	Знакомство с требованиями к разрабатываемому ПО. Фиксация требований к ПО. Составление спецификации требований. Моделирование потребности заказчика. Методы выявления требований. Процесс анализа предметной области. Разработка модели системы в шаблоне «ввод-обработка-вывод». Принципы анализа: информационная область, моделирование, разделение на части, ракурсы видения основной информации и деталей реализации. Моделирование данных: объекты, свойства и связи данных, диаграммы связей между объектами. /Лек/	6	2		Л1.1 Л1.2Л2.1 Л2.2Л3.2 Л3.3 Э1 Э2 Э3 Э4		КМ2	
3.4	Подготовка к экзамену /Ср/	6	10		Л1.1 Л1.2Л2.1 Л2.2Л3.2		КМ4	Р2,Р3,Р4,Р5
	Раздел 4. Архитектура ПО							
4.1	Понятие архитектуры ПО. Цели и принципы системного проектирования. Структурное проектирование. Особенности объектно-ориентированного проектирования программных средств. /Лек/	6	2		Л1.1 Л1.2Л2.1 Л2.2Л3.2		КМ5	
4.2	Разработка внутренней структуры приложений при помощи диаграмм классов. Основы проектирования ПО. /Лаб/	6	10		Л1.1 Л1.2Л2.1 Л2.2Л3.1 Л3.2 Л3.3 Э1 Э2 Э3 Э4			Р2,Р10
	Раздел 5. Подготовка к контрольным мероприятиям и выполняемым работам							
5.1	Объем часов самостоятельной работы на подготовку к КМ /Ср/	6	12	УК-2-31	Л1.1 Л1.2Л2.1 Л2.3Л3.2 Л3.3 Э1 Э3 Э4		КМ5	Р2,Р3,Р4,Р5

5.2	Объем часов самостоятельной работы на подготовку к ВР /Ср/	6	10	УК-2-З1	Л1.1 Л1.2Л2.1 Л2.3Л3.1 Л3.2 Л3.3 Э1 Э4		КМ6	Р7
	Раздел 6. Качество ПО							
6.1	Стандартизация качества. Методы обеспечения качества ПО. Понятие тестирования. Инструменты тестирования. Виды тестирования. Принципы верификации и тестирования программ. Надежность и безопасность /Лек/	7	3		Л1.1 Л1.2Л2.1 Л2.2 Э1 Э2 Э3 Э4		КМ7,КМ8	
6.2	Тестирование программ и систем. Выдача задания для курсовой работы. /Пр/	7	14		Л1.1 Л1.2Л2.1 Л2.2Л3.1 Э1 Э2 Э3 Э4			Р12
	Раздел 7. Сопровождение ПО							
7.1	Организация и методы сопровождения программных средств. Этапы и процедуры при сопровождении программных средств. Задачи и процессы переноса программ и данных на иные платформы. Ресурсы, для обеспечения сопровождения и мониторинга программных средств. /Лек/	7	6		Л1.1 Л1.2Л2.1 Л2.2 Э1 Э2 Э3 Э4		КМ7	
7.2	Сопровождение программного обеспечения. /Пр/	7	6		Л1.1 Л1.2Л2.1 Л2.2Л3.1 Э1 Э2 Э3 Э4			Р13
7.3	Составление заявок предложений о модификации и поиски возможности их удовлетворения. Организация работ по сопровождению информационных систем. /Лаб/	7	18		Л1.1 Л1.2Л2.1 Л2.2Л3.1 Л3.3 Э1 Э2 Э3 Э4			Р11,Р10
	Раздел 8. Проектная деятельность в разработке ПО							

8.1	Управление проектами. Категории управления проектами. Особенности управления ИТ-проектами. Треугольник ограничений проекта. Ролевая модель команды. Роли и их ответственности. Модель управления командой. Критерии выбора модели. Административная модель, модель хаоса, модель открытой архитектуры. Особенности, преимущества и недостатки. Управление проектом разработки программного обеспечения. Концепция. Риски. Управление проектом разработки программного обеспечения. Планирование. Диаграмма Ганта. Средства управления проектом. Функции систем управления проектом. Обзор систем управления проектами. Оценка трудоемкости программного проекта. /Лек/	7	8		Л1.1 Л1.2Л2.1 Э1 Э2 Э3 Э4		КМ5,КМ6	
8.2	Конструирование программы /Лаб/	7	16		Л1.1 Л1.2Л2.1Л3.1 Л3.3 Э1 Э2 Э3 Э4			Р10,Р11, Р8
8.3	Диаграмма Ганта /Пр/	7	6					Р7
8.4	Командная разработка и работа над проектами. Диаграмма классов /Пр/	7	8	УК-2-З1	Л1.1 Л1.2Л3.1 Л3.3 Э1 Э3			Р14,Р6, Р9
	Раздел 9. Подготовка к контрольным мероприятиям и выполняемым работам							
9.1	Объем часов самостоятельной работы на подготовку к КМ /Ср/	7	14	УК-2-З1	Л1.2Л2.1 Л2.3Л3.1 Л3.3 Э1 Э3		КМ1,КМ2	Р9
9.2	Объем часов самостоятельной работы на подготовку к ВР /Ср/	7	18	ОПК-7-З1 УК-2-З1	Л1.1 Л1.2Л2.1 Л2.3Л3.1 Л3.3 Э1 Э3 Э4		КМ4	Р1

5. ФОНД ОЦЕНОЧНЫХ МАТЕРИАЛОВ

5.1. Контрольные мероприятия (контрольная работа, тест, коллоквиум, экзамен и т.п), вопросы для самостоятельной подготовки

Код КМ	Контрольное мероприятие	Проверяемые индикаторы компетенций	Вопросы для подготовки

КМ1	Тест	УК-2-З1	1. Программная инженерия – это а) Совокупность инструментальных средств и методов, предназначенных для создания качественного программного обеспечения. б) Совокупность инструментальных средств, предназначенных для создания качественного программного обеспечения. в) Совокупность навыков, инструментальных средств и методов, предназначенных для создания качественного программного обеспечения. г) Наука, изучающая построение программных систем д) Правила проектирования систем со сложной архитектурой 2. Программная инженерия занимается а) Вопросами оптимизации кода б) Вопросами разработки новых алгоритмов обработки данных в) Вопросами эффективной разработки программного обеспечения г) Применением средств быстрой разработки программного обеспечения д) Применением средств автоматизированного тестирования программного обеспечения 3. Стадии разработки программных систем, общие формы алгоритмов и схем, описывающих эти системы, регламентируются а) Стандартами ЕСПД б) Пунктами ТЗ в) Никак не регламентируются г) Эксплуатационными документами д) Спецификацией ПС 4. Псевдокод представляет собой а) Частично формализованный язык для представления описаний метода пошаговой детализации б) Язык, использующий конструкции структурного программирования в) Язык программирования высокого уровня г) Язык с неформальными фрагментами на естественном языке для представления обобщенных операторов и условий д) Формальная запись конструкций языка программирования Фортран

КМ2	Тест	УК-2-З1	2. Укажите основные процессы жизненного цикла по ГОСТ Р ИСО/МЭК 12207-99. «Информационная технология. Процессы жизненного цикла программных средств» а) Процесс заказа б) Процесс документирования в) Процесс разработки г) Процесс управления д) Процесс сопровождения 3. Проблемы, решаемые конфигурационным управлением а) Работа в команде б) Одновременная модификация в) Ограниченное уведомление г) Управление пользователями д) Множество версий 4. Этапы последовательной разработки ("водопад") а) Снятие с эксплуатации б) Тестирование в) Анализ требований г) Проектирование д) Системный анализ е) Использование и сопровождение 8. Этапы итеративного цикла разработки а) Тестирование б) бизнес-моделирование в) Реализация г) Анализ и проектирование д) Требования 5. Порядок разработки программного модуля а) Программирование (кодирование) модуля б) Шлифовка текста модуля в) Изучение и проверка спецификации модуля, выбор языка программирования г) Выбор алгоритма и структуры данных д) Компиляция модуля е) Проверка модуля

КМ3	Лабораторная работа	УК-2-З1	РАЗРАБОТКА СТРУКТУРЫ СИСТЕМЫ Построение структурной схемы программной системы. На данном этапе система по функциональному признаку разделяется на основные подсистемы, между ними указываются информационные связи и/или связи по управлению, описывается основное назначение подсистем. При разработке структурной схемы используется методология структурного проектирования, в основе которой лежит алгоритмическая декомпозиция и иерархия вида «часть-целое», учитывающая, что внутренние связи элементов внутри подсистем сильнее, чем связь между подсистемами. Декомпозиция системы может повторяться многократно, вплоть до уровня конкретных процедур, при этом должна быть обеспечена целостность системы, а все составляющие компоненты взаимоувязаны. Для этого используются такие принципы разработки, как «сверху-вниз», «разделяй и властвуй», «иерархическое упорядочивание» и другие. В первом приближении можно придерживаться нормативного понятия системы. Система (греч. - «составленное из частей», «соединение» от «соединяю») - множество элементов, находящихся в отношениях и связях друг с другом, которое образует определённую целостность, единство. Главным свойством системы является ее целостность: комплекс объектов, рассматриваемых в качестве системы, должен обладать общими свойствами и поведением. Необходимо рассматривать связи системы с внешней средой. В самом общем случае понятие «система» характеризуется: наличием множества элементов; наличием связей между ними; целостным характером данного устройства или процесса. Система должна представлять собой совокупность элементов (объектов, субъектов), находящихся между собой в определенной зависимости и составляющих некоторое единство (целостность), направленное на достижение определенной цели. Система может являться элементом другой системы более высокого порядка (надсистема) и включать в себя системы более низкого порядка (подсистемы). То есть систему можно рассматривать как набор подсистем, организованных для достижения определенной цели и описанных с помощью набора моделей, а подсистему – как группу элементов, часть которых составляет спецификацию поведения, представленного другими ее составляющими. К типовым можно отнести следующие подсистемы: подсистему управления; подсистемы ввода-вывода: подсистему настройки параметров; файловую подсистему; подсистему визуализации; подсистему документирования; подсистему взаимодействия с базой данных; справочную подсистему. Полученная в результате декомпозиции структура системы должна сопровождаться кратким описанием включенных в нее подсистем. В состав системы входят следующие подсистемы: Подсистема управления, которая отвечает за взаимодействие подсистем между собой и представлена в виде иерархического меню; Справочная подсистема, которая содержит сведения о системе (руководство пользователю) и ее об ее разработчиках.

КМ4	Экзамен	УК-2-З1	1. Основные понятия и определения программной инженерии. 2. IT-проекты: понятие, особенности реализации, статистика успешности, причины неудач. 3. Инженерия программного обеспечения (программная инженерия): история, определения, инженерная деятельность, область действия программной инженерии. 4. Требования к ПО: функциональные и нефункциональные. 5. Проектирование ПО: виды дизайна, цели архитектуры, техники проектирования. 6. Проектирование ПО: ключевые вопросы проектирования, уровни архитектуры, оценка качества проектирования. 7. Проектирование ПО: стратегии и методы, нотации проектирования. 8. Конструирование ПО: основы конструирования. 9. Конструирование ПО: управление конструированием, языки конструирования. 10. Конструирование ПО: кодирование, тестирование в конструировании, оценка качества, интеграция. 11. Тестирование ПО: основные концепции и определение тестирования, уровни тестирования. 12. Тестирование ПО: техники тестирования, метрики тестирования, управление тестированием. 13. Сопровождение ПО: основные концепции, категории сопровождения, ключевые вопросы сопровождения ПО. 14. Сопровождение ПО: процессы сопровождения, техники сопровождения. 15. Управление конфигурацией ПО: основные понятия, составляющие части. 16. Управление проектами ПО: организационное, задачи, инженерия измерений. 17. Процесс инженерии ПО: определения, составляющие части. 18. Методы инженерии ПО: классификация методов, категории методов. 19. Инструменты инженерии ПО: классификация, области применения. 20. Качество ПО: внешние и внутренние характеристики качества, основные и дополнительные, техники гарантии качества. 21. Стандарт ISO 12207. Основные определения. 22. Стандарт ISO 12207. Процессы жизненного цикла (ЖЦ). 23. Модель жизненного цикла разработки ПО как процесс (обобщенная схема, иерархия элементов). 24. Каскадная (водопадная) модель жизненного цикла ПП. 25. Итеративная и инкрементальная модель жизненного цикла ПП. 26. Спиральная модель жизненного цикла ПП. 27. V-образная модель жизненного цикла ПП. 28. Инкрементная (пошаговая) модель жизненного цикла ПП. 29. Модель быстрого прототипирования. 30. Модель жизненного цикла MSF. 31. Модель жизненного цикла RUP. 32. Модель жизненного цикла XP. 33. Управление командой проекта: ролевая модель команды. 34. Модели организации команд: административная модель,

КМ5	Тест	УК-2-З1	1. Что включает в себя треугольник ограничений проекта? • A) Время, стоимость, качество • B) Время, стоимость, риски • C) Время, стоимость, ресурсы • D) Время, стоимость, коммуникации _____ 2. Какая из перечисленных ролей отвечает за общее руководство проектом? • A) Менеджер проекта • B) Разработчик • C) Тестировщик • D) Бизнес-аналитик _____ 3. Какая модель управления командой предполагает полную свободу действий для участников команды? • A) Административная модель • B) Модель хаоса • C) Модель открытой архитектуры • D) Иерархическая модель _____ 4. Что является ключевым критерием выбора модели управления командой? • A) Размер команды • B) Опыт участников • C) Сложность проекта • D) Все перечисленное . _____ 5. Какая модель управления командой предполагает строгую иерархию и централизованное принятие решений? • A) Административная модель • B) Модель хаоса • C) Модель открытой архитектуры • D) Самоорганизующаяся модель 6. Какая из перечисленных ролей отвечает за анализ требований к проекту? • A) Менеджер проекта • B) Бизнес-аналитик • C) Разработчик • D) Тестировщик 7. Что является особенностью управления ИТ-проектами? • A) Высокая неопределённость требований • B) Быстрое изменение технологий • C) Необходимость частой коммуникации с заказчиком • D) Все перечисленное _____ 8. Какая модель управления командой предполагает гибкость и адаптивность к изменениям? • A) Административная модель • B) Модель хаоса • C) Модель открытой архитектуры • D) Иерархическая модель _____ 9. Что является основным преимуществом административной модели управления? • A) Быстрое принятие решений • B) Гибкость • C) Минимальный контроль • D) Высокая автономия команды . _____ 10. Какая концепция лежит в основе управления проектами разработки программного обеспечения? • A) Agile • B) Waterfall • C) Scrum • D) Все перечисленное

КМ6	Тест	УК-2-З1	1. Что такое риск в управлении проектами? • A) Возможность отклонения от плана • B) Гарантированное событие, влияющее на проект • C) Ресурс, необходимый для выполнения проекта • D) План действий на случай непредвиденных обстоятельств 2. Какой инструмент используется для визуализации расписания проекта? • A) Диаграмма Ганта • B) Критический путь • C) Матрица рисков • D) Бюджет проекта 3. Что такое критический путь в управлении проектами? • A) Наиболее длинная последовательность задач, определяющая сроки проекта • B) Наиболее рискованные задачи проекта • C) Задачи, которые можно выполнить параллельно • D) Задачи, которые не влияют на сроки проекта 4. Какая функция систем управления проектами позволяет отслеживать выполнение задач? • A) Планирование • B) Контроль • C) Отчётность • D) Все перечисленное . 5. Какой метод используется для оценки трудоёмкости программного проекта? • A) Метод функциональных точек • B) Метод экспертных оценок • C) Метод аналогий • D) Все перечисленное 6. Какая из перечисленных систем управления проектами является наиболее популярной? • A) Microsoft Project • B) Jira • C) Trello • D) Все перечисленное 7. Что включает в себя планирование проекта разработки ПО? • A) Определение задач, сроков и ресурсов • B) Оценку рисков • C) Создание диаграммы Ганта • D) Все перечисленное 8. Какой метод используется для анализа рисков в проекте? • A) SWOT-анализ • B) Анализ критического пути • C) Матрица вероятности и воздействия • D) Все перечисленное 9. Какая из перечисленных систем управления проектами подходит для Agile-проектов? • A) Jira • B) Trello • C) Asana • D) Все перечисленное 10. Что такое оценка трудоёмкости программного проекта? • A) Определение времени, необходимого для выполнения задач • B) Определение стоимости проекта • C) Определение рисков проекта • D) Определение качества проекта

КМ7	Тест	УК-2-З1	1. Что такое стандартизация качества в разработке ПО? A) Процесс формализации требований к продукту B) Следование международным стандартам (ISO, IEEE) для обеспечения качества C) Документирование кода без тестирования D) Использование только Agile-методологий 2. Какой стандарт наиболее часто применяется для управления качеством ПО? A) ISO 9001 B) IEEE 802.11 C) GDPR D) HTML5 3. Какой метод обеспечения качества ПО предполагает проверку кода без его выполнения? A) Динамическое тестирование B) Статический анализ C) Регрессионное тестирование D) Нагрузочное тестирование 4. Что такое тестирование "черного ящика"? A) Тестирование без доступа к исходному коду, только по требованиям B) Тестирование внутренней структуры кода C) Тестирование безопасности системы D) Автоматизированное тестирование 5. Какой инструмент используется для автоматизированного тестирования веб-приложений? A) Selenium B) Jira C) Git D) Docker 6. Какой вид тестирования проверяет взаимодействие между модулями системы? A) Модульное тестирование (Unit Testing) B) Интеграционное тестирование C) Системное тестирование D) Регрессионное тестирование 7. Какой принцип тестирования гласит: "Тестирование не может доказать отсутствие ошибок"? A) Принцип раннего тестирования B) Принцип исчерпывающего тестирования C) Принцип отсутствия ошибок (No Free Bugs) D) Принцип пестицида 8. Что такое верификация в контексте разработки ПО? A) Проверка соответствия продукта требованиям (правильно ли мы делаем продукт?) B) Проверка, удовлетворяет ли продукт потребностям пользователя C) Тестирование производительности системы D) Анализ уязвимостей безопасности 9. Какой вид тестирования оценивает устойчивость системы под нагрузкой? A) Регрессионное тестирование B) Нагрузочное тестирование (Load Testing) C) Юзабилити-тестирование D) Модульное тестирование 10. Какой стандарт регулирует безопасность информационных систем? A) ISO 27001 B) IEEE 829 C) SQL-92 D) REST API

КМ8	Тест	УК-2-З1	Тест по теме: Тестирование программ и систем 1. Что является основной целью тестирования программного обеспечения? A) Доказать, что программа не содержит ошибок B) Найти как можно больше дефектов C) Подтвердить соответствие программы требованиям D) Ускорить разработку 2. Какой вид тестирования проверяет отдельные модули или функции программы? A) Интеграционное тестирование B) Системное тестирование C) Модульное тестирование (Unit Testing) D) Регрессионное тестирование 3. Что такое "регрессионное тестирование"? A) Тестирование новых функций программы B) Проверка, что исправление ошибок не повредило существующий функционал C) Тестирование интерфейса пользователя D) Оценка производительности системы 4. Какой метод тестирования предполагает знание внутренней структуры кода? A) Тестирование "черного ящика" B) Тестирование "белого ящика" C) Тестирование "серого ящика" D) Нагрузочное тестирование 5. Какой инструмент используется для автоматизированного тестирования API? A) Selenium B) Postman C) Jira D) Docker 6. Какой вид тестирования оценивает удобство использования интерфейса? A) Юзабилити-тестирование B) Нагрузочное тестирование C) Модульное тестирование D) Интеграционное тестирование 7. Что проверяет "нагрузочное тестирование"? A) Корректность работы функций B) Устойчивость системы под высокой нагрузкой C) Безопасность системы D) Соответствие требованиям 8. Какой принцип тестирования гласит: "Раннее тестирование экономит время и деньги"? A) Принцип пестицида B) Принцип раннего тестирования C) Принцип отсутствия ошибок D) Принцип кластеризации дефектов 9. Какой вид тестирования имитирует атаки на систему для проверки уязвимостей? A) Нагрузочное тестирование B) Регрессионное тестирование C) Тестирование безопасности (Penetration Testing) D) Модульное тестирование 10. Какой стандарт описывает процесс тестирования ПО? A) ISO 9001 B) IEEE 829 C) GDPR D) HTML5

5.2. Перечень работ, выполняемых по дисциплине (Курсовая работа, Курсовой проект, РГР, Реферат, ЛР, ПР и т.п.)			
Код работы	Название работы	Проверяемые индикаторы компетенций	Содержание работы
Р1	Курсовая работа	УК-2-З1	Примерная тематика курсовых работ 1. Программное обеспечение банкомата. 2. Информационная система публичной библиотеки. 3. Информационная система поликлиники. 4. Информационная система деканата. 5. Система мгновенного обмена сообщениями. 6. Информационная система склада. 7. Система учета рабочего времени. 8. Информационная система жилищного агентства. 9. Информационная система технической экспертизы. 10. Система продажи билетов для проезда 11. Пакет программного обеспечения для регистратора в больнице 12. Программная система для call-центра банка 13. Организация и ведение спортивного чемпионата 14. Построение расписания занятий в ВУЗе 15. Автоматизация работы компании по аренде жилых и нежилых помещений 16. Автоматизация работы автосалона 17. Программа ведения личной библиотеки 18. Программа учета транспортных средств предприятия 19. Автоматизация отдела кадров предприятия 20. Автоматизация работы торгового представителя розничных продовольственных това- ров
Р2	Лабораторная работа	ОПК-7-З1	Выбор темы проекта В качестве темы возможно использовать курсовую работу по объектно-ориентированному программированию и для нее выполнить этапы: бизнес-анализа, бизнес-моделирования, проектирования архитектуры, документирования указанных этапов. «Исходные данные к проекту» включает в себя следующие подразделы: Характеристики объекта автоматизации (или управления); Требования к информационному обеспечению. Требования к техническому обеспечению. Требования к программному обеспечению. Общие требования к проектируемой системе. Перечень дополнительных работ (если необходимо). Характеристики объекта автоматизации. Здесь указываются общие характеристики объекта автоматизации, характерные для рассматриваемой предметной области: полное название объекта (ов); условия его функционирования; количественные и качественные показатели объекта, которые являются ограничениями процесса функционирования.

Р3	Лабораторная работа	ОПК-7-З1	ОПИСАНИЕ И АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ RAD включает в себя элементы методологии объектно-ориентированного проектирования и анализа предметной области. Для быстрой и эффективной разработки программной системы с минимальным браком требуется определить верное направление работы. Чтобы правильно построить систему, сначала необходимо построить ее модель. Принцип моделирования: «Лучшие модели - те, что ближе к реальности». Сначала нужно подробно изучить предметную область, для которой разрабатывается программа. В соответствии с методологией ООАП выделяются следующие шаги работы над проектом (системой). Описание предметной области: «Под предметной областью (application domain) принято понимать ту часть реального мира, которая имеет существенное значение или непосредственное отношение к процессу функционирования программы. Предметная область включает в себя только те объекты и взаимосвязи между ними, которые необходимы для описания требований и условий решения некоторой задачи. Задание 1. Выделите основные объекты (компоненты), участвующие в функционировании системы, определите их наиболее существенные характеристики, взаимосвязи в рамках решаемой задачи. Определите основные информационные потоки в системе. Компоненты выбираются таким образом, чтобы при последующей разработке их было удобно представить в форме классов и объектов. Задание 2. Выберите язык представления информации о концептуальной схеме предметной области. Сложность предметной области определяет количество объектов и связей между ними. Опишите базовые термины и определения предметной области. Подберите различные примеры. Можно приводить различного рода классификации, поясняющие различные свойства описываемых объектов. Если в системе используются математические модели, то они также должны быть описаны с учетом специфики применения.
Р4	Лабораторная работа	УК-2-З1	ПОСТАНОВКА ЗАДАЧИ Постановка задачи – заключительный этап первой фазы ЖЦ системы. На данном этапе формулируются все требования, которым должна удовлетворять система. Постановка задачи пишется в повествовательной форме в будущем времени на основе ТЗ, в ней должны быть обязательно взаимоувязаны виды автоматизируемой деятельности (с привязкой к объекту(ам) автоматизации) со всеми ограничениями, накладываемыми на них, учтены особенности разрабатываемого информационного обеспечения и перечислены функции, которые должна выполнять система (с привязкой к процессам и информационному обеспечению). В системе также должна быть обеспечена возможность получения справочной информации как о самой системе, так и предоставляемых ею возможностях. Система должна выполнять следующие функции: (здесь перечисляются все функции, которые были определены в разделе 2.5.1 ТЗ)». 5 см. раздел 2.2 ТЗ «Требования к информационному обеспечению системы» 6 Последняя функция не обязательна, т.к. достаточно сложна при реализации. 7 см. раздел 2.5.1 ТЗ «Функции, реализуемые системой». 8 Эта часть постановки задачи обязательна.

Р5	Лабораторная работа	УК-2-З1	РАЗРАБОТКА СТРУКТУРЫ СИСТЕМЫ Построение структурной схемы программной системы. На данном этапе система по функциональному признаку разделяется на основные подсистемы, между ними указываются информационные связи и/или связи по управлению, описывается основное назначение подсистем. При разработке структурной схемы используется методология структурного проектирования, в основе которой лежит алгоритмическая декомпозиция и иерархия вида «часть-целое», учитывающая, что внутренние связи элементов внутри подсистем сильнее, чем связь между подсистемами. Декомпозиция системы может повторяться многократно, вплоть до уровня конкретных процедур, при этом должна быть обеспечена целостность системы, а все составляющие компоненты взаимоувязаны. Для этого используются такие принципы разработки, как «сверху-вниз», «разделяй и властвуй», «иерархическое упорядочивание» и другие. Система (греч. - «составленное из частей», «соединение» от «соединяю») - множество элементов, находящихся в отношениях и связях друг с другом, которое образует определённую целостность, единство. Главным свойством системы является ее целостность: комплекс объектов, рассматриваемых в качестве системы, должен обладать общими свойствами и поведением. Необходимо рассматривать и связи системы с внешней средой. В самом общем случае понятие «система» характеризуется: наличием множества элементов; наличием связей между ними; целостным характером данного устройства или процесса. Система должна представлять собой совокупность элементов (объектов, субъектов), находящихся между собой в определенной зависимости и составляющих некоторое единство (целостность), направленное на достижение определенной цели. Система может являться элементом другой системы более высокого порядка (надсистема) и включать в себя системы более низкого порядка (подсистемы). То есть систему можно рассматривать как набор подсистем, организованных для достижения определенной цели и описанных с помощью набора моделей (возможно, с различных точек зрения), а подсистему – как группу элементов, часть которых составляет спецификацию поведения, представленного другими ее составляющими. К типовым можно отнести следующие подсистемы: подсистему управления; подсистемы ввода-вывода: подсистему настройки параметров; файловую подсистему; подсистему визуализации; подсистему документирования; подсистему взаимодействия с базой данных; справочную подсистему. Полученная в результате декомпозиции структура системы должна сопровождаться кратким описанием включенных в нее подсистем. В состав системы входят следующие подсистемы: Подсистема управления, которая отвечает за взаимодействие подсистем между собой и представлена в виде иерархического меню; ... Файловая подсистема, которая … Подсистема работы со словарем, которая … Подсистема визуализации, которая … Справочная подсистема, которая содержит сведения о системе (руководство пользователю) и ее об ее разработчиках.

Р6	Практическая работа	УК-2-З1	Практичекое занятие 1 Тема: Командная разработка и работа над проектами Шаг 1: Определение идеи для проекта Начнем с определения идеи для нашего проекта. Это может быть мобильное приложение, веб-сервис или любое другое программное обеспечение. Важно выбрать такую идею, которая будет интересна всей команде и при этом выполнима за отведенное время. Пример идеи: Разработка мобильного приложения для отслеживания фитнеса и здоровья. Шаг 2: Распределение ролей в команде Теперь распределим роли среди участников команды. Это поможет обеспечить эффективное взаимодействие и четкое понимание обязанностей каждого члена команды. Роли:Менеджер проекта: Ответственный за координацию работы команды, контроль сроков и качества выполнения задач.Разработчики: Занимаются написанием кода, реализацией функционала и интеграцией различных модулей.Тестировщики: Проверяют качество продукта, выявляют баги и ошибки, предлагают улучшения. Шаг 3: Разработка плана проекта Составляем детальный план проекта, который включает ключевые этапы и сроки выполнения каждой задачи. План должен быть реалистичным и учитывать возможные риски. План проекта: Анализ требований и проектирование (1 неделя):Определение основных функций приложения.Создание технического задания.Проектирование архитектуры системы. Разработка прототипа (2 недели):Создание макетов интерфейса.Разработка основной логики приложения.Первичная интеграция модулей. Основная разработка (4 недели):Написание основного кода.Интеграция всех модулей.Оптимизация производительности. Тестирование и исправление багов (2 недели):Проведение тестов на разных устройствах и платформах.Исправление найденных ошибок.Улучшение UX/UI. Финальная подготовка и защита (1 неделя):Подготовка презентации проекта.Полировка финального продукта.Демонстрация перед аудиторией. Шаг 4: Начало разработки проекта Начинаем реализацию проекта согласно составленному плану. Важно следить за выполнением задач и соблюдением сроков. Задачи разработчиков:Разработка backend-сервиса.Разработка frontend-интерфейса.Интеграция базы данных.Реализация функционала трекинга активности. Задачи тестировщиков:Регулярное тестирование новых версий приложения.Составление отчетов о найденных ошибках.Предложение улучшений и оптимизации. Шаг 5: Регулярные встречи для обсуждения прогресса

			Проводим еженедельные встречи для обсуждения текущего состояния проекта, решения возникающих проблем и координации дальнейших действий. Повестка встреч:Обзор выполненных задач.Обсуждение текущих проблем и вопросов.Постановка новых задач на следующую неделю.Оценка общего прогресса проекта. Шаг 6: Завершение разработки проекта По мере завершения всех запланированных этапов переходим к финальным шагам подготовки проекта к защите. Завершающие задачи:Окончательная проверка качества продукта.Подготовка документации и инструкций по использованию.Создание презентационных материалов. Шаг 7: Демонстрация готового проекта Готовимся к демонстрации проекта перед аудиторией. Это может быть презентация перед преподавателями, коллегами или потенциальными инвесторами. Подготовка к защите:Репетиция презентации.Подготовка ответов на возможные вопросы.Оформление раздаточных материалов. Демонстрация:Презентация проекта.Ответы на вопросы аудитории.Получение обратной связи и предложений по улучшению. Выполняя все эти шаги, ваша команда сможет успешно разработать и представить качественный проект, соответствующий всем установленным требованиям.

Р7	Практическая работа	ОПК-7-З1	Практическое занятие 2. Основы проектирования программного обеспечения
			
			Задание:
			
			Создать диаграмму классов и последовательность для небольшого проекта.
			
			Шаги выполнения:
			
			Шаг 1: Выбор темы проекта
			
			Выберите тему для своего проекта. Например, система управления библиотекой.
			
			Шаг 2: Описание основных сущностей системы
			
			Опишите основные сущности системы (книги, читатели, сотрудники библиотеки).
			
			Книга: Название, Автор, Год издания, ISBN, Количество экземпляров.
			
			Читатель: Имя, Фамилия, Дата рождения, Адрес, Телефон.
			
			Сотрудник библиотеки: Имя, Фамилия, Должность, Логин, Пароль.
			
			Шаг 3: Рисование диаграммы классов
			
			Нарисуйте диаграмму классов, отражающую отношения между этими сущностями.
			
			Пример диаграммы классов:
			
			@startuml
			
			class Book {
			
			- title: String
			
			- author: String
			
			- yearOfPublication: int
			
			- isbn: String
			
			- numberOfCopies: int
			
			}
			
			class Reader {
			
			- firstName: String
			
			- lastName: String
			
			- dateOfBirth: Date
			
			- address: String
			
			- phoneNumber: String
			
			}

			class LibraryEmployee {
			- firstName: String
			- lastName: String
			- position: String
			- login: String
			- password: String
			}
			
			Book "1" -- "*" Reader : Borrowed by
			Reader "1" -- "*" Book : Borrows
			LibraryEmployee "1" -- "*" Book : Manages
			@enduml
			Шаг 4: Разработка сценариев использования системы
			Разработайте несколько сценариев использования системы (например, выдача книги читателю).
			Пример сценария:
			1. Читатель приходит в библиотеку и просит выдать книгу.
			2. Сотрудник библиотеки проверяет наличие книги в каталоге.
			3. Если книга доступна, сотрудник выдает ее читателю и фиксирует дату возврата.
			4. Если книга недоступна, сотрудник предлагает альтернативные варианты или записывает запрос на резервирование книги.
			Шаг 5: Рисование диаграммы последовательности
			Для каждого сценария нарисуйте диаграмму последовательности, показывающую взаимодействие объектов.
			Пример диаграммы последовательности:
			@startuml
			actor Reader
			participant LibrarySystem
			participant LibraryEmployee
			participant Book
			
			Reader -> LibrarySystem : Request book
			activate LibrarySystem
			
			LibrarySystem -> LibraryEmployee : Check availability

			activate LibraryEmployee LibraryEmployee -> Book : Get status activate Book Book --> LibraryEmployee : Return status deactivate Book alt Available LibraryEmployee -> Reader : Issue book deactivate LibraryEmployee else Not available LibraryEmployee -> Reader : Offer alternatives or reserve deactivate LibraryEmployee end deactivate LibrarySystem @enduml Шаг 6: Представление результатов работы Представьте результаты своей работы и объясните, почему вы выбрали именно такие классы и взаимодействия. Пример объяснения: Мы создали диаграмму классов, которая отражает основные сущности системы управления библиотекой: Книга, Читатель и Сотрудник библиотеки. Между ними установлены ассоциации, которые показывают, как эти объекты связаны друг с другом. Например, книга может быть взята несколькими читателями, а сотрудник библиотеки управляет книгами. Диаграмма последовательности иллюстрирует процесс выдачи книги читателю. Сначала читатель отправляет запрос на получение книги через систему библиотеки. Затем система передает запрос сотруднику библиотеки, который проверяет доступность книги. Если книга доступна, сотрудник выдает ее читателю, иначе предлагает альтернативные варианты или резервирует книгу. Эта структура классов и взаимодействий позволяет эффективно моделировать процессы, происходящие в библиотеке, и обеспечивает гибкость и расширяемость системы.

Р8	Лабораторная работа	ОПК-7-З1	Задание: Разработать простой проект с использованием Kanban-доски. Шаги выполнения: Шаг 1: Регистрация на платформе Trello Перейти на сайт Trello и зарегистрироваться, если у вас еще нет аккаунта.Войти в систему, используя созданные учетные данные. Шаг 2: Создание нового проекта Нажмите кнопку «+» в правом верхнем углу экрана и выберите «Создать новую доску».Назовите доску именем вашего проекта, например, «Simple Project Management System».Выберите видимость доски («Public» или «Private») и нажмите «Create Board». Шаг 3: Определение этапов разработки На созданной доске добавьте колонки, соответствующие различным этапам разработки:«Backlog»«To Do»«In Progress»«Testing»«Done» Эти колонки будут представлять собой этапы разработки вашего проекта. Шаг 4: Создание карточек для задач В колонке «Backlog» начните добавлять карточки для задач, которые нужно выполнить в рамках проекта. Например: «Определить требования к проекту» «Спроектировать архитектуру системы» «Реализовать функционал регистрации пользователей» «Провести тестирование функционала» Для каждой задачи добавьте описание, срок выполнения и назначьте ответственного участника команды. Шаг 5: Перемещение карточек между этапами Когда задача готова к выполнению, переместите карточку из колонки «Backlog» в колонку «To Do».После того как кто-то начал работать над задачей, переместите её в колонку «In Progress».Когда задача завершена, но требует тестирования, переместите её в колонку «Testing».Если задача прошла тестирование и готова к релизу, переместите её в колонку «Done». Шаг 6: Отчет о выполнении Просмотрите историю выполненных задач и оцените эффективность использования Kanban-доски. Подготовьте краткий отчет, в котором отразите следующие моменты: Какую роль сыграла Kanban-доска в организации работы над проектом? Какие трудности возникли при использовании этого инструмента? Какие улучшения можно внести в процесс работы с Kanban-досками? Пример отчета: Kanban-доска оказалась очень полезным инструментом для организации работы нашей команды. Она позволила нам наглядно видеть текущий статус всех задач, а также легко отслеживать прогресс выполнения проекта. Благодаря четкому разделению задач на разные стадии, каждый член команды знал, чем ему следует заниматься в данный момент. Опишите что ещё полезного есть в этом инструменте. Однако, мы столкнулись с некоторыми трудностями. Иногда было сложно определить, когда задачу стоит считать выполненной и

			перемещать её в следующую колонку. Также иногда возникали ситуации, когда одна и та же задача одновременно находилась в разных стадиях у разных членов команды. Опишите с чем столкнулись вы. Чтобы улучшить процесс работы с Kanban-досками, мы предлагаем ввести дополнительные правила для определения статуса задач и использовать комментарии внутри карточек для уточнения деталей выполнения. Предложите их. Таким образом, выполнение этой лабораторной работы позволило нам освоить основы работы с Kanban-досками и понять, как этот инструмент может помочь в управлении проектами.

Р9	Практическая работа	ОПК-7-31	Тема: Версионность и управление конфигурацией Задание: Использовать Git для управления проектом. Шаги выполнения: Шаг 1: Регистрация на GitHub Перейдите на сайт GitHub и зарегистрируйтесь, если у вас ещё нет аккаунта.Войдите в систему, используя созданные учётные данные. Шаг 2: Создание нового репозитория Нажмите кнопку «+» в правом верхнем углу экрана и выберите «Создать новый репозиторий».Назовите репозиторий именем вашего проекта, например, «laboratory-work-git`.Выберите видимость репозитория («Public» или «Private») и нажмите «Create repository». Шаг 3: Инициализация репозитория Склонируйте созданный репозиторий на локальный компьютер: git clone https://github.com/Ваш_аккаунт_GitHub/laboratory-work-git.git 2. Перейдите в директорию проекта: cd laboratory-work-git 3. Инициализируйте Git-репозиторий: git init Шаг 4: Добавление файлов в репозиторий 1. Создайте файл README.md и добавьте туда описание проекта. 2. Добавьте файл .gitignore, чтобы исключить ненужные файлы из репозитория. 3. Добавьте файлы в репозиторий: git add README.md .gitignore 4. Сделайте первый коммит: git commit -m "Initial commit" 5. Свяжите локальный репозиторий с удалённым: git remote add origin https://github.com/Ваш_аккаунт_GitHub/laboratory-work-git.git 6. Загрузите изменения на удалённый репозиторий: git push -u origin master Шаг 5: Создание новой ветки и работа с ней 1. Создайте новую ветку для внесения изменений и переключитесь на неё: git checkout -b feature

			2. Сделайте изменения в коде, закоммитите их и объедините ветку с основной веткой (master или main): git add . git commit -m "Добавлены новые функции" git checkout master git merge feature 3. Решите конфликт, если он возникнет при слиянии веток: git mergetool --abort git reset --hard HEAD~1 git cherry-pick -x <commit_hash> 4. Удалите ветку feature после успешного слияния: git branch -d feature Шаг 6: Запуск приложения Запустите приложение, выполнив команду: python app.py Откройте браузер и перейдите по адресу http://localhost:5000/, чтобы увидеть работающее приложение. Заключение Использование Git для управления проектом — это мощный инструмент, который позволяет хранить историю изменений, отслеживать версии кода и работать над одним проектом совместно с командой. Освоение Git — важный навык для любого разработчика, так как он помогает поддерживать порядок в кодовой базе и минимизировать риски потери данных. Перенести в Умный редактор

Р10	Лабораторная работа	УК-2-З1	Основы проектирования программного обеспечения
			
			Задание: Создать диаграмму классов и последовательность для небольшого проекта.
			
			Шаги выполнения:
			
			Шаг 1: Выбор темы проекта. Выберите тему для своего проекта. Например, система управления библиотекой.
			
			Шаг 2: Описание основных сущностей системы
			
			Опишите основные сущности системы (книги, читатели, сотрудники библиотеки).
			
			Книга: Название, Автор, Год издания, ISBN, Количество экземпляров.
			
			Читатель: Имя, Фамилия, Дата рождения, Адрес, Телефон.
			
			Сотрудник библиотеки: Имя, Фамилия, Должность, Логин, Пароль.
			
			Шаг 3: Рисование диаграммы классов
			
			Нарисуйте диаграмму классов, отражающую отношения между этими сущностями.
			
			Пример диаграммы классов:
			
			@startuml
			
			class Book {
			
			- title: String
			
			- author: String
			
			- yearOfPublication: int
			
			- isbn: String
			
			- numberOfCopies: int
			
			}
			
			class Reader {
			
			- firstName: String
			
			- lastName: String
			
			- dateOfBirth: Date
			
			- address: String
			
			- phoneNumber: String
			
			}
			
			class LibraryEmployee {
			
			- firstName: String
			
			- lastName: String

			- position: String - login: String - password: String } Book "1" -- "*" Reader : Borrowed by Reader "1" -- "*" Book : Borrows LibraryEmployee "1" -- "*" Book : Manages @enduml Шаг 4: Разработка сценариев использования системы Разработайте несколько сценариев использования системы (например, выдача книги читателю). Пример сценария: 1. Читатель приходит в библиотеку и просит выдать книгу. 2. Сотрудник библиотеки проверяет наличие книги в каталоге. 3. Если книга доступна, сотрудник выдает ее читателю и фиксирует дату возврата. 4. Если книга недоступна, сотрудник предлагает альтернативные варианты или записывает запрос на резервирование книги. Шаг 5: Рисование диаграммы последовательности Для каждого сценария нарисуйте диаграмму последовательности, показывающую взаимодействие объектов. Пример диаграммы последовательности: @startuml actor Reader participant LibrarySystem participant LibraryEmployee participant Book Reader -> LibrarySystem : Request book activate LibrarySystem LibrarySystem -> LibraryEmployee : Check availability activate LibraryEmployee LibraryEmployee -> Book : Get status

			activate Book Book --> LibraryEmployee : Return status deactivate Book alt Available LibraryEmployee -> Reader : Issue book deactivate LibraryEmployee else Not available LibraryEmployee -> Reader : Offer alternatives or reserve deactivate LibraryEmployee end deactivate LibrarySystem @enduml Шаг 6: Представление результатов работы Представьте результаты своей работы и объясните, почему вы выбрали именно такие классы и взаимодействия. Пример объяснения: Мы создали диаграмму классов, которая отражает основные сущности системы управления библиотекой: Книга, Читатель и Сотрудник библиотеки. Между ними установлены ассоциации, которые показывают, как эти объекты связаны друг с другом. Например, книга может быть взята несколькими читателями, а сотрудник библиотеки управляет книгами. Диаграмма последовательности иллюстрирует процесс выдачи книги читателю. Сначала читатель отправляет запрос на получение книги через систему библиотеки. Затем система передает запрос сотруднику библиотеки, который проверяет доступность книги. Если книга доступна, сотрудник выдает ее читателю, иначе предлагает альтернативные варианты или резервирует книгу. Эта структура классов и взаимодействий позволяет эффективно моделировать процессы, происходящие в библиотеке, и обеспечивает гибкость и расширяемость системы.

Р11	Лабораторная работа	УК-2-З1	Тестирование программного обеспечения
			
			Задание:
			
			Написать юнит-тесты для существующего кода.
			
			Шаги выполнения:
			
			Шаг 1: Выбор фрагмента кода для тестирования
			
			Найдите или создайте небольшой фрагмент кода, который необходимо протестировать. Например, функцию для вычисления суммы двух чисел.
			
			def add(a, b):
			
			"""Суммирует два числа."""
			
			return a + b
			
			Шаг 2: Установка и настройка среды для написания тестов
			
			Установите и настройте среду для написания тестов. Используйте подходящий инструмент, такой как JUnit для Java или PyTest для Python.
			
			Для примера будем использовать PyTest в Python.
			
			1. Установите PyTest:
			
			pip install pytest
			
			2. Создайте файл test_add.py, где будете писать тесты.
			
			Шаг 3: Написание тестов для различных случаев поведения функции
			
			Напишите тесты для различных случаев поведения функции: правильных входных данных, неправильных входных данных, граничных условий.
			
			Пример теста для функции add():
			
			import pytest
			
			@pytest.mark.parametrize("a, b, expected", [
			
			(1, 2, 3),
			
			(-1, 1, 0),
			
			(100, -50, 50),
			
			])
			
			def test_add(a, b, expected):
			
			assert add(a, b) == expected
			
			Здесь мы используем параметризацию тестов с помощью pytest.mark.parametrize. Это позволяет проверять одну и ту же функцию с разными наборами входных данных.
			
			Также добавим тест для проверки исключений:
			
			def test_add_type_error():

			with pytest.raises(TypeError): add("1", 2) Этот тест проверяет, что функция add() вызывает исключение TypeError, если передать ей строки вместо чисел. Шаг 4: Запуск тестов и проверка их прохождения Запустите тесты и убедитесь, что они проходят успешно. 1. Запустите тесты: pytest 2. Если тесты прошли успешно, вы увидите результат вроде следующего: ============================= test session starts ============================== platform darwin -- Python 3.x.y, pytest-6.x.y, py-1.x.y, pluggy-0.x.y rootdir: /path/to/your/project collected 2 items test_add.py .. [100%] ============================ 2 passed in 0.01s ============================ Это означает, что оба теста прошли успешно. Шаг 5: Документирование тестов и предоставление отчета Документируйте свои тесты и предоставьте отчет о проделанной работе. Пример документации: """ Тестирование функции add() с использованием PyTest. Тесты включают проверку правильности суммирования целых чисел, а также обработку неверных типов данных. Все тесты прошли успешно. """ Отчет может включать: · Описание задачи. · Используемые инструменты.

			· Примеры тестов. · Результаты запуска тестов. · Выводы и предложения по улучшению тестирования. Заключение Выполнение этой лабораторной работы позволит вам освоить основы написания юнит-тестов, что является важным этапом в разработке качественного программного обеспечения. Тестирование помогает выявлять ошибки на ранних этапах разработки и обеспечивать надежность и стабильность кода.

Р12	Практическая работа	УК-2-З1	Тема: Автоматизация сборки и развертывания Задание: Настроить CI/CD пайплайн для учебного проекта. Шаги выполнения: Шаг 1: Выбор платформы CI/CD Выберите платформу для CI/CD, такую как Jenkins, TravisCI или Docker. Рассмотрим установку и настройку Jenkins.Скачайте и установите Jenkins: wget -qO -O jenkins.war java -jar jenkins.war --httpPort=8080 2. Откройте браузер по адресу http://localhost:8080 и войдите в систему, используя admin:admin для первого входа. 3. Создайте аккаунт администратора и измените пароль. Шаг 2: Настройка CI/CD пайплайна Настройте автоматизацию сборки проекта: компиляция, упаковка, запуск тестов. 1. Создайте новый проект в Jenkins. 2. Настройте источники исходников: укажите URL репозитория GitHub. 3. Настройте выполнение шагов сборки: компиляция, тестирование, упаковка артефактов. 4. Настройте автоматическое развёртывание на тестовом сервере после успешной сборки. Шаг 3: Запуск тестов Запустите тесты, которые вы написали для проекта. 1. В настройках проекта добавьте шаги для запуска тестов. 2. Проверьте успешное прохождение тестов. Шаг 4: Настройка CI/CD пайплайна Убедитесь, что ваш пайплайн работает корректно, выполнив несколько сборок и развёрток. 1. Проверьте журналы сборок и развёрток. 2. Исправьте ошибки, если они возникли. Заключение Автоматизация сборки и развёртывания с использованием инструментов CI/CD существенно упрощает жизнь разработчикам, позволяя сократить рутинные задачи и уменьшить вероятность ошибок при сборке и развёртывании. Настройка CI/CD пайплайна позволяет интегрировать тестирование и автоматизацию выпуска обновлений, что повышает надёжность и устойчивость разрабатываемых решений.

			Это означает, что вы должны запустить те тесты, которые вы предварительно подготовили для вашего проекта. Обычно это делается с помощью специальных инструментов, таких как JUnit, PyTest, Nose или других, в зависимости от языка программирования, который вы используете. Например, если вы используете Python, то для запуска тестов с использованием PyTest достаточно выполнить команду: pytest Если вы используете другой язык программирования, например, Java, тогда вам потребуется запустить тесты с помощью соответствующего инструмента, например, JUnit: mvn clean test Эти команды запустят тесты, которые вы подготовили, и покажут результаты их выполнения. Перенести в Умный редактор

Р13	Практическая работа	УК-2-З1	Тема: Безопасность программного обеспечения Задание: Анализировать безопасность программного обеспечения, находить уязвимые места и устранять их. Шаги выполнения: Шаг 1: Идентификация угроз Определите основные угрозы безопасности, такие как SQL-инъекция, XSS, CSRF и др.Найдите слабые места в коде, такие как отсутствие проверки ввода данных, недостаточная обработка ошибок и др. Шаг 2: Разработка плана по устранению уязвимостей Создайте план по устранению выявленных уязвимостей.Определите методы защиты, такие как шифрование данных, аутентификация и авторизация. Шаг 3: Реализация мер безопасности Внедрите механизмы защиты, такие как фильтрация входных данных, проверка на SQL-инъекции, защиту от XSS и CSRF.Обеспечьте корректную обработку ошибок и логирование событий безопасности. Шаг 4: Тестирование безопасности Проведите тестирование безопасности на реальных данных.Используйте специализированные инструменты для поиска уязвимостей, такие как OWASP Zulu, Nessus и др.Проверьте, что меры безопасности эффективны и надежно защищают систему. Шаг 5: Документирование и отчетность Подготовьте документацию по мерам безопасности, принятым в проекте.Представьте отчет о проведенных работах руководству или заказчику. Пример выполнения: Шаг 1: Идентификация угроз Допустим, что в вашем проекте есть риск SQL-инъекции. В этом случае:Проблема: Непроверенный ввод данных в SQL-запросы.Решение: Фильтрация и валидизация данных перед их использованием в SQL-запросах. Шаг 2: План устранения уязвимостей Метод защиты: Использование ORM для работы с базой данных.Процедура: Валидатор проверяет вводимые данные перед их передачей в SQL. Шаг 3: Реализация мер безопасности Шифрование данных: Использование SSL/TLS для передачи секретных данных.Аутентификация: Введение механизмов аутентификации и авторизации.Логирование: Сохранение информации о попытках несанкционированного доступа. Шаг 4: Тестирование безопасности Инструмент: Использование сканера безопасности, например, Acunet Security Scanner.Цель: Найти оставшиеся уязвимости и

			подтвердить эффективность принятых мер. Шаг 5: Документирование и отчетность Документация: Описание методов защиты и используемых технологий.Отчет: Презентация результатов работы руководителю или заказчику. Заключение Лабораторная работа по безопасности программного обеспечения помогает разработать безопасные и надежные системы, защищающие данные пользователей и компании от возможных атак. Это важный аспект современной разработки, так как кибератаки становятся все более частыми и разрушительными.

Р14	Практическое занятие	УК-2-З1	Архитектурные стили и шаблоны проектирования Шаги выполнения: Шаг 1: Выбор темы проекта Выбралось создание системы управления библиотекой. Шаг 2: Описание основных сущностей системы Основные сущности системы управления библиотекой включают:Книги — коллекция книг, доступных для выдачи читателям.Читатели — пользователи библиотеки, которые регистрируются и получают доступ к ресурсам библиотеки.Сотрудники библиотеки — персонал, ответственный за управление библиотекой, включая администраторов, библиотекарей и помощников. Шаг 3: Проектирование диаграмм классов и последовательностей Диаграма классов:Создание схемы классов, отражающей структуру системы управления библиотекой. Например, класс Book представляет собой сущность книги, а класс Librarian — сотрудника библиотеки, который занимается управлением книг.Диаграма последовательностей:Описание потоков взаимодействия между объектами системы, включая чтение книг, возврат книг и взаимодействие с библиотекой через интерфейс. Шаг 4: Реализация архитектуры На основе спроектированных диаграмм создается реализация компонентов системы управления библиотекой. Например, библиотека реализует функционалку выдачи и возврата книг, а также регистрацию читателей. Шаг 5: Презентация и защита проекта Студенты представляют проект руководству или комиссии, демонстрируя разработанную систему управления библиотекой. Включаются демонстрация интерфейса, описания функциональности и разборка архитектуры. Пример выполнения: Шаг 1: Выбор темы проекта Выбрано создание системы управления библиотекой, включающая функционалку выдачи и возврата книг, а также учетную систему читателей. Шаг 2: Описание основных сущностей системы Основные сущности системы управления библиотекой:Книга — сущность, хранящаяся в библиотеке, состоящая из названия, автора, аннотации и количества экземпляров.Читатель — пользователь библиотеки, который регистрируется и получает доступ к библиотечным ресурсам.Библиотекарь — сотрудник библиотеки, отвечающий за выдачу и прием книг, а также ведение учёта. Шаг 3: Проектирование диаграмм классов и последовательностей Пример создания диаграммы классов: Диаграма классов: plantuml

			@startuml
			класс Book {
			- **Название**:
			- **Book**
			- **Атрибуты**:
			- **Title**:
			- **Строка**
			- **Методы**:
			- **addBook**:
			- **Добавление книги в библиотеку**
			- **Ассоциации**:
			- **HasMany**:
			- **Книги**
			- **Композиция**:
			- **Монолитная архитектура**
			Диаграма последовательностей:
			plantuml
			@startuml
			сценарий использования:
			- **Акторы**:
			- **Читатель**
			- **Действие**:
			- **Читатель** обращается к библиотекарю с просьбой выдать книгу.
			- **Последовательность**:
			- **Библиотекарь** находит книгу и возвращает её читателю.
			- **Итерация**:
			- **Читатель** возвращает книгу обратно, если завершил чтение.
			- **Завершение**:
			- **Система** сохраняет информацию о статусе книги и учителю, что книга возвращена.
			Шаг 4: Реализация архитектуры
			Класс Book:
			python
			class Book:

```
    def __init__(self, title, author, annotation):
        self.title = title
        self.author = author
        self.annotation = annotation
        self.status = "Доступно"

    def addBook(self, library):
        library.addBook(self)
        self.status = "В выдаче"

    def returnBook(self, reader):
        self.status = "Возвратно"
        library.returnBook(self, reader)
```

Класс Librarian:

python

```
class Librarian:
    def manageBooks(self, library):
        books = library.getAllBooks()
        for book in books:
            if book.status == "Доступно":
                book.status = "Готово к выдаче"

    def issueBook(self, reader, book):
        book.status = "Выдана"
        reader.notifyAboutReturn(book)

    def takeBackBook(self, reader, book):
        book.status = "Возвратно"
        reader.notifyAboutReturn(book)

    def checkAvailability(self, book):
        if book.status == "Доступно":
            return True
        else:
```

			return False
			Класс Reader:
			python
			class Reader:
			def register(self, library):
			self.library = library
			self.registerInLibrary()
			def borrowBook(self, book):
			self.borrowBookFromLibrary(book)
			def returnBook(self, book):
			self.returnBookToLibrary(book)
			def notifyAboutReturn(self, book):
			self.notifyReaderAboutReturn(book)
			def getAvailableBooks(self):
			books = self.library.getAvailableBooks()
			return books
			#### Шаг 5: Презентация и защита проекта
			Студенты демонстрируют проект руководству или комиссии, показывая функциональность системы управления библиотекой. Руководитель оценивает проект на основе критериев качества и полноты выполнения задания.
			### Пример демонстрации:
			Сценарий использования системы управления библиотекой:
			1. **Читатель** регистрируется в системе и выбирает книгу для чтения.
			2. **Библиотекарь** проверяет наличие книги и её статус, затем выдаёт её читателю.
			3. **Читатель** начинает читать книгу, а после завершения возвращается её в библиотеку.
			4. **Библиотека** фиксирует статус книги как "Возвратно" и

			уведомляет студента о необходимости вернуть её. 5. **Студент** возвращает книгу, и система снова переходит в режим ожидания выдачи. 6. **Руководитель** оценивает работу системы управления библиотекой и даёт оценку проекту.

5.3. Оценочные материалы, используемые для экзамена (описание билетов, тестов и т.п.)

Формой промежуточной аттестации по дисциплине является экзамен.
Ниже представлен образец билета для экзамена, проводимого в устной форме.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ
«МИСИС»
НОВОТРОИЦКИЙ ФИЛИАЛ

Кафедра математики и естествознания

БИЛЕТ К ЭКЗАМЕНУ № 0

Дисциплина: «Программная инженерия»
Направление: 09.03.03 «Прикладная информатика»
Форма обучения: очная
Форма проведения экзамена: устная

1. Жизненный цикл программного продукта. Процесс, действие, задача жизненного цикла. Фазы (этапы) жизненного цикла и их связь с процессами.
2. Тестирование программного обеспечения. Цели и задачи тестирования.

Составил доцент: _____ Р.Р. Абдулвелеева

Зав. кафедрой МиЕ: _____ А.В. Швалёва

«01» сентября 2024 г.

Образец заданий для экзамена, проводимого дистанционно :

1. Программная инженерия – это
а) Совокупность инструментальных средств и методов, предназначенных для создания качественного программного обеспечения.
б) Совокупность инструментальных средств, предназначенных для создания качественного программного обеспечения.
в) Совокупность навыков, инструментальных средств и методов, предназна¬ченных для создания качественного программного обеспечения.
г) Наука, изучающая построение программных систем
д) Правила проектирования систем со сложной архитектурой

2. Программная инженерия занимается
а) Вопросами оптимизации кода
б) Вопросами разработки новых алгоритмов обработки данных
в) Вопросами эффективной разработки программного обеспечения
г) Применением средств быстрой разработки программного обеспечения
д) Применением средств автоматизированного тестирования программного обеспечения

3. Стадии разработки программных систем, общие формы алгоритмов и схем, описывающих эти системы, регламентируются
а) Стандартами ЕСПД
б) Пунктами ТЗ
в) Никак не регламентируются
г) Эксплуатационными документами

д) Спецификацией ПС

4. Псевдокод представляет собой
а) Частично формализованный язык для представления описаний метода пошаговой детализации
б) Язык, использующий конструкции структурного программирования
в) Язык программирования высокого уровня
г) Язык с неформальными фрагментами на естественном языке для представ¬ления обобщенных операторов и условий
д) Формальная запись конструкций языка программирования Фортран

5. Укажите основные процессы жизненного цикла по ГОСТ Р ИСО/МЭК 12207-99. «Информационная технология. Процессы жизненного цикла программных средств»
а) Процесс заказа
б) Процесс документирования
в) Процесс разработки
г) Процесс управления
д) Процесс сопровождения

6. Проблемы, решаемые конфигурационным управлением
а) Работа в команде
б) Одновременная модификация
в) Ограниченное уведомление
г) Управление пользователями
д) Множество версий

7. Этапы последовательной разработки ("водопад")
а) Снятие с эксплуатации
б) Тестирование
в) Анализ требований
г) Проектирование
д) Системный анализ
е) Использование и сопровождение

8. Этапы итеративного цикла разработки
а) Тестирование
б) бизнес-моделирование
в) Реализация
г) Анализ и проектирование
д) Требования

9. Порядок разработки программного модуля
а) Программирование (кодирование) модуля
б) Шлифовка текста модуля
в) Изучение и проверка спецификации модуля, выбор языка программирования
г) Выбор алгоритма и структуры данных
д) Компиляция модуля
е) Проверка модуля

10. Содержание технического задания на программный продукт в порядке следования
а) Технико-экономические показатели
б) Назначение разработки
в) Стадии и этапы разработки
г) Введение
д) Требования к программной документации
е) Порядок контроля и приёмки
ж) Основания для разработки
з) Требования к программе или программному изделию

11. Чем определяется сложность ПО?
а) количеством пользователей
б) объемом обрабатываемых данных
в) требованиями по быстродействию

12. В чем заключается согласованность ПО?
а) в том, что ПО основывается на объективных посылках
б) в том, что ПО должно быть согласовано с большим количеством интерфейсов
в) в согласованности заказчика и исполнителя

13. К какому типу проектов относятся проекты по разработке ПО:
а) и к творческим, и к промышленным проектам
б) к промышленным проектам
в) к творческим проектам

14. На каком уровне процессы в полной мере существуют лишь в рамках отдельных проектов:
а) на начальном уровне
б) на управляемом уровне
в) на оптимизирующемся уровне

15. Какой из участников создания модели при описании системы не несет ответственности за качество моделирования:
а) автор
б) эксперт
в) читатель

16. Какой вопрос решается в сфере программной инженерии:
а) вопросы создания компьютерных программ и/или программного обеспечения
б) бизнес-реинжиниринг
в) вопрос поддержки жизненного цикла разработки ПО

17. Как называется процесс разбиения одной сложной задачи на несколько простых подзадач?
а) абстракция
б) декомпозиция
в) реинжиниринг
г) верификация

18. Интерфейс пользователя — это
а) набор методов взаимодействия компьютерной программы и пользователя этой программы
б) набор методов для взаимодействия между программами
в) способ взаимодействия между объектами
г) прежде всего, набор правил

19. Что из приведенного является критериями оценки удобства интерфейсов?
а) скорость обучения
б) адаптация к стилю работы пользователя
в) абстракция
г) реинжиниринг

20. Легкость применения программного обеспечения это
а) характеристики ПО, позволяющие минимизировать усилия пользователя по
б) подготовке исходных данных, применению ПО
в) отношение уровня услуг, предоставляемых ПО пользователю при заданных условиях, к объему используемых ресурсов
г) характеристики ПО, позволяющие минимизировать усилия по внесению изменений для устранения в нем ошибок и по его модификации
д) способность ПО выполнять набор функций, которые удовлетворяют потребности пользователей

21. Мобильность программного обеспечения это
а) способность ПО выполнять набор функций, которые удовлетворяют потребности пользователей
б) способность ПО безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени
в) способность ПО быть перенесенным из одной среды (аппаратного / программного) в другое
г) отношение уровня услуг, предоставляемых ПО пользователю при заданных условиях, к объему используемых ресурсов

22. Под отладкой программного средства понимают
а) Деятельность, направленная на обнаружение и исправление ошибок в ПС с использованием процессов выполнения его программ
б) Процесс выполнения его программ на некотором наборе данных, для кото¬рого заранее известен результат применения или известны правила поведения этих программ
в) Отладка = Тестирование + Поиск ошибок + Редактирование
г) Процесс поиска и исправления ошибок (без тестирования)
д) Настройка ПС на требуемые наборы данных

23. Устойчивость программного обеспечения — это
а) свойство, характеризующее способность ПС завершать автоматически корректное функционирование ПК, несмотря на неправильные (ошибочные) входные данные
б) свойство, способное противостоять преднамеренным или непреднамеренным деструктивным действиям пользователя
в) свойство, характеризующее способность ПС продолжать корректное функционирование, несмотря на неправильные (ошибочные) входные данные
г) отношение уровня услуг, предоставляемых ПО пользователю при заданных условиях, к объему используемых ресурсов

24. UML — это:
а) язык программирования, имеющий синтаксис схожий с С ++
б) унифицированный язык визуального моделирования, использует нотацию диаграмм
в) набор стандартов и спецификаций качества программного обеспечения
г) адрес в сети интернет

25. При конструировании программного обеспечения на этапе разработки или выбора алгоритма решения реализуется следующее
а) архитектурные обработки программы
б) выбор языка программирования
в) совершенствование программы
г) синтаксические отладки

26. Проектирование ПО в основном рассматривается как
а) архитектурное проектирование
б) коммуникационные методы
в) детальные методы
г) совершенствование программы

27. На этапе тестирования пользователь выполняет следующее
а) синтаксические отладки
б) выбор тестов и метода тестирования
в) определение формы выдачи результатов
г) архитектурные обработки программы

28. Что из приведенного не является одним из методов проектирования программного обеспечения?
а) структурное программирование
б) объектно-ориентированное программирование
в) алгебраическое программирования
г) синтаксические отладки

29. В обсуждении требований на систему принимают участие:
а) аналитики и разработчики будущей системы
б) представители заказчика из нескольких профессиональных групп
в) специалисты, производящие инсталляцию системы

30. Спецификация требований к ПО — это:
а) процесс проверки правильности спецификации требований на их соответствие, непротиворечивость, полноту и выполнимость, а также на соответствие стандартам
б) формализованное описание функциональных, нефункциональных и системных требований, требований к характеристикам качества, а также к структуре ПО, принципам взаимодействия с другими компонентами, алгоритмам и структуре данных системы
в) проверка требований, для того чтобы убедиться, что они определяют именно данную систему

31. Объект предметной области — это:
а) образ с поведением, которое обусловлено его характеристиками и взаимоотношениями с другими объектами предметной области
б) конкретный образ с поведением, которое обусловлено его характеристиками и взаимоотношениями с другими объектами предметной области
в) значение некоторой абстрактной сущности предметной области

32. Реинженерия (reengineering) — это:
а) внесение изменений в компоненты или интерфейсы (добавление, расширение и т. д.), добавление экземпляров компонентов, новых функций или системных сервисов
б) эволюция программы путем ее изменения в целях повышения удобства ее эксплуатации, сопровождения или изменения ее функций
в) полная переделка компонентов, а иногда и перепрограммирование всей системы

33. Функциональный аудит конфигурации проводится:
а) для подтверждения информации о текущем статусе идентифицированных объектов конфигурационного управления, предложенных изменениях, а также о выявленных дефектах и отклонениях
б) для подтверждения взаимного соответствия документации и фактической конфигурации продукта
в) (Правильный ответ) для подтверждения соответствия фактических характеристик конфигурации продукта требованиям заказчика

34. После получения новой версии системы заказчику передаются:
а) версия
б) инструменты управления версиями для самостоятельного внесения изменений при сопровождении системы
в) документация
г) отчеты о выявленных ошибках
д) конфигурация

35. Какую роль выполняет менеджер в процессе работы над ошибками:
а) нахождение ошибок
б) контроль хода проекта
в) исправление ошибок

36. Какой из участников создания модели при описании системы не несет ответственности за качество моделирования:
а) автор
б) эксперт
в) читатель

37. С какой ролью можно совмещать разработку:
а) архитектура
б) управление продуктом
в) тестирование

38. На каком уровне зрелости осуществляется анализ причин возникновения проблем и предотвращение их появления в будущем:
а) на уровне зрелости 3
б) на уровне зрелости 4
в) на уровне зрелости 5

5.4. Методика оценки освоения дисциплины (модуля, практики. НИР)

Критерии оценки ответов на экзамене, проводимом в устной форме
Оценка «Отлично» ставится, если
- на теоретические вопросы даны развернутые ответы, при необходимости изложен математический аппарат (формулы, графики и т.д.) приведены соответствующие схемы, таблицы, рисунки и т.д., правильно решена задача
- обучающийся хорошо ориентируется в материале, владеет терминологией, приводит примеры, обосновывает, анализирует, высказывает свою точку зрения по анализируемым явлениям и процессам, правильно применяет полученные знания при решении практических задач. Ответы излагаются свободно, уверенно без использования листа устного опроса
Оценка «Хорошо» ставится, если
- на теоретические вопросы даны полные ответы, но имела место неточность в определении каких-либо понятий, явлений и т.д. Задача решена.
- обучающийся ориентируется в материале хорошо, но допускает ошибки при формулировке, описании отдельных категорий
Оценка «Удовлетворительно» ставится, если
- на теоретические вопросы даны общие неполные ответы
- обучающийся слабо ориентируется в материале, не может решать задачи, не может привести пример, не может анализировать и обосновывать
Оценка «Неудовлетворительно» ставится, если
- не решена задача и правильный ответ дан на один вопрос (либо ни на один)
- обучающийся в материале дисциплины практически не ориентируется, т.е. не может дать даже общих сведений по вопросу.

Критерии оценки ответов на экзамене, проводимом в дистанционной форме
$90 \leq$ Процент верных ответов ≤ 100 - отлично
$75 \leq$ Процент верных ответов < 90 - хорошо
$60 \leq$ Процент верных ответов < 75 – удовлетворительно

Критерии оценки выполнения курсовой работы:
1. Теоретические сведения изложены в достаточном объеме, четко и последовательно
2. Использованы выводы (позиции, мнения и др.) известных ученых, профессионалов
3. Исследуются и сравниваются разные подходы, методики, приводятся собственные суждения и выводы

4. Описана актуальность работы и предметная область.
5. Описывается процесс анализа и моделирования предметной области
6. Описываются алгоритмы работы и интерфейс программы
7. Приведен процесс тестирования ПО
8. Текст написан грамотно, стилистически выдержан
9. Текст оформлен в соответствии с требованиями

Работа оценивается на отлично, если:
теоретические сведения изложены в достаточном объеме, четко и последовательно, использованы выводы (позиции, мнения и др.) известных ученых, профессионалов, исследуются и сравниваются разные подходы, методики, приводятся собственные суждения и выводы, имеются примеры, даются ссылки на источники, текст написан грамотно, стилистически выдержан и оформлен в соответствии с требованиями.
Процесс анализа и моделирования предметной области описан полностью, обязательно должен включать в себя требования к ПО; архитектуру ПО, включая три модели: информационную, состояний и процессов; спецификацию ПО; схему БД.
В полном объеме описываются алгоритмы работы и интерфейс программы, приведены все диалоговые окна с подробным описанием функционала каждого элемента интерфейса.
Приведен процесс тестирования ПО, который включает в себя: стратегию тестирования; тест-план; отчет по проведенному тестированию; результаты выполнения автоматизированного теста.
В целом по работе: расставлены ссылки на источники, текст написан грамотно, стилистически выдержан, оформлен в соответствии с требованиями.
Выполнение работы оценивается как хорошее, если она соответствует всем критериям, перечисленным выше, но в работе отсутствует отсутствуют некоторые элементы описания процесса анализа и моделирования предметной области, отсутствует полное описание функционала каждого элемента интерфейса, описание тестирования приведено не полностью.
В целом по работе: расставлены ссылки на источники, текст написан грамотно, стилистически выдержан, оформлен в соответствии с требованиями.
Выполнение работы оценивается как удовлетворительное, если она соответствует всем критериям, перечисленным выше, но в работе отсутствуют некоторые элементы описания процесса анализа и моделирования предметной области, отсутствует описание функционала каждого элемента интерфейса, описание тестирования не приведено. Отсутствует описание актуальности работы и предметной области.
Если работа допущена до защиты с оценкой «отлично», в процессе защиты студент хорошо владеет материалом, не использует при этом опорных конспектов и т.д., с легкостью отвечает на любой вопрос по курсовой работе, то в этом случае студенту за выполнение курсовой работы ставится оценка «отлично», которая и проставляется в зачетную книжку и в ведомость.
В процессе защиты оценка повышаться не может, т.е. если студент допущен до защиты с оценкой «хорошо», «отлично» он уже в любом случае не сможет получить, а вот «удовлетворительно» может – если при защите возникают определенные трудности с ориентацией в материале, ответами на вопросы по курсовой работе.
Если студент совершенно не владеет материалом курсовой работы, то получает «неудовлетворительно».
Если работа не соответствует критериям выполнения курсовой работы, то оценивается неудовлетворительно и до защиты не допускается.

Критерии оценки выполнения расчетно-графической работы:
1. Теоретические сведения изложены в достаточном объеме, четко и последовательно
2. Исследуются и сравниваются разные подходы, методики, приводятся собственные суждения и выводы
3. Приведены основные цели разработки
4. Приведены требования к программному продукту
5. Определены сроки и этапы разработки
6. Регламентирован процесс приемосдаточных мероприятий.
7. Расставлены ссылки на источники
8. Текст написан грамотно, стилистически выдержан
9. Текст оформлен в соответствии с требованиями

- теоретические сведения изложены в достаточном объеме, четко и последовательно, приводятся собственные суждения и выводы, даются ссылки на источники, текст написан грамотно, стилистически выдержан и оформлен в соответствии с требованиями.
- требования к программному продукту приведены в полном объеме, верно определены сроки и этапы разработки, полностью регламентирован процесс приемосдаточных мероприятий. Верно определены основные цели разработки.
В целом по работе: расставлены ссылки на источники, текст написан грамотно, стилистически выдержан, оформлен в соответствии с требованиями.
Выполнение работы оценивается как хорошее, если она соответствует всем критериям, перечисленным выше, но требования к программному продукту, процесс приемосдаточных мероприятий приведены не полностью.
В целом по работе: расставлены ссылки на источники, текст написан грамотно, стилистически выдержан, оформлен в соответствии с требованиями.
Выполнение работы оценивается как удовлетворительное, если она соответствует всем критериям, перечисленным выше, но требования к программному продукту приведены не полностью, отсутствует описание процесса приемосдаточных мероприятий, отсутствуют основные цели разработки.
Если расчетно-графическая работа не соответствует критериям, перечисленным выше, то оценивается

неудовлетворительно.

6. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ

6.1. Рекомендуемая литература

6.1.1. Основная литература

	Авторы, составители	Заглавие	Библиотека	Издательство, год, эл. адрес
Л1.1	Лаврищева Е.М.	Программная инженерия. Парадигмы, технологии и CASE-средства: учебник		М.: Юрайт, 2019,
Л1.2	Антамошкин О.А.	Программная инженерия. Теория и практика: учебник		Красноярск : Сибирский федеральный университет, 2012, http://biblioclub.ru/index.php?page=book&id=363975

6.1.2. Дополнительная литература

	Авторы, составители	Заглавие	Библиотека	Издательство, год, эл. адрес
Л2.1	Абдулаев В.И.	Программная инженерия: учебное пособие		Йошкар-Ола : ПГТУ, 2016, http://biblioclub.ru/index.php?page=book&id=459449
Л2.2	Флоренсов А.Н.	Системное программное обеспечение: учебное пособие		Омск : Издательство ОмГТУ, 2017, http://biblioclub.ru/index.php?page=book&id=493301
Л2.3	Романов, Е. Л.	Программная инженерия : учебное пособие		Новосибирск : Новосибирский государственный технический университет, 2017, https://biblioclub.ru/index.php?page=book&id=573945

6.1.3. Методические разработки

	Авторы, составители	Заглавие	Библиотека	Издательство, год, эл. адрес
Л3.1	В.Т. Николаев, С.В. Купцов, В.Н. Тикменов	Практика программирования в инженерных расчётах: учебное пособие		Москва :Физматлит, 2018, http://biblioclub.ru/index.php?page=book&id=485295
Л3.2	Э.Р. Ипатова, Ю.В. Ипатов	Методологии и технологии системного проектирования информационных систем: учебник		Москва : Издательство «Флинта», 2016, http://biblioclub.ru/index.php?page=book&id=79551
Л3.3	Долженко, А.И.	Технологии командной разработки программного обеспечения информационных систем		Москва : Национальный Открытый Университет «ИНТУИТ», 2016, http://biblioclub.ru/index.php?page=book&id=428801

6.2. Перечень ресурсов информационно-телекоммуникационной сети «Интернет»

Э1	Научная электронная библиотека eLIBRARY	https://www.elibrary.ru/
Э2	LMS Canvas	https://lms.misis.ru
Э3	НФ НИТУ МИСиС	http://nf.misis.ru/
Э4	Университетская библиотека ONLINE	https://biblioclub.ru/

6.3 Перечень программного обеспечения

П.1	Microsoft Office Professional Plus 2013 Russian OLP NL AcademicEdition;
П.2	1С: Предприятие 8.2, комплект для обучения в высших и средних учебных заведениях
П.3	Microsoft Office 2010 Russian Academic OPEN 1 License No Level
П.4	Браузер Google Chrome
П.5	Microsoft Teams
П.6	Python
П.7	Wing 101
П.8	PyCharm
П.9	Unity

6.4. Перечень информационных справочных систем и профессиональных баз данных

И.1	https://www.computer.org/education/bodies-of-knowledge/software-engineering - Свод знаний по программной инженерии (SWEBOK)
И.2	http://www.firststeps.ru/java/java1.html - Разработка приложений на Java
И.3	http://www.gpntb.ru - Государственная публичная научно-техническая библиотека

7. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

Ауд.	Назначение	Оснащение
121	Учебная аудитория для занятий лекционного типа, практических занятий	Комплект учебной мебели на 56 мест для обучающихся, 1 стационарный компьютер для преподавателя (выход в интернет), проектор, экран настенный, колонки, доска аудиторная меловая, веб камера Logitech, лицензионные программы MS Office, MS Teams, антивирус Dr.Web.
123	Учебная лаборатория (компьютерный класс) Кабинет курсового и дипломного проектирования, самостоятельной работы обучающихся	Комплект учебной мебели на 12 мест для обучающихся, 12 стационарных компьютеров для студентов, 1 стационарный компьютер для преподавателя (у всех выход в интернет), проектор, экран, коммутатор, веб камера, доска-флипчарт магн.-маркерная передвижная, доступ к ЭИОС Университета МИСИС через личный кабинет на платформе LMS Canvas и Moodle, лицензионные программы MS Office, MS Teams, антивирус Dr.Web.

8. МЕТОДИЧЕСКИЕ УКАЗАНИЯ ДЛЯ ОБУЧАЮЩИХСЯ

Освоение дисциплины предполагает как проведение традиционных аудиторных занятий, так и работу в электронной информационно-образовательной среде НИТУ «МИСиС» (ЭИОС), частью которой непосредственно предназначенной для осуществления образовательного процесса является Электронный образовательный ресурс LMS
Чтобы эффективно использовать возможности LMS а соответственно и успешно освоить дисциплину, нужно:
1) зарегистрироваться на курс. Для этого нужно перейти по ссылке … Логин и пароль совпадает с логином и паролем от личного кабинета НИТУ МИСиС;
2)в рубрике «В начало» ознакомиться с содержанием курса, вопросами для самостоятельной подготовки, условиями допуска к аттестации, формой промежуточной аттестации (зачет/экзамен), критериями оценивания и др.;
3)в рубрике «Модули», заходя в соответствующие разделы изучать учебные материалы, размещенные преподавателем. В т.ч. пользоваться литературой, рекомендованной преподавателем, переходя по ссылкам;
4)в рубрике «Библиотека» возможно подбирать для выполнения письменных работ (контрольные, домашние работы, курсовые работы/проекты) литературу, размещенную в ЭБС НИТУ «МИСиС;
5)в рубрике «Задания» нужно ознакомиться с содержанием задания к письменной работе, сроками сдачи, критериями оценки. В установленные сроки выполнить работу(ы), подгрузить здесь же для проверки. Удобно называть файл работы следующим образом (название предмета (сокращенно), группа, ФИО, дата актуализации (при повторном размещении)). Например, Экономика_Иванов_И.И._БМТ-19_20.04.2020. Если работа содержит рисунки, формулы, то с целью сохранения форматирования ее нужно подгружать в pdf формате.
Работа, подгружаемая для проверки, должна:
- содержать все структурные элементы: титульный лист, введение, основную часть, заключение, список источников, приложения (при необходимости);
- быть оформлена в соответствии с требованиями.
Преподаватель в течение установленного срока (не более десяти дней) проверяет работу и размещает в комментариях к заданию рецензию. В ней он указывает как положительные стороны работы, так замечания. При наличии в рецензии замечаний и рекомендаций, нужно внести поправки в работу, подгрузить ее заново для повторной проверки. При этом важно следить за сроками, в течение которых должно быть выполнено задание. При нарушении сроков, указанных преподавателем возможность подгрузить работу остается, но система выводит сообщение о нарушении сроков. По окончании семестра подгрузить работу не получится;
6)в рубрике «Тесты» пройти тестовые задания, освоив соответствующий материал, размещенный в рубрике «Модули»;
7)в рубрике «Оценки» отслеживать свою успеваемость;
8)в рубрике «Объявления» читать объявления, размещаемые преподавателем, давать обратную связь;
9)в рубрике «Обсуждения» создавать обсуждения и участвовать в них (обсуждаются общие моменты, вызывающие вопросы у большинства группы). Данная рубрика также может быть использована для взаимной проверки;
10)проявлять регулярную активность на курсе.
Преимущественно для синхронного взаимодействия между участниками образовательного процесса посредством сети «Интернет» используется Microsoft Teams (MS Teams). Чтобы полноценно использовать его возможности нужно установить приложение MS Teams на персональный компьютер и телефон. Старостам нужно создать группу в MS Teams. Участие в группе позволяет:
- слушать лекции;
- работать на практических занятиях;
- быть на связи с преподавателем, задавая ему вопросы или отвечая на его вопросы в общем чате группы в рабочее время с 9.00 до 17.00;
- осуществлять совместную работу над документами (вкладка «Файлы).

При проведении занятий в дистанционном синхронном формате нужно всегда работать с включенной камерой.
Исключение – если преподаватель попросит отключить камеры и микрофоны в связи с большими помехами. На аватарках должны быть исключительно деловые фото.
При проведении лекционно-практических занятий ведется запись. Это дает возможность просмотра занятия в случае невозможности присутствия на нем или при необходимости вновь обратится к материалу и заново его просмотреть.