МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ГЕОДЕЗИИ И КАРТОГРАФИИ
(МИИГАиК)

Факультет геоинформатики и информационной безопасности
Тюменский государственный университет

**ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ**

Сертификат: 5BFEE1D286E9C95A0C0D2C611E434BD5
Владелец: Степанченко Алексей Леонидович
Действителен: с 23.09.2026 по 17.12.2027

«УТВЕРЖДАЮ»
Проректор
А. Л. Степанченко
24 марта 2025 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ) «Б1.О.09 ПРОГРАММНАЯ ИНЖЕНЕРИЯ»

Уровень высшего образования: магистратура

Направление подготовки: 09.04.03 Прикладная информатика

Направленность (профиль) подготовки: Сквозные цифровые технологии в управлении недвижимостью

Квалификация (степень) выпускника: магистр

Форма обучения: очная

Год набора (приема на обучение): 2025

Срок получения образования: 1 года 10 месяца(-ев)

Объем: в зачетных единицах: 9 з.е.
в академических часах: 324 ак.ч.

Москва, 2025

Разработчики:
Профессор академического департамента, доктор физико-математических наук Шевляков А. Н.

Рабочая программа дисциплины (модуля) составлена в соответствии с требованиями ФГОС ВО по направлению подготовки 09.04.03 Прикладная информатика, утвержденного приказом Минобрнауки от 19.09.2017 № 916

Согласование и утверждение

№	Подразделение	Ответственное лицо	ФИО	Виза	Дата, протокол (при наличии)
1	факультет Геоинформатики и информационной безопасности	Председатель ученого совета факультета	Матерухин А. В.	Рассмотрено	14.10.2024, № 10-24

Содержание

1. Цель и задачи освоения дисциплины (модуля)

2. Планируемые результаты обучения по дисциплине (модулю), соотнесенные с планируемыми результатами освоения образовательной программы

3. Место дисциплины в структуре ОП

4. Объем дисциплины и виды учебной работы

5. Содержание дисциплины
   5.1. Разделы, темы дисциплины и виды занятий
   5.2. Содержание разделов, тем дисциплины и формы текущего контроля

6. Оценочные материалы текущего контроля

7. Оценочные материалы промежуточной аттестации

8. Порядок проведения промежуточной аттестации

9. Материально-техническое и учебно-методическое обеспечение дисциплины
   9.1. Перечень основной и дополнительной учебной литературы
   9.2. Профессиональные базы данных и ресурсы «Интернет», к которым обеспечивается доступ обучающихся
   9.3. Программное обеспечение и информационно-справочные системы, используемые при осуществлении образовательного процесса по дисциплине
   9.4. Оборудование и технические средства обучения

10. Методические указания по освоению дисциплины (модуля)

1. Цель и задачи освоения дисциплины (модуля)

Цель освоения дисциплины - приобретение обучающимися знаний в области современных методологий, принципов и стандартов программной инженерии, а также инструментальных средств разработки программного обеспечения, необходимых для эффективного управления процессами создания, сопровождения и эволюции программных продуктов.

Задачи изучения дисциплины:

- изучить основные процессы жизненного цикла программного обеспечения, стандарты и регламенты, регулирующие разработку программных продуктов;
- сформировать практические навыки по сбору и анализу требований, объектно-ориентированному анализу и проектированию, а также по документированию результатов проектирования;
- научиться применять современные инструментальные средства и метрики для оценки качества, трудоемкости и сроков разработки программного обеспечения.

2. Планируемые результаты обучения по дисциплине (модулю), соотнесенные с планируемыми результатами освоения образовательной программы

Компетенции, индикаторы и результаты обучения

ОПК-2 Способен разрабатывать оригинальные алгоритмы и программные средства, в том числе с использованием современных интеллектуальных технологий, для решения профессиональных задач

ОПК-2.1 Умеет разрабатывать оригинальные алгоритмы и программные средства с использованием современных интеллектуальных технологий

Знать:

ОПК-2.1/Зн1 Современные информационно-коммуникационные и интеллектуальные технологии, инструментальные среды, программно-технические платформы для решения профессиональных задач

ОПК-2.1/Зн2 Методы разработки оригинальных программных средств, в том числе с использованием современных информационно-коммуникационных и интеллектуальных технологий, для решения профессиональных задач

ОПК-2.1/Зн3 Архитектурные принципы построения систем искусственного интеллекта, методы декомпозиции основных подсистем (компонентов) и реализации их взаимодействия

Уметь:

ОПК-2.1/Ум1 Обосновывать выбор современных информационно-коммуникационных и интеллектуальных технологий, разрабатывать оригинальные программные средства для решения профессиональных задач

ОПК-2.1/Ум2 Разрабатывать программное и аппаратное обеспечение информационных и автоматизированных систем для решения профессиональных задач

Владеть:

ОПК-2.1/Нв1 Навыками использования инструментов и методов разработки программных компонентов систем, основанных на знаниях

ОПК-2.1/Нв2 Навыками разработки систем искусственного интеллекта и их подсистем на основе различных архитектурных принципов

ОПК-2.2 Применяет расширенный математический аппарат при построении и обосновании оригинальных алгоритмов

Знать:

ОПК-2.2/Зн1 Методы построения программных алгоритмов

Уметь:

ОПК-2.2/Ум1 Применять расширенный математический аппарат при построении и обосновании оригинальных алгоритмов

Владеть:

ОПК-2.2/Нв1 Навыками обоснования оригинальных программных решений и алгоритмов

ОПК-2.3 Разрабатывает программные средства на основе методов интеллектуального анализа данных для решения профессиональных задач

Знать:

ОПК-2.3/Зн1 Методы интеллектуального анализа данных для решения профессиональных задач

Уметь:

ОПК-2.3/Ум1 Разрабатывать программные средства на основе интеллектуального анализа данных

Владеть:

ОПК-2.3/Нв1 Навыками разработки программных средств с использованием системного подхода

3. Место дисциплины в структуре ОП

Дисциплина (модуль) Б1.О.09 «Программная инженерия» относится к обязательной части образовательной программы и изучается в семестре(ах): 2.

В процессе изучения дисциплины студент готовится к решению типов задач профессиональной деятельности, предусмотренных ФГОС ВО и образовательной программой.

4. Объем дисциплины (модуля) и виды учебной работы

Период обучения	Общая трудоемкость (часы)	Общая трудоемкость (ЗЕТ)	Лекционные занятия (часы)	Практические занятия (часы)	Самостоятельная работа (часы)	Промежуточная аттестация (часы)
2 семестр	324	9	16	32	267	Зачет с оценкой (9)
Всего	324	9	16	32	267	9

5. Содержание дисциплины (модуля)

5.1. Разделы, темы дисциплины и виды занятий

(часы промежуточной аттестации не указываются)

		ия	тия	абота	ьтаты нные с ения

Наименование раздела, темы	Всего	Лекционные занятия	Практические занятия	Самостоятельная работа	Планируемые результаты обучения, соотнесенные с результатами освоения программы
Раздел 1. Введение в программную инженерию. Жизненный цикл программного продукта	**39**	**2**	**4**	**33**	ОПК-2.1 ОПК-2.2 ОПК-2.3
Тема 1.1. Модели жизненного цикла и методологии разработки ПО	39	2	4	33	
Раздел 2. Разработка приложений и систем машинного обучения	**39**	**2**	**4**	**33**	ОПК-2.1 ОПК-2.2 ОПК-2.3
Тема 2.1. Особенности инженерии ML-проектов (MLOps)	39	2	4	33	
Раздел 3. Системы контроля версий и командная разработка	**39**	**2**	**4**	**33**	ОПК-2.1 ОПК-2.2 ОПК-2.3
Тема 3.1. Коллективная работа с Git и платформенные решения (GitHub/GitLab)	39	2	4	33	
Раздел 4. Основы Continuous Integration (CI)	**39**	**2**	**4**	**33**	ОПК-2.1 ОПК-2.2 ОПК-2.3
Тема 4.1. Автоматизация сборки и проверки кода с помощью CI-серверов	39	2	4	33	
Раздел 5. Архитектура программного обеспечения. Разработка API для систем машинного обучения	**39**	**2**	**4**	**33**	ОПК-2.1 ОПК-2.2 ОПК-2.3
Тема 5.1. Проектирование архитектуры и способы поставки ML-моделей через API	39	2	4	33	
Раздел 6. Переиспользование программного кода	**39**	**2**	**4**	**33**	ОПК-2.1 ОПК-2.2 ОПК-2.3
Тема 6.1. Пакетирование, менеджеры зависимостей и повторное использование кода	39	2	4	33	
Раздел 7. Стиль и качество кода. Рецензирование кода (Code Review)	**39**	**2**	**4**	**33**	ОПК-2.1 ОПК-2.2 ОПК-2.3
Тема 7.1. Стандарты оформления кода и практика код-ревью	39	2	4	33	
Раздел 8. Тестирование систем машинного обучения	**42**	**2**	**4**	**36**	ОПК-2.1 ОПК-2.2 ОПК-2.3
Тема 8.1. Многоуровневое тестирование в ML-проектах	42	2	4	36	

Итого	**315**	**16**	**32**	**267**	

5.2. Содержание разделов, тем дисциплин и формы текущего контроля

Раздел 1. Введение в программную инженерию. Жизненный цикл программного продукта

Тема 1.1. Модели жизненного цикла и методологии разработки ПО

Программная инженерия представляет собой индустриальный подход к разработке программного обеспечения, который принципиально отличается от просто программирования тем, что охватывает все аспекты производства ПО — от начальной концепции до финального сопровождения. В рамках данной темы рассматривается понятие жизненного цикла программного продукта, включающее такие ключевые этапы, как анализ требований, проектирование архитектуры, реализация кода, тестирование, внедрение и последующее сопровождение. Особое внимание уделяется обзору основных моделей жизненного цикла: каскадной модели, предполагающей последовательное выполнение этапов, итеративному подходу с постепенным наращиванием функциональности, спиральной модели, ориентированной на управление рисками, а также инкрементальной модели. Проводится сравнение классических моделей, таких как Waterfall и V-Model. Завершается тема введением в гибкие методологии разработки (Agile, Scrum, Kanban), которые стали современным стандартом индустрии благодаря своей ориентации на быструю поставку ценности и адаптацию к изменяющимся требованиям.

Текущий контроль

Вид (форма) контроля
Тестирование

Раздел 2. Разработка приложений и систем машинного обучения

Тема 2.1. Особенности инженерии ML-проектов (MLOps)

Разработка систем машинного обучения имеет свою уникальную специфику, которая отличает ее от создания традиционного программного обеспечения. В отличие от классического ПО, где код является единственным изменяемым артефактом, в ML-проектах ключевую роль играют данные, которые постоянно эволюционируют, а также эксперименты с моделями, что требует принципиально иного подхода к организации рабочего процесса. Данная тема освещает полный жизненный цикл ML-проекта: от сбора данных и их очистки (data wrangling) через исследовательский анализ данных (EDA) к обучению моделей, их валидации и последующему развертыванию в productive-среду. Отдельно рассматриваются проблемы управления версиями не только кода, но и данных, а также экспериментов, для чего существуют специализированные инструменты вроде DVC и MLflow. В итоге студенты знакомятся с концепцией MLOps, которая представляет собой практику унификации процессов разработки и эксплуатации систем машинного обучения, обеспечивая их надежность и воспроизводимость.

Текущий контроль

Вид (форма) контроля
Тестирование

Раздел 3. Системы контроля версий и командная разработка

Тема 3.1. Коллективная работа с Git и платформенные решения (GitHub/GitLab)

Современная разработка программного обеспечения немыслима без использования систем контроля версий, и стандартом де-факто в этой области является Git. В рамках темы изучаются основы распределенных систем контроля версий: понятие репозитория, создание коммитов как фиксация изменений, работа с ветками (branching), позволяющая вести параллельную разработку функциональности. Особое внимание уделяется различным моделям ветвления, таким как Git Flow с его строгой иерархией веток и более легковесный GitHub Flow, а также практическим навыкам разрешения конфликтов слияния (merge conflicts), неизбежно возникающих при командной работе. Важной частью темы является организация коллективной разработки через механизм форков (forks) и pull request (или merge request), который не только служит для внесения изменений, но и является точкой коммуникации в команде. Рассматриваются также дополнительные возможности платформ GitHub и GitLab, такие как Issues для учета задач, проектные доски и Wiki для документирования, которые превращают эти сервисы в полноценную среду для командного взаимодействия.

Текущий контроль

Вид (форма) контроля
Тестирование

Раздел 4. Основы Continuous Integration (CI)

Тема 4.1. Автоматизация сборки и проверки кода с помощью CI-серверов

Непрерывная интеграция (Continuous Integration, CI) представляет собой практику разработки, при которой все изменения кода регулярно и часто интегрируются в общий репозиторий, после чего автоматически запускаются процессы сборки и тестирования. Главная цель CI заключается в раннем обнаружении ошибок и конфликтов интеграции, что значительно снижает риски при разработке и позволяет поддерживать код в работоспособном состоянии в любой момент времени. В ходе изучения темы студенты знакомятся с основными компонентами CI-процесса: центральным репозиторием, автоматической сборкой проекта и автоматическим прогоном тестов. Практическая часть включает настройку простого CI-пайплайна (pipeline) с использованием современных инструментов, таких как GitHub Actions или GitLab CI/CD. Важным выводом темы становится понимание того, что линтеры, статические анализаторы кода и автоматические тесты должны быть обязательными этапами любого CI-процесса, формируя защитную сетку, которая ловит ошибки до того, как они попадут к конечным пользователям.

Текущий контроль

Вид (форма) контроля
Тестирование

Раздел 5. Архитектура программного обеспечения. Разработка API для систем машинного обучения

Тема 5.1. Проектирование архитектуры и способы поставки ML-моделей через API

Архитектура программного обеспечения определяет фундаментальные структуры системы и принципы ее организации, включая выделение компонентов, их взаимосвязи и паттерны взаимодействия. В данной теме рассматриваются основные архитектурные стили, такие как многослойная архитектура, клиент-сервер и микросервисы, а также ключевые принципы проектирования качественной архитектуры, включая SOLID и подходы к созданию чистой архитектуры, которые применимы и к Data Science проектам. Особое внимание уделяется способам интеграции моделей машинного обучения в существующие информационные системы, и здесь REST API становится стандартным решением для предоставления сервисов инференса моделей. Студенты изучают, как проектируются эндпоинты API, какие методы HTTP-запросов используются (например, POST для отправки данных модели и получения предсказаний), и получают практические навыки упаковки обученной модели в веб-сервис с использованием современных фреймворков, таких как FastAPI или Flask, что позволяет модели выдавать предсказания в реальном времени по запросу.

Текущий контроль

Вид (форма) контроля
Тестирование

Раздел 6. Переиспользование программного кода

Тема 6.1. Пакетирование, менеджеры зависимостей и повторное использование кода

Принцип DRY (Don't Repeat Yourself) лежит в основе эффективной разработки, и его практическая реализация достигается через модульность и переиспользование кода. В рамках данной темы рассматривается, как создавать собственные пакеты и библиотеки, которые можно использовать в различных проектах, включая структуру такого проекта, организацию файлов и назначение файлов setup.py или pyproject.toml для описания метаданных пакета. Важнейшим аспектом переиспользования является управление зависимостями — внешними библиотеками, от которых зависит проект, и здесь студенты знакомятся с работой менеджеров пакетов, таких как pip, poetry и conda-env, а также с проблемой конфликта версий, известной как dependency hell. Тема также охватывает вопросы версионирования собственных разработок, где стандартом является семантическое версионирование (SemVer), позволяющее четко коммуницировать характер изменений (мажорные, минорные, патчи) между разработчиками и пользователями библиотек.

Текущий контроль

Вид (форма) контроля
Тестирование

Раздел 7. Стиль и качество кода. Рецензирование кода (Code Review)

Тема 7.1. Стандарты оформления кода и практика код-ревью

Качество кода напрямую влияет на стоимость его сопровождения и скорость внесения изменений, а плохо написанный код неизбежно ведет к накоплению технического долга (technical debt). Данная тема посвящена формированию культуры написания чистого, читаемого кода, что начинается со следования стандартам оформления, таким как PEP 8 для Python, и автоматизации их проверки с помощью линтеров (flake8, pylint, black) и форматтеров. Центральное место в обеспечении качества в командной разработке занимает процесс рецензирования кода (Code Review), который рассматривается не как инструмент контроля, а как механизм совместного улучшения кода и распространения знаний в команде. Студенты изучают, как строить эффективные чек-листы для ревьювера, охватывающие проверку логики, стиля оформления, производительности и безопасности, и, что не менее важно, осваивают культуру общения в код-ревью: как формулировать конструктивные замечания и как правильно реагировать на критику своего кода.

Текущий контроль

Вид (форма) контроля
Тестирование

Раздел 8. Тестирование систем машинного обучения

Тема 8.1. Многоуровневое тестирование в ML-проектах

Тестирование систем машинного обучения представляет собой более сложную задачу по сравнению с тестированием традиционного ПО, поскольку требует проверки не только корректности кода, но и качества данных, а также адекватности поведения моделей. В рамках данной темы изучается многоуровневый подход к тестированию ML-проектов, который включает юнит-тесты для проверки отдельных функций обработки и трансформации данных, а также интеграционные тесты для проверки корректной работы всего пайплайна обучения или инференса в целом. Отдельное внимание уделяется специфическим для машинного обучения видам тестирования, таким как проверка качества обученной модели на отложенной выборке (hold-out set) и валидация метрик (precision, recall, accuracy, F1-мера) на соответствие бизнес-требованиям. Завершается тема рассмотрением вопросов тестирования моделей на устойчивость к возмущениям во входных данных и, что особенно важно для эксплуатации, методов обнаружения дрейфа данных (data drift detection), когда статистические характеристики входных данных меняются со временем, что может приводить к деградации качества предсказаний модели.

Текущий контроль

Вид (форма) контроля
Тестирование

6. Порядок проведения промежуточной аттестации

Порядок проведения промежуточной аттестации регламентируется локальными нормативными актами Университета:
- Положением о формах, периодичности, порядке текущего контроля успеваемости и промежуточной аттестации обучающихся по основным образовательным программам высшего образования – программам бакалавриата, программам специалитета, программам магистратуры в федеральном государственном бюджетном образовательном учреждении высшего образования «Московский государственный университет геодезии и картографии», п.6 «Организация и проведение промежуточной аттестации».
- Положением о порядке осуществления образовательного процесса, проведения текущего контроля успеваемости и промежуточной аттестации обучающихся с применением электронного обучения и дистанционных образовательных технологий в федеральном государственном бюджетном образовательном учреждении высшего образования «Московский государственный университет геодезии и картографии», п.8 «Проведение промежуточной аттестации с применением ЭО и ДОТ».

Промежуточная аттестация: Зачет с оценкой, 2 семестр.
Зачет оценивается по пятибалльной шкале:
- зачтено с оценкой «отлично» выставляется, если обучающийся имеет глубокие, исчерпывающие знания в объеме освоенной программы; знание основной (обязательной) литературы; дает правильные и уверенные ответы, свидетельствующие о наличии твердых знаний и навыков в использовании технических средств; полное, четкое, грамотное и логически стройное изложение материала; демонстрирует свободное применение теоретических знаний при анализе практических вопросов;
- зачтено с оценкой «хорошо» выставляется обучающемуся в случае соблюдения тех же требований, но наличии в ответе обучающегося по некоторым перечисленным показателям недостатков принципиального характера, что вызывает замечания или поправки преподавателя;
- зачтено с оценкой «удовлетворительно» выставляется обучающемуся в случае соблюдения тех же требований, но в случае наличия в ответе ошибок, необходимости существенной помощи в виде поправок и наводящих вопросов преподавателя;
- не зачтено («не удовлетворительно») – обучающийся не смог ответить на 2/3 вопросов; не справился с практическим заданием или выполнено менее 50 % практических заданий.

7. Оценочные материалы текущего контроля

Раздел 1. Введение в программную инженерию. Жизненный цикл программного продукта
Контролируемые ИДК: ОПК-2.1 ОПК-2.2 ОПК-2.3

Тема 1.1. Модели жизненного цикла и методологии разработки ПО
Форма контроля/оценочное средство: Тестирование
Вопросы/Задания:
1. Как называется модель разработки с последовательными этапами?
2. Как называется семейство гибких методологий разработки?
3. Как называется модель жизненного цикла, ориентированная на управление рисками?

Раздел 2. Разработка приложений и систем машинного обучения
Контролируемые ИДК: ОПК-2.1 ОПК-2.2 ОПК-2.3

Тема 2.1. Особенности инженерии ML-проектов (MLOps)
Форма контроля/оценочное средство: Тестирование
Вопросы/Задания:
1. Как называется практика объединения разработки и эксплуатации ML-систем?

2. Какой инструмент используется для версионирования данных в ML-проектах?
3. Какой инструмент используется для отслеживания экспериментов в ML?
4. Как называется процесс получения предсказаний от обученной модели?

Раздел 3. Системы контроля версий и командная разработка
Контролируемые ИДК: ОПК-2.1 ОПК-2.2 ОПК-2.3

Тема 3.1. Коллективная работа с Git и платформенные решения (GitHub/GitLab)
Форма контроля/оценочное средство: Тестирование
Вопросы/Задания:
1. Какая система контроля версий является наиболее популярной?
2. Как сокращенно называют запрос на слияние изменений в Git?
3. Как называется стратегия ветвления Git с долгоживущими ветками main и develop?

Раздел 4. Основы Continuous Integration (CI)
Контролируемые ИДК: ОПК-2.1 ОПК-2.2 ОПК-2.3

Тема 4.1. Автоматизация сборки и проверки кода с помощью CI-серверов
Форма контроля/оценочное средство: Тестирование
Вопросы/Задания:
1. Как называется процесс автоматической сборки и тестирования при каждом изменении кода?
2. Как называется этап проверки стиля кода в CI-пайплайне?
3. Назовите один из самых популярных CI-серверов.

Раздел 5. Архитектура программного обеспечения. Разработка API для систем машинного обучения
Контролируемые ИДК: ОПК-2.1 ОПК-2.2 ОПК-2.3

Тема 5.1. Проектирование архитектуры и способы поставки ML-моделей через API
Форма контроля/оценочное средство: Тестирование
Вопросы/Задания:
1. Какой архитектурный стиль чаще всего используется для создания API?
2. Какой формат данных является стандартом для REST API?
3. Как называется архитектурный стиль, предполагающий разбиение приложения на мелкие независимые сервисы?

Раздел 6. Переиспользование программного кода
Контролируемые ИДК: ОПК-2.1 ОПК-2.2 ОПК-2.3

Тема 6.1. Пакетирование, менеджеры зависимостей и повторное использование кода
Форма контроля/оценочное средство: Тестирование
Вопросы/Задания:
1. Как называется принцип программирования "не повторяйся"?
2. Как называется стандартный менеджер пакетов для Python?
3. Как называется стандарт версионирования ПО в формате Major.Minor.Patch?
4. Как называется инструмент для создания изолированных окружений Python?

Раздел 7. Стиль и качество кода. Рецензирование кода (Code Review)
Контролируемые ИДК: ОПК-2.1 ОПК-2.2 ОПК-2.3

Тема 7.1. Стандарты оформления кода и практика код-ревью

Форма контроля/оценочное средство: Тестирование

Вопросы/Задания:

1. Как называется процесс рецензирования кода коллегами?

2. Как называется инструмент статического анализа кода для проверки стиля?

3. Как называется накопление проблем в коде, вызванное компромиссными решениями?

4. Как называется стандарт оформления кода на Python?

Раздел 8. Тестирование систем машинного обучения

Контролируемые ИДК: ОПК-2.1 ОПК-2.2 ОПК-2.3

Тема 8.1. Многоуровневое тестирование в ML-проектах

Форма контроля/оценочное средство: Тестирование

Вопросы/Задания:

1. Как называются тесты, проверяющие работу отдельного модуля или функции?

2. Как называется явление изменения статистических характеристик входных данных модели с течением времени?

3. Как называется метрика, показывающая долю истинно положительных результатов среди всех положительных предсказаний?

4. Как называется набор данных, используемый для финальной оценки качества модели?

5. Как называется метрика, показывающая долю найденных истинно положительных объектов?

6. Как называется процесс развертывания модели в продуктивную среду?

8. Оценочные материалы промежуточной аттестации

2 семестр, Зачет с оценкой

Контролируемые ИДК: ОПК-2.1 ОПК-2.2 ОПК-2.3

Вопросы/Задания:

1. Дайте определение понятию «программная инженерия». Чем она отличается от программирования?

2. Перечислите и кратко охарактеризуйте основные этапы жизненного цикла программного обеспечения.

3. В чем заключаются ключевые различия между каскадной (Waterfall) и итеративной моделями разработки?

4. Опишите основные принципы и ценности гибких методологий разработки (Agile).

5. В чем заключается специфика жизненного цикла систем машинного обучения по сравнению с традиционным ПО?

6. Какие дополнительные артефакты (помимо кода) появляются в ML-проектах и почему они требуют управления версиями?

7. Что такое MLOps? Какие проблемы призвана решать эта практика?

8. Опишите типовой пайплайн разработки системы машинного обучения (от постановки задачи до эксплуатации).

9. Объясните принцип работы распределенных систем контроля версий на примере Git. В чем их преимущество перед централизованными?

10. Для чего в Git используются ветки (branches)? Опишите одну из популярных стратегий ветвления (Git Flow или GitHub Flow).

11. Что такое pull request (merge request)? Опишите его жизненный цикл и роль в командной разработке.

12. Как разрешить конфликт слияния (merge conflict) в Git? Опишите последовательность действий разработчика.

13. Дайте определение непрерывной интеграции (Continuous Integration). Какие практики она в себя включает?

14. Какие проблемы в разработке решает внедрение CI и какие риски снижает?

15. Опишите типовую структуру CI-пайплайна. Какие этапы являются обязательными?

16. Какую роль выполняют автоматические тесты и статические анализаторы кода в процессе непрерывной интеграции?

17. Что такое архитектура программного обеспечения? Назовите основные архитектурные стили, которые вы знаете.

18. В чем заключается идея микросервисной архитектуры? Каковы ее преимущества и недостатки?

19. Для чего при разработке систем машинного обучения используется REST API?

20. Опишите структуру типового запроса и ответа при обращении к API модели машинного обучения.

21. Раскройте смысл принципа DRY (Don't Repeat Yourself). Как он связан с модульностью и переиспользованием кода?

22. Для чего нужны менеджеры зависимостей (например, pip, poetry) и какую проблему они решают?

23. Что такое семантическое версионирование (SemVer)? Как по номеру версии определить характер изменений?

24. Какие существуют способы создания изолированных окружений для Python-проектов и зачем это нужно?

25. Дайте определение понятию «технический долг» (technical debt). Какие практики помогают его минимизировать?

26. С какой целью используются стандарты оформления кода (например, PEP 8) и инструменты их проверки (линтеры)?

27. Опишите процесс Code Review. На что обращает внимание ревьювер при проверке кода?

28. Каковы правила конструктивного общения при проведении Code Review (как писать комментарии и как на них реагировать)?

29. Чем тестирование систем машинного обучения отличается от тестирования традиционного ПО? Какие новые уровни проверки появляются?

30. Что такое дрейф данных (data drift) и дрейф концепции (concept drift)? Почему их мониторинг критически важен для ML-систем в продуктивной среде?

9. Материально-техническое и учебно-методическое обеспечение дисциплины

9.1. Перечень основной и дополнительной учебной литературы

Основная литература

1. Кознов Д. В. Введение в программную инженерию: учебное пособие / Д. В. Кознов. - Введение в программную инженерию - Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2024. - 305 c. - 978-5-4497-2385-7. - Текст: электронный // IPR SMART: [сайт]. - URL: https://www.iprbookshop.ru/133932.html - Режим доступа: по подписке

2. Волк В. К. Практическое введение в программную инженерию / В. К. Волк. - 2-е изд., стер. - Санкт-Петербург: Лань, 2022. - 100 с. - 978-5-507-44920-0. - Текст: электронный // Издательство Лань: [сайт]. - URL: https://e.lanbook.com/book/249848 - Режим доступа: по подписке

3. Соловьев Н. А. Введение в программную инженерию: учебное пособие / Н. А. Соловьев, Л. А. Юркевская. - Введение в программную инженерию - Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017. - 112 c. - 978-5-7410-1685-5. - Текст: электронный // IPR SMART: [сайт]. - URL: https://www.iprbookshop.ru/71267.html - Режим доступа: по подписке

Дополнительная литература

1. Минакова О. В. Программная инженерия. Основные принципы, методы и инструменты: учебник для вузов / О. В. Минакова. - Санкт-Петербург: Лань, 2024. - 212 с. - 978-5-507-49278-7. - Текст: электронный // Издательство Лань: [сайт]. - URL: https://e.lanbook.com/book/414989 - Режим доступа: по подписке

2. Шуваев А. В. Программная инженерия: учебное пособие для магистрантов направления подготовки 09.04.02 – информационные системы и технологии / А. В. Шуваев. - Программная инженерия - Ставрополь: Ветеран, 2020. - 84 c. - 2227-8397. - Текст: электронный // IPR SMART: [сайт]. - URL: https://www.iprbookshop.ru/121736.html - Режим доступа: по подписке

3. Романов Е. Л. Программная инженерия: учебное пособие / Е. Л. Романов. - Программная инженерия - Новосибирск: Новосибирский государственный технический университет, 2017. - 395 c. - 978-5-7782-3455-0. - Текст: электронный // IPR SMART: [сайт]. - URL: https://www.iprbookshop.ru/91681.html - Режим доступа: по подписке

9.2. Профессиональные базы данных и ресурсы «Интернет», к которым обеспечивается доступ обучающихся

Профессиональные базы данных
Не используются.

Ресурсы «Интернет»

1. https://www.elibrary.ru/ - НЭБ eLIBRARY
2. https://znanium.ru/ - Электронно-библиотечная система "Znanium"
3. https://scholar.google.ru/ - Гугл Академия
4. https://link.springer.com/ - eBook Collections Springer Nature
5. https://rusneb.ru/ - Национальная электронная библиотека (НЭБ)
6. https://e.lanbook.com/ - Электронная библиотечная система "Лань"
7. https://www.iprbookshop.ru/ - IPR BOOKS

9.3. Программное обеспечение и информационно-справочные системы, используемые при осуществлении образовательного процесса по дисциплине

Перечень программного обеспечения
(обновление производится по мере появления новых версий программы)
Не используется.

Перечень информационно-справочных систем
(обновление выполняется еженедельно)
Не используется.

9.4. Оборудование и технические средства обучения

Компьютерный класс
Маркерная доска - 0 шт.
Моноблок - 64 шт.
Проектор - 4 шт.

10. Методические указания по освоению дисциплины (модуля)

С целью успешного освоения данного предмета студенту необходимо выполнять домашние работы по указанным преподавателям темам и представленным на занятиях материалам. Для более успешного выполнения работы студенту предлагается использование учебно-методических материалов, соответствующих указанной теме.

Самостоятельная работа студентов описывается и регулируется методическими рекомендациями по дисциплине.

Описание возможностей изучения дисциплины лицами с ОВЗ и инвалидами

При необходимости обучения студентов-инвалидов и лиц с ограниченными возможностями здоровья аудиторные занятия могут быть заменены или дополнены изучением полнотекстовых лекций, презентаций, видео- и аудиоматериалов в электронной информационно-образовательной среде (ЭИОС) университета. Индивидуальные задания подбираются в адаптированных к ограничениям здоровья формах (письменно или устно, в форме презентаций). Выбор методов обучения зависит от их доступности для инвалидов и лиц с ограниченными возможностями здоровья.

Рабочая программа дисциплины для инвалидов и лиц с ограниченными возможностями здоровья разрабатывается индивидуально с учетом особенностей психофизического развития, индивидуальных возможностей и состояния здоровья таких обучающихся.