Министерство науки и высшего образования российской федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Кафедра прикладной математики и программирования

Анализ требований и проектирование систем искусственного интеллекта

Методические указания по выполнению лабораторных работ студентов бакалавриата «Прикладная математика и искусственный интеллект» по направлению 01.03.02 «Прикладная математика и информатика»

2023

Содержание

1. Цели и задачи дисциплины

Целью преподавания дисциплины является обучение студентов методам выявления, анализа и разработки требований и методам проектирования программных систем с использованием искусственного интеллекта.

Задачи дисциплины заключаются в том, чтобы студенты овладели навыками анализа предметной области, создания и описания объектно-ориентированных моделей предметной области, выполнения системного анализа и разработки на его основе архитектуры, алгоритмических и программных решений системного и прикладного программного обеспечения, систем с использованием искусственного интеллекта; навыками создания спецификаций, как для всей системы в целом, так для отдельных подсистем и модулей.

2. Методы и формы организации обучения

Для успешного освоения дисциплины применяются различные образовательные технологии, которые обеспечивают достижение планируемых результатов обучения согласно основной образовательной программе, с учетом требований к объему занятий в интерактивной форме.

Для контроля освоения компетенций используются следующие формы контроля: защита лабораторных работ, опрос по изучаемым разделам дисциплины, тесты.

3. Место дисциплины в структуре ООП

Дисциплина «Современные технологии разработки программных систем искусственного интеллекта» относится к вариативной части профессионального цикла ООП.

Успешное овладение дисциплиной предполагает предварительные знания таких дисциплин как «Основы защиты данных в интеллектуальных системах», «Компьютерные сети», «Объектно-ориентированное программирование», «Современные технологии разработки программных систем искусственного интеллекта,» в объеме, предусмотренном направлением 01.03.02 – «Прикладная математика и информатика».

4. Лабораторные работы

Лабораторные работы дисциплины "Анализ требований и проектирование систем искусственного интеллекта" позволяет получить практические навыки выявления, анализа и разработки требований и методам проектирования программных систем с использованием искусственного интеллекта. .

Порядок выполнения лабораторных работ

1) изучить теоретический материал по теме лабораторной работы;
2) выполнить лабораторную работу для заданного варианта задания;
3) показать результаты преподавателю;
4) ответить на контрольные вопросы по лабораторной работе.

4.1 Задание 1

Цель лабораторной работы – знакомство с системами управления требований, характеристиками требований и их классификацией.

Теория

Ошибки в определении требований являются наиболее распространенными и приводят к затратам, составляющим 25-40% бюджета проекта разработки в целом.

Управление требованиями – это систематический подход к выявлению, организации и документированию требований к системе, а также процесс, в ходе которого вырабатывается и обеспечивается соглашение между заказчиком и выполняющей проект группой по поводу меняющихся требований к системе.

Проблема - это разница между желаемым и воспринимаемым.

Потребность - это отражение некой личной, рабочей или бизнес-проблемы, решение которой оправдывает замысел, покупку или использование новой системы.

Анализ проблемы - это процесс осознания реальных проблем и потребностей пользователя и предложение решений для удоблетворения этих потребностей.

Функция - обслуживание предоставляемое системой для выполнения одной или нескольких потребностей заказчика.

Систему произвольной сложности можно определить с помощью списка из 25-99 функций.

Требование к ПО:

- функциональное требование - свойство ПО, необходимое пользователю для решения проблемы при достижении поставленной цели;
- нефункциональное требование - свойство ПО, которым должна обладать система или её компонент, чтобы удовлетворить требования контракта, стандарта, спецификации или иной формальной документации.

Разработка требований должна обязательно принимать во внимание природу системы. Нельзя рассматривать системные свойства изолированно, - необходим комплексный подход: учет условий, которые привносит внешнее окружение, накладываемые кем-то или чем-то ограничения, а также интерфейсы с окружающими системами

Прежде чем приступать к разработке любой системы необходимо сначала определить - для чего эта система нужна. Если же цель создания системы не определена, то совершенно непонятно какой система должна быть и невозможно даже предположить устроит ли разработанная система ее пользователей.

Если вы не знаете куда идете, то вы вряд ли туда дойдете.

Изначально потребности могут быть выражены весьма смутно. Например, «мне нужна система, которая бы увеличила эффективность работы моего отдела».

Действия, в результате которых из нечеткой первоначальной характеристики потребностей получаются требования, которые могут служить в последствии основой для приобретения или создания системы, и формируют процесс разработки пользовательских требований (stakeholder requirements).

Системы управления требованиями (СУТ) — это программные приложения, облегчающие процесс сбора, документирования, анализа, проверки, определения приоритетов и утверждения требований к продукту. Управление требованиями — это непрерывный процесс, поскольку они могут меняться на протяжении всего жизненного цикла продукта. Он помогает согласовывать цели членов команды, способствуя выпуску продукта, который удовлетворит заинтересованные стороны.

СУТ имеют важные ключевые особенности:

- возможность указания связей между требованиями;
- возможность построения выборок в различных представлениях;
- функции отслеживания изменений.

Декомпозиция требований

СУТ должна обеспечивать лёгкое разбиение сложных требований на составные части до атомарных требований, с возможностью их «обратной сборки» в сводные требования. При этом все функции управления требованиями должны быть применимы к этим атомарным требованиям.

Лёгкость разбиения в данном случае является ключевой особенностью. В идеале у аналитика, разрабатывающего требования, должна быть возможность сделать атомарным требованием любой фрагмент текста «в один клик», не отвлекаясь при этом на оформление дополнительных атрибутов.

Типизация и шаблонизация требований

СУТ должна обеспечивать возможность создания требований различных типов из настраиваемых шаблонов. Для этого в ней должны быть реализованы:

- классификация типов требований;
- управление шаблонами требований (настройка шаблонов для каждого типа требований).

Типы требований определяются особенностями внутреннего устройства и подхода к разработке продукта. Например, если сайт содержит совокупность страниц, виджетов и плагинов, то для него естественно будет определить типы «Требования к странице», «Требования к виджету» и «Требования к плагину», и для каждого из этих типов разработать свой шаблон.

Классификация требований

СУТ должна обеспечивать классификацию требований по различным настраиваемым классификационным признакам, а также поиск и отбор требований по этим признакам.

Отличие классификации от типизации в том, что она обеспечивает удобную навигацию по множеству требований, но не определяет их форму и прочие характеристики. При разработке любого сравнительно сложного продукта выделяются функциональные области. Требования, относящиеся к одной

функиональной области, могут влиять друг на друга, и поэтому при разработке и управлении требованиями должна быть возможность оценить это влияние.

Например, при разработке сайта интернет-магазина могут быть выделены функциональные области: каталог товаров, оплата, управление заказами. Если мы добавляем новое требование в области управления заказами, то с помощью классификационных признаков мы можем отобрать другие требования из этой области, чтобы выявить возможные противоречия или взаимное влияние существующих требований с новым.

Трассировка требований друг на друга

СУТ должна поддерживать связи между требованиями. Должен поддерживаться настраиваемый набор видов связей.

Например: связывание бизнес-требований с функциональными требованиями, связывание влияющих друг на друга функциональных требований.

Наиболее очевидный вид связи — между родительскими и дочерними требованиями. В зависимости от особенностей продукта, могут потребоваться связи других видов.

Графическое моделирование требований

СУТ должна поддерживать возможность разработки требований в виде визуальных моделей (например, моделей UML и моделей бизнес-процессов).

Эта возможность может обеспечиваться интегрированной поддержкой средств моделирования или включением моделей в тексты требований. При этом к графическим моделям должны быть применимы все возможности разработки и управления требованиями.

Согласование требований с клиентами

СУТ должна обеспечивать возможность совместной разработки и согласования требований с заинтересованными лицами, не являющимися сотрудниками компании.

Эта возможность предполагает настраиваемый ограниченный доступ в СУТ из-за пределов компании, а также реализацию в ней настраиваемых процедур согласования.

Хранение первичных документов с требованиями

СУТ должна поддерживать хранение документов, являющихся источниками требований (например, полученные от заказчика), и обеспечивать лёгкий доступ к ним.

Эта возможность подразумевает учёт, настраиваемую классификацию, ведение версий документов и трассировку требований на них — так, чтобы при разработке и использовании требований пользователи могли легко найти и просмотреть связанные с ними документы.

Экспорт сводных документов требований

СУТ должна поддерживать экспорт наборов требований, отобранных по различным критериям, в виде файлов, основанных на шаблонах.

Эта возможность необходима в тех случаях, когда стандарты разработки требуют создания таких документов (например, если процесс должен

соответствовать ГОСТу), или если на определённых этапах создания продукта требования должны быть представлены в виде сводных документов — например, как приложения к договорам.

Версионирование требований

СУТ должна поддерживать ведение версий каждого отдельного требования.

Эта возможность предполагает, что каждая версия отдельного требования может рассматриваться как самостоятельное требование, к которому применимы некоторые из перечисленных возможностей СУТ. Должна быть также реализована возможность сравнения различных версий требования.

Варианты требований

СУТ должна поддерживать ведение нескольких вариантов отдельных требований.

В отличие от версионирования требований, эта возможность означает создание параллельных веток с вариантами требований — например, для разных локализаций. Эта возможность может быть необходима в тех случаях, когда продукт существует в виде нескольких параллельно поддерживаемых вариантов.

Управление статусами требований

В СУТ должен быть реализован настраиваемый набор статусов требований, отражающий их жизненный цикл в рамках бизнес-процессов компании. Должна быть возможность управления жизненным циклом как требования в целом, так и конкретной версии требования.

Например, нужно отслеживать статусы: готовности требований, согласования, реализации, документирования и т. п.

Если в компании используются автоматизированные инструменты для управления этими процессами, то эта возможность реализуется в рамках интеграции с этими инструментами.

Трассировка требований на рабочие продукты

СУТ должна обеспечивать связь требований с рабочими продуктами процессов разработки. Для этого СУТ должна поддерживать интеграцию с инструментами управления этими рабочими продуктами.

Под рабочими продуктами, в частности, понимаются:

- исходный код
- рабочие продукты тестирования — тесты, тестовые планы и т. п.
- документация

Управление задачами, связанными с требованиями

В СУТ должна быть реализована внутренняя поддержка или интеграция с внешней системой управления работами для создания задач, связанных с разработкой, согласованием и использованием требований. Должна поддерживаться привязка требований к отдельным задачам и отслеживание статусов требований по мере выполнения этих работ.

В частности, это нужно для таких видов задач: разработка и согласование требований, работы по реализации, тестированию, исправлению ошибок, выпуску релизов, версий и других видов поставок.

Управление базовыми линиями (baseline) требований

СУТ должна поддерживать управление базовыми линиями требований. Должна быть возможность отнесения к выбранной базовой линии конкретных версий отдельных требований, в том числе с использованием групповых операций. Кроме того, для управления базовыми линиями в СУТ должны быть реализованы отчёты, позволяющие оценивать состояние базовых линий по различным критериям.

Базовые линии (aka срезы) могут использоваться для планирования и фиксации требований к различным видам поставок (релизам, сборкам, версиям, патчам и т. п.).

Удалённый многоплатформенный доступ

Система должна обеспечивать удалённый доступ для совместной разработки, обсуждения, согласования и использования требований. Каждый участник должен иметь возможность получить доступ к системе со своего рабочего места, где бы оно ни находилось, и какую бы операционную систему не использовало.

На сегодняшний день стандартом универсального удалённого доступа является веб-интерфейс, позволяющий входить в систему из любой точки и с разных устройств.

Интеграция с используемыми в компании инструментами

Система должна поддерживать возможность интеграции с используемыми в компании инструментами: базой знаний, трекером задач, системой управления исходным кодом, инструментами тестирования и документирования и т. п.

Интеграция предполагает настраиваемый автоматический двусторонний обмен данными с этими инструментами.

Методические рекомендации

1. Выполнить регистрация участников группы в системе управления требованиями
2. Зафиксировать не менее 5 существующих требований к ПО (reverse engineering), например, Калькулятор
3. Предложить изменение требования, улучшение функциональности ПО, связанное с ИИ
4. Выполнить классификация требований

Примеры контрольных вопросов

1. Каково назначение системы управления требованиями (СУТ)?
2. Какие характеристики к требованиям можно указать в СУТ?

Критерии оценивания

1. Выполнена регистрация участников группы в СУТ — 1 балл
2. Зафиксированы не менее 5 существующих требований к ПО (reverse engineering) — 5 баллов (по 1 баллу за каждое требование)
3. Предложено изменение требования, улучшение функциональности ПО - 2 балла, иначе 0 баллов
4. Выполнена классификация требований — 2 балла, иначе 0 баллов

Итого 10 баллов

4.2 Задание 2

Цель лабораторной работы – знакомство с этапами анализа проблемы и выявления требований.

Теория

Задача аналитика - прежде всего выяснить, для чего нужна пользователям новая система, и затем определить пользовательские, функциональные и качественные требования, на основе которых команды смогут оценить и спланировать проект, а также спроектировать, построить и проверить продукт. Аналитик — это посредник в общении, проясняющий смутные представления пользователей и обращающий их в четкие спецификации, которыми руководствуется команда разработчиков продукта.

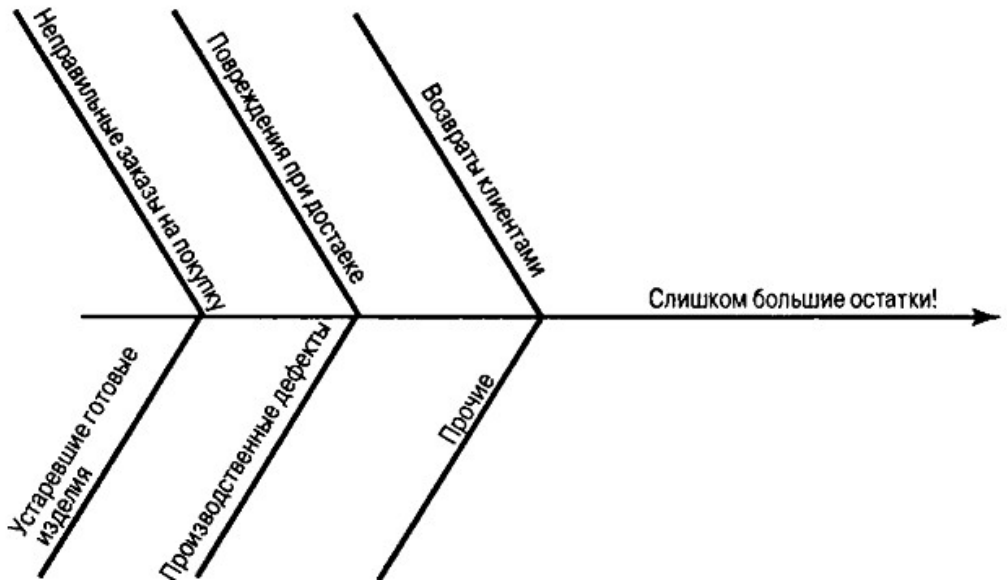
Помимо специальных навыков и личных качеств, бизнес-аналитик должен обладать обширными знаниями, большая часть которых приходит с опытом.

Этап 1. Достижение соглашения об определении проблемы

Простейший способ – просто записать проблему и выяснить, все ли согласны с такой постановкой. Примерная форма:

Элемент	Описание
Проблема	Описание проблемы
Воздействует на	Указание лиц, на которых оказывает влияние данная проблема
Результатом чего является	Описание воздействия данной проблемы на заинтересованных лиц и бизнес-деятельность
Выигрыш от	Указание предлагаемого решения
Может состоять в следующем	Список основных предоставляемых решением преимуществ

Этап 2. Выделение основных причин – проблем, стоящих за проблемой



Используется метод анализа корневых причин (рыбий скелет). Многие корневые причины не стоят того, чтобы их устранять, поскольку затраты на их устранение превысят причиняемый проблемам ущерб. Поэтому необходимо определить вклад каждой корневой причины в проблему.

Методические рекомендации

1. Записать проблему
2. Нарисовать диаграмму «рыбий скелет» для основных причин – проблем, стоящих за проблемой. Есть ли причины, которые можно решить с помощью методов искусственного интеллекта?
3. Найти не менее 2 конкурирующих продуктов, решающие предложенную проблему, которые будут использованы для сравнения

Примеры контрольных вопросов

1. Как выделять причины проблем?
2. Что такое конкурирующий продукт?

Критерии оценивания

1. Поставлена реальная проблема(ы) - 1 балл
2. Указано не менее 5 причин для проблем - 5 баллов (по 1 баллу за причину)
3. Есть причины причин - 1 балл
4. Есть причины, которые можно решить с помощью методов искусственного интеллекта - 1 балл
5. Есть список из не менее 2 конкурирующих продуктов, которые будут использованы для сравнения (ссылки) - 2 балла (по 1 баллу за продукт)

Итого 10 баллов

4.3 Задание 3

Цель лабораторной работы – знакомство с этапами анализа проблемы и выявления требований.

Теория

Этап 3. Выявление заинтересованных лиц и пользователей

Заинтересованные лица – это прямые и непрямые пользователи, которых затрагивает реализация новой системы или приложения.

Непрямые пользователи - клиенты, надзорные организации, лица, осуществляющие сопрождение и экплуатацию системы.

Этап 4. Определение границ системы-решения

Делим мир на две части – наша система и то, что взаимодействует с нашей системой.

Актант (актор) – это находящееся вне системы нечто (или некто), взаимодействующее с системой.

Этап 5. Выявление ограничений, налагаемых на решение

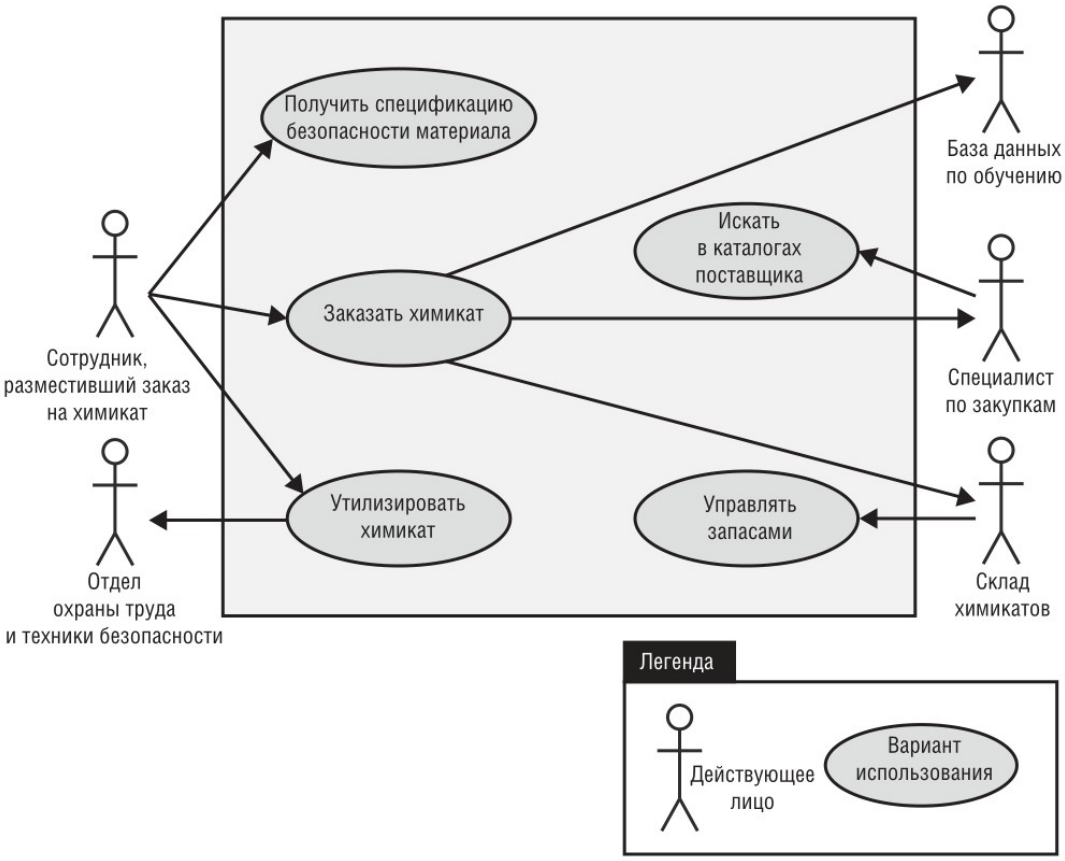
Ограничения уменьшают степень свободы, которой мы располагаем при предложении решения.

Некоторые из обнаруженных ограничений могут стать требованиями к системе. Возможные источники: экономические, политические, технические, системные, эксплуатационные, на разработку.

Методические рекомендации

1. Выявить заинтересованных лиц и пользователей
2. Определить границы системы-решения
3. Выявить ограничения, налагаемых на решение
4. Нарисовать диаграмму вариантов использования и дать описание одного из вариантов использования

Пример диаграммы вариантов использования и описания



Примечание идентификатор используется для трассировки, чтобы не писать полное имя, идентификатор содержит тип и номер: UC (user case), ВИ (вариант использования). Номер может простым или составным (номер диаграммы.номер ВИ). Примеры UC-4, ВИ-2.5

Идентификатор и название: UC-4 Заказать химикат

Автор: Лори

Дата создания: 22.08.13

Основное действующее лицо: Сотрудник, разместивший заказ на химикат

Дополнительное действующее лицо: Поставщик; Склад химикатов; База данных по обучению

Описание: Сотрудник, разместивший заказ на химикат, указывает в запросе необходимый химикат, вводя его название или идентификатор или импортируя его структуру из соответствующего графического средства. Система выполняет запрос, предлагая контейнер с химикатом со склада или позволяя создать запрос на заказ у поставщика.

Триггер: Сотрудник указывает, что хочет заказать химикат.

Предварительные условия:

PRE-1. Личность пользователя аутентифицирована.

PRE-2. Пользователь имеет право запрашивать химикаты.

PRE-3. База данных по запасам химикатов в данный момент доступна.

Выходные условия:

POST-1. Запрос сохраняется в Chemical Tracking System.

POST-2. Запрос отправлен на склад химикатов или поставщику.

Нормальное направление развития варианта использования:

4.0 Запросить химикат со склада

1. Сотрудник указывает требуемый химикат.
2. Система перечисляет контейнеры с необходимым химикатом, имеющиеся на складе.
3. Сотрудник может просмотреть историю любого контейнера.
4. Сотрудник выбирает определенный контейнер или просит отправить запрос поставщику (см. 4.1).
5. Сотрудник вводит остальную информацию, чтобы завершить запрос.
6. Система сохраняет запрос и отправляет его на склад химикатов.

Альтернативное направление развития варианта использования:

4.1 Запросить химикат у поставщика

1. Сотрудник ищет химикат по каталогам поставщика (см. 4.1.E1).
2. Система отображает список поставщиков, где также указаны размеры, класс и цена контейнеров.
3. Сотрудник выбирает поставщика, размер, класс и количество контейнеров.
4. Сотрудник вводит остальную информацию, необходимую для запроса.

5. Система сохраняет запрос и перенаправляет его поставщику.

Исключения:

4.1.E1. Химиката нет в продаже

1. Система отображает сообщение «У поставщиков нет такого химиката».
2. Система предлагает сотруднику запросить другой химикат или выйти из программы.

3а. Сотрудник просит запросить другой химикат.

4а. Система заново начинает нормальное направление варианта использования.

3б. Сотрудник решает выйти из системы.

4б. Система завершает вариант использования.

Приоритет: Высокий

Частота использования: Примерно пять раз в неделю каждым химиком, 200 раз в неделю каждым работником склада.

Бизнес-правила: BR-28, BR-31

Специальные требования: Система должна импортировать химические структуры в стандартной закодированной форме из любых средств, поддерживающих рисование химических структур.

Предположения: Импортированные химические структуры должны быть верными.

Примеры контрольных вопросов
1. Кто является заинтересованным лицом?
2. Что такое вариант использования?

Критерии оценивания
1. Нарисована диаграмма вариантов использования для всех прямых пользователей - 3 балла (оценка снижается на 1 балл за каждого отсутствующего пользователя)
2. Варианты использования соответствуют потребностям и целям пользователей - 2 балла (оценка снижается на 1 балл за каждую ошибку)
3. Есть описание одного из вариантов использования - 3 балла, иначе 0 баллов
4. В описании нет противоречий - 2 балла (оценка снижается на 1 балл за каждую ошибку)
Итого 10 баллов

4.4 Задание 4

Цель лабораторной работы – выявление задач, решаемых с помощью автоматизации и искусственного интеллекта.

Теория

В качестве первого шага следует разложить задачи, стоящие перед акторами, на отдельные компоненты. Сделав это, вы сразу поймете, что вопрос «На каких именно должностях автоматика заменит человека?» сформулирован неправильно. Ведь любая работа состоит из множества отдельных элементов, и для каждого из них использование автоматики будет иметь различный потенциал, да и эффект тоже окажется неодинаковым. Поэтому гораздо целесообразнее будет поинтересоваться: «Какие обязанности лучше всего передать машинам?»

Мы можем описать задачи с точки зрения возможностей для автоматизации, оценивая их в нескольких системах координат.

Стандартность – нестандартность. Данная задача шаблонна, состоит из предсказуемого набора операций с постоянными критериями успеха? Или она, напротив, носит нестандартный характер, а способы ее выполнения варьируются, равно как и параметры оценки? Быть может, она даже требует инновационных шагов и принятия решений в новых или всякий раз уникальных обстоятельствах?

Однообразная работа предсказуема и монотонна, она соответствует четким критериям. Меняющиеся задачи, напротив, невозможно спрогнозировать, такая деятельность требует разнообразных подходов, постоянной адаптации и самостоятельного принятия решений.

Независимость – взаимодействие. Эта задача возлагается на одного человека, который действует индивидуально, или же ее выполнение предполагает контакт с другими людьми, продуктивное сотрудничество и взаимопонимание?

Задачи, выполняемые сотрудником независимо, практически не требуют контактов и сотрудничества с другими людьми, в то время как существуют должности, где просто необходимо общаться и налаживать совместную работу, демонстрируя навыки общения и взаимопонимание.

Физический труд – умственная работа. Задача требует применения физического труда, силы и ловкости? Или в данном случае необходимы определенные знания, творческий подход, способность анализировать и давать оценку?

Задачи, связанные с ручным, физическим трудом, требуют ловкости и, как правило, силы, тогда как для решения интеллектуальных задач необходимо умение думать.

Далее надо определить, как связаны между собой качество работы и создаваемая ею дополнительная ценность для организации, – в дальнейшем мы будем употреблять словосочетание «прибыль от улучшения результатов работы» (ПУРР).

Как можно оценить ПУРР этой работы:

а) исключение ошибок - переход от крайне неудовлетворительного выполнения задачи, генерирующего отрицательную ценность, к минимально приемлемому уровню, дополнительная ценность при улучшении качества работы создается здесь за счет устранения потерь;

б) снижение вариативности - разница в качестве работы не производит дополнительной ценности, часто описывает задачи, для решения которых существует несколько способов: допустим, рабочие собирают детали в разной последовательности, но результат сборки при этом всегда одинаков;

в) плавное возрастание ценности - каждое улучшение в процессе работы приводит к стабильному увеличению выгоды, например, способность оператора убедить клиента потратить больше денег на товары и услуги, предлагая ему дополнительные опции или ускоренную доставку;

г) быстрое возрастание ценности - улучшение качества работы приводит к взрывному росту ценности.

Какую роль в данном случае сыграет автоматика: сможет ли она полностью заменить человека, расширит его возможности или создаст новую сферу деятельности?

Какие виды автоматизации доступны: роботизированная автоматизация процессов, познавательная автоматика, социальная робототехника?

Вида автоматизации:

• Роботизированная автоматизация процессов используется для решения масштабных шаблонных задач. Особенно эффективна для передачи данных из одной сети в другую, в случаях, не требующих обучения техники в процессе взаимодействия.

• Познавательная автоматика (искусственный интеллект, машинное обучение) используется для решения нестандартных, сложных, творческих, исследовательских задач. Особенно эффективна для распознавания закономерностей и смыслов в больших данных, а также там, где требуется обучение в процессе взаимодействия.

• Социальная (коллаборативная) робототехника используется для решения как типичных, так и нетипичных задач, требующих взаимодействия человека и машины. Роботы мобильны, они функционируют рядом с нами, их можно программировать и приспосабливать для решения новых проблем.

Каждый вид автоматизации соответствует своим типам рабочих задач и обеспечивает различные выгоды.

Каков оптимальный способ объединить возможности людей и механизмов в рамках связанных видов работ и производственных процессов?

ТАБЛИЦА П.1

Люди и механизмы: оптимальное сочетание трудовых ресурсов

	Характеристики работы					
	Стандартная/ нестандартная	**Выполняется индивидуально/ требует взаимодействия**	**Умственная/ физическая**	**Ценность (в соответствии с графиком ПУРР)**	**Роль автоматизации**	**Вид автоматизации**
1	Стандартная	Выполняется индивидуально	Физическая	Отрицательная	Заменяет человека; снижение числа ошибок	Социальная робототехника
2	Стандартная	Выполняется индивидуально	Физическая	Стабильная	Заменяет человека или расширяет его возможности; снижение вариативности	Социальная робототехника
3	Стандартная	Выполняется индивидуально	Физическая	Плавно растущая	Заменяет человека или расширяет его возможности; повышение производительности труда	Социальная робототехника
4	Стандартная	Выполняется индивидуально	Физическая	Быстро растущая	Расширяет возможности человека; повышение качества работы	Социальная робототехника
5	Стандартная	Требует взаимодействия	Физическая	Отрицательная	Заменяет человека; снижение числа ошибок	Социальная робототехника
6	Стандартная	Требует взаимодействия	Физическая	Стабильная	Заменяет человека или расширяет его возможности; снижение вариативности	Социальная робототехника
7	Стандартная	Требует взаимодействия	Физическая	Плавно растущая	Заменяет человека или расширяет его возможности; повышение производительности труда	Социальная робототехника

Продолжение табл. П. 1

	Характеристики работы					
	Стандартная/ нестандартная	**Выполняется индивидуально/ требует взаимодействия**	**Умственная/ физическая**	**Ценность (в соответствии с графиком ПУРР)**	**Роль автоматизации**	**Вид автоматизации**
23	Стандартная	Требует взаимодействия	Умственная	Плавно растущая	Заменяет человека или расширяет его возможности; повышение производительности труда	Познавательная автоматика
24	Стандартная	Требует взаимодействия	Умственная	Быстро растущая	Расширяет возможности человека; повышение качества работы	Познавательная автоматика
25	Нестандартная	Выполняется индивидуально	Умственная	Отрицательная	Заменяет человека; снижение числа ошибок	Познавательная автоматика
26	Нестандартная	Выполняется индивидуально	Умственная	Стабильная	Заменяет человека или расширяет его возможности; снижение вариативности	Познавательная автоматика
27	Нестандартная	Выполняется индивидуально	Умственная	Плавно растущая	Заменяет человека или расширяет его возможности; повышение производительности труда	Познавательная автоматика
28	Нестандартная	Выполняется индивидуально	Умственная	Быстро растущая	Расширяет возможности человека; повышение качества работы	Познавательная автоматика
29	Нестандартная	Требует взаимодействия	Умственная	Отрицательная	Заменяет человека; снижение числа ошибок	Познавательная автоматика
30	Нестандартная	Требует взаимодействия	Умственная	Стабильная	Заменяет человека или расширяет его возможности; снижение вариативности	Познавательная автоматика

Методические рекомендации

1. Выявить задачи для автоматизации
2. Определить пригодность методов искусственного интеллекта и машинного обучения для каждой задачи
3. Выполнить классификация всех задач для автоматизации
4. Выполнить оценка окупаемости (целесообразности) разработки

Примеры контрольных вопросов

1. Каковы критериям окупаемости?
2. Как классифицируют задачи для автоматизации?

Критерии оценивания

1. Выявлены задачи для автоматизации — 3 балла (1 балл за каждую задачу, но не более 3 баллов)
2. Определена пригодность методов искусственного интеллекта и машинного обучения- для каждой задачи 3 балла (оценка снижается на 1 балл за каждую ошибку в оценке)
3. Выполнена классификация всех задач для автоматизации - 3 балла (оценка снижается на 1 балл за каждую ошибку)
4. Выполнена оценка окупаемости (целесообразности) разработки - 1 балл

Итого 10 баллов

4.5 Задание 5

Цель практического задания – определение пригодности данных и задачи для машинного обучения и методов искусственного интеллекта.

Теория

Прежде чем приступить к какой-либо деятельности, связанной с машинным обучением, аналитик должен собрать и подготовить данные. Доступные аналитику данные не всегда «правильные» и не всегда представлены в форме, подходящей для алгоритма машинного обучения.

Данные в проекте машинного обучения могут использоваться для формирования примеров прямо или косвенно. На вход модели поступает последовательность слов. Мы должны преобразовать каждое слово естественного языка в машиночитаемый массив атрибутов, называемый вектором признаков. Для создания таких бинарных признаков мы можем прибегнуть к словарям, справочным таблицам, географическим справочникам или другим моделям машинного обучения, делающим предсказания относительно слов.

Набор последовательностей слов – это данные, используемые для формирования обучающих примеров непосредственно, тогда как данные, содержащиеся в словарях, справочных таблицах и географических справочниках, используются косвенно.

Первичные данные – это набор сущностей в их естественной форме; их не всегда можно задействовать для машинного обучения непосредственно. Например, документ Word или JPEG-файл являются первичными данными; алгоритм машинного обучения не может использовать их напрямую.

Необходимым (но не достаточным) условием использования данных в машинном обучении является их аккуратность. большинство алгоритмов машинного обучения принимают обучающие данные только в виде набора векторов числовых признаков. Атрибут «Регион» является категориальным, а не числовым. Алгоритм обучения на основе решающего дерева может работать со значениями категориальных атрибутов, но большинство алгоритмов обучения на это не способны и нужно преобразовать категориальный атрибут в числовой.

Получив данные в виде набора примеров, первое, что нужно сделать в проекте, – перемешать примеры и разделить данные на три отдельных набора: обучающий, контрольный и тестовый. Обучающий набор обычно является самым крупным; он используется в алгоритме обучения для порождения модели. Контрольный и тестовый наборы имеют примерно одинаковый, гораздо меньший, размер. Алгоритму обучения не разрешается использовать примеры из контрольного или тестового набора для обучения модели. Поэтому оба этих набора иногда называют зарезервированными.

Машинное обучение – мощный инструмент для решения практических задач. Однако, как и любой инструмент, его следует использовать в правильном контексте. Пытаться решать все задачи с помощью машинного обучения было бы ошибкой.

Возможность использования машинного обучения следует рассматривать в одной из следующих ниже ситуаций.

1. В ситуации, когда задача настолько сложна или велика, а входные данные, содержат слишком много параметров, которые коррелируют неизвестным образом, так что попытка написать все правила для ее решения попросту безнадежна, а частичное решение допустимо и представляет интерес.

2. Некоторые задачи со временем постоянно изменяются, поэтому исходный код необходимо регулярно обновлять.

3. Необходимо решать задачи восприятия, такие как распознавание речи, изображений и видео.

4. Если требуется предсказывать явление, которое недостаточно изучено с научной точки зрения, но наблюдаемо.

5. Когда задачу можно сформулировать как оптимизацию простой целевой функции: например, когда требуется найти бинарное решение типа да/нет или одно-единственное число. С другой стороны, машинное обучение не подойдет для построения модели, где число принимаемых решений слишком велико (например, видеоигра).

Три основных источника затрат в машинном обучении таковы:

- сбор, подготовка и очистка данных;
- обучение модели;
- создание и эксплуатация инфраструктуры для выполнения и мониторинга модели, а также трудовые ресурсы для ее сопровождения.

Применение оправдано, если 1) машинное обучение может заменить сложную часть инженерного проекта либо 2) получение недорогих (но, вероятно, несовершенных) предсказаний дает значительные преимущества.

Например, сложная часть существующей системы может быть основана на правилах со множеством вложенных правил и исключений. Процесс построения и сопровождения такой системы бывает чрезвычайно сложным, трудоемким и подвержен ошибкам. К тому же программисты не испытывают никакой радости от сопровождения этой части системы. Можно ли обучить, а не программировать правила? Можно ли использовать существующую систему для генерирования помеченных данных? Если да, то такой проект машинного обучения будет иметь высокую отдачу и низкую стоимость.

Недорогие и несовершенные предсказания бывают полезны, например, в системе, которая обрабатывает большое число запросов. Допустим, многие из таких запросов «легкие», так что на них можно быстро ответить с помощью существующей автоматической подсистемы. Остальные запросы считаются «трудными», так что решать их приходится вручную. Основанная на машинном обучении система, которая распознает «легкие» задачи и отправляет их автоматической подсистеме, сэкономит много времени людям, которые будут тратить усилия и время на решение трудных вопросов.

Успешная модель обладает следующими четырьмя свойствами:

- учитывает спецификации входных и выходных данных и требования к качеству;
- приносит пользу организации (измеряется снижением затрат, увеличением продаж или прибыли);

- помогает пользователю (измеряется продуктивностью, заинтересованностью и эмоциональным настроем);
- является строгой в научном отношении.

Методические рекомендации

1. Определить набор исходных данных для решения задачи
2. Выполнить оценку исходных данных по списку вопросов
   - Доступны ли данные?
   - Насколько велик объем данных?
   - Пригодны ли данные для использования?
   - Понятны ли данные?
   - Надежны ли данные?
   - Есть ли проблемы с данными? (Высокая стоимость, Плохое качество, Зашумленность, Смещение, Низкая предсказательная способность, Устаревшие примеры, Выбросы, Просачивание данных)
   - Нужна ли обработка?
   - Где хранятся данные?
3. Выполнить оценку эффекта от использования результатов

Примеры контрольных вопросов

1. Как оценить эффект от использования результатов?
2. Как оценить качество данных?

Критерии оценивания

1. Определен набор исходных данных для решения задачи - 3 балла (оценка снижается на 1 балл за каждую ошибку)
2. Выполнена оценка исходных данных по списку вопросов - 4 балла (оценка снижается на 1 балл за каждую ошибку)
3. Выполнена оценка эффекта от использования результатов - 3 балла (оценка снижается на 1 балл за каждую ошибку)

Итого 10 баллов

4.6 Задание 6

Цель лабораторной работы – изучение структуры и написание документа-концепции.

Теория

Документ-концепция описывает приложение в общих чертах, а также содержит описания целевых рынков, пользователей системы и функций приложения (как предлагаемых к реализации в версии 1, так и выявленные), ограничения системы.

Содержание документа-концепции

1. Введение

В данном разделе необходимо представить общую характеристику документа-концепции в целом

1.1. Цель документа-концепции

Цель данного документа состоит в сборе, анализе и определении высокоуровневых потребностей пользователей и функций продукта.

1.2. Общая характеристика продукта

В данном разделе определяется цель приложения, его версия и новые предоставляемые функции. Здесь следует

- указать продукт или приложение, которое создается или изменяется;
- дать общее описание того, что продукт будет делать и, если необходимо, чего не будет делать;
- описать применение продукта, в том числе достижимые с его помощью выгоды, цели и задачи.

1.3. Ссылки

Этот подраздел содержит список всех документов, упоминаемых где-либо в документе-концепции и список источников, к которым можно обратиться за справками.

2. Описание пользователя

2.1. Характеристика рынка/пользователя

Здесь необходимо кратко перечислить основные характеристики рынка, которые послужили мотивацией решений, касающихся продукта: описать и указать целевые сегменты, а также оценить объем и перспективы роста рынка, ориентируясь на число потенциальных пользователей или количество денег, которые в настоящее время тратят ваши заказчики при решении задач.

2.2. Описания пользователей

Здесь следует описать все типы пользователей. Для каждого типа пользователей указать

- Технический уровень и опыт
- Основные обязанности
- Тенденции, упрощающие или усложняющие работу пользователя
- В чем пользователь видит успех и как пользователь вознаграждается

2.3. Среда пользователя

- Сколько человек участвует в выполнении данной задачи? Изменится ли их число?
- Сколько времени длится цикл выполнения задачи? Изменится ли это?
- Существуют ли некие уникальные ограничения среды: на мобильную связь, по работе вне помещения, в полете и т.д.
- Какие системные платформы используются в настоящее время? Какие платформы предполагается использовать в будущем?
- Какие еще приложения используются? Должно ли ваше приложение объединяться с ними?

2.4. Основные потребности пользователя

Следует перечислить основные проблемы или потребности так, как они осознаются пользователем. Для каждой проблемы нужно прояснить следующие моменты

- В чем причины данной проблемы?
- Как она решается в настоящее время?

• Какие решения представляет себе пользователь?

2.5. Альтернативы и конкуренты

Нужно указать возможные альтернативы поведения пользователя. Среди них может быть покупка продукта конкурентов, создание собственного решения или просто сохранение существующей ситуации. Опишите основные преимущества и недостатки каждого варианта с точки зрения конечного пользователя.

2.5.1. Конкурент 1

3. Характеристика продукта

В данном разделе предлагается общее описание возможностей продукта, интерфейсов с другими приложениями и конфигураций систем.

3.1. Общее описание продукта

В данном подразделе следует описать, как продукт взаимодействует с другими связанными с ним продуктами и средой пользователя. Если продукт является независимым и самодостаточным, это необходимо указать. Если продукт является компонентом более крупной системы, в данном подразделе необходимо описать, как эти системы взаимодействуют, а также указать соответствующие интерфейсы между системами.

3.2. Определение позиции продукта

Предлагается общее определение, характеризующее на самом высоком уровне абстракции назначение продукта и важность проекта.

3.3. Краткий обзор возможностей

Краткая характеристика основных возможностей и функций продукта с точки зрения преимуществ.

3.4. Предположения и зависимости

Описываются предположения, изменение которых приведет к изменению концепции продукта.

3.5. Вопросы затрат и цены

4. Атрибуты функций

Функции имеют атрибуты, предоставляющие дополнительную информацию, которую можно использовать для оценки, отслеживания и определения очередности предлагаемых для реализации элементов разработки, а также управления ими. Нужно описывать в данном разделе только те атрибуты (и их значения), которые вы выберете, чтобы все участники могли лучше понять содержание каждой функции.

4.1. Статус (предложена, принята, включена)

4.2. Приоритет (критический, важный, полезный)

4.3. Уровень трудозатрат

4.4. Риск

4.5. Стабильность

4.6. Целевая версия

4.7. Кому предназначена

4.8. Обоснование

5. Функции продукта

Поскольку документ-концепция изучается широким кругом причастных к проекту лиц и служит основой для достижения соглашения, функции должны

описываться на естественном языке пользователя. Описание функции должно быть кратким и ясным, как правило, одно-два предложения. Для эффективного управления сложностью приложения мы рекомендуем, чтобы описание возможностей любой новой системы (или усовершенствования существующей) производилось на достаточно высоком уровне абстракции и состояло из 25-99 функций. Эти функции составляют основу для определения продукта, а также управления масштабом и проектом в целом. Каждая из них будет описана более подробно в последующих спецификациях.

5.1. Функция 1

5.2. Функция 2

6. Основные варианты использования

Следует описать несколько основных вариантов использования, которые важны для архитектуры или лучше всего помогут читателю понять, как предполагается использовать систему

7. Другие требования к продукту (ограничения)

7.1. Применяемые стандарты

7.2. Системные требования

7.3. Лицензирование и инсталляция

7.4. Требования производительности

8. Требования к документации

В данном разделе описывается, какую документацию необходимо разработать для поддержки успешного внедрения приложения.

8.1. Руководство пользователя

8.2. Интерактивная подсказка

8.3. Руководства по инсталляции, конфигурация и файл ReadMe

8.4. Маркировка и упаковка

9. Глоссарий

Глоссарий описывает все присущие данному проекту термины, в том числе все аббревиатуры, которые могут быть непонятны пользователю или другим читателям данного документа.

Методические рекомендации

Задание выполняется в группе из 3-4 человек

Написать документ-концепцию.

Документ-концепция описывает приложение в общих чертах, а также содержит описания целевых рынков, пользователей системы и функций приложения (как предлагаемых к реализации в версии 1, так и выявленные), ограничения системы.

Примеры контрольных вопросов

1. Что такое функции продукта?
2. Сколько функций достаточно для определения системы?

Критерии оценивания

1. Структура документа-концепции соответствует шаблону - 2 балла, иначе 0 баллов
2. Достаточное описание пользователя и его потребностей (пп 2.2,2.3,2.4) - 3 балла (оценка снижается на 1 балл за каждую ошибку)
3. Рассмотрено не менее 2 альтернатив - 2 балла, 1 альтернатива - 1 балл, иначе 0 баллов
4. В п.3 указано достаточное обоснование важности проекта - 1 балл
5. Указано не менее 5 функций - 2 балла, от 3 до 4 функций - 1 балл, менее 3 функций - 0 баллов

Итого 10 баллов

4.7 Задание 7

Цель лабораторной работы – изучение структуры и написание спецификаций требований к ПО.

Теория

Высококачественный пакет спецификаций должен быть:

- корректным;
- недвусмысленным;
- полным;
- непротиворечивым;
- упорядоченным по важности и стабильности;
- поддающимся проверке (верифицируемым);
- модифицируемым;
- трассируемым;
- понимаемым.

Шаблон спецификаций

Содержание

1. Введение

1.1 Цель

В данном разделе нужно указать цель данной SRS, которая должна полностью описывать внешнее поведение конкретного приложения или подсистемы, а также нефункциональные требования, ограничения проектирования и другие элементы, необходимые для обеспечения всестороннего описания требований к программному обеспечению,

1.2. Масштаб

Данный раздел содержит краткое описание программного приложения (функций иди подсистем, на которые разбита система), для которого создается спецификация; кроме того, описывается, с какой моделью (моделями) вариантов использования оно связано, а также все остальное на что оказывает влияние данный документ.

1.3. Ссылки.

Список ссылок или прилагаемых документов, связанных с данным проектом.

1.4. Предположения и зависимости

В данном разделе описывается техническая достижимость, доступность подсистем или компонентов и другие предположения, от которых может зависеть жизнеспособность описываемого данной SRS программного обеспечения.

2. Краткая характеристика модели вариантов использования

Данный раздел содержит краткую характеристику модели вариантов использования. Она предназначена для тех, кто интересуется поведением системы, — заказчиков, пользователей, архитекторов, авторов вариантов использования, разработчиков, разработчиков вариантов использования, тестологов, менеджеров, ревизоров и авторов документации. Для каждого варианта использования необходимо указать следующее.

• Название варианта использования.

• Краткое описание, объясняющее функцию варианта использования и его роль в системе.

• Перечень акторов данного варианта использования.

• Диаграмма модели вариантов использования. (Здесь следует поместить диаграмму модели вариантов использования в целом.)

3. Характеристика акторов

Здесь описываются все упомянутые в характеристике модели вариантов использования акторы. Для каждого актора следует указать следующее.

* Имя

* Краткое описание

4. Требования

4.1. Функциональные требования

В данном разделе описываются функциональные требования к системе, выраженные на естественном языке. Для многих приложений это достаточно объемная информация, и следует продумать, как организовать данный раздел. Как правило, его организуют По функциям, но можно применять и другие методы, например по пользователям или подсистемам. При использовании для сбора функций вспомогательных средств разработки приложений (инструментальных средств разработки требований, средств моделирования и т.д.) данный раздел документа будет содержать ссылки на эти данные и указывать местоположение и название применяемого для сбора данных инструментального средства.

4.2. Нефункциональные требования

Большая часть нефункциональных требований обычно записывается на естественном языке в данном разделе спецификации. Но нефункциональные требования могут также входить в спецификации вариантов использования.

4.2.1. Практичность

В данный раздел следует включить все требования, влияющие на практичность программного обеспечения. Как правило, указывается следующее.

* Время, необходимое для обучения рядовых пользователей и пользователей с большими полномочиями, чтобы они научились эффективно выполнять определенные действия.

* Время выполнения типичных задач; или же практичность новой системы, сравнивается с практичностью известных систем, которые пользователь знает и любит.

* Требования соответствия общепринятым стандартам практично¬сти, таким как CUA IBM или опубликованные компанией Microsoft стандарты GUI для системы Windows 98.

4.2.2. Надежность

В данном разделе указываются требования к надежности системы.

• Доступность. Указывается, какой процент времени система доступна (xx.xx %), определяются часы использования и доступа для обслуживания, операции при ухудшении параметров системы и т.д.

• Среднее время между отказами (mean time between failures, MTBF). Обычно выражается в часах, но может указываться в днях, месяцах и годах.

• Среднее время восстановления (mean time to repair, MTTR) Сколько времени система может находиться в нерабочем состоянии после сбоя.

• Точность. С помощью некоего известного стандарта указывается требуемая точность (разрешающая способность) выводимой системой информации.

• Максимально допустимый коэффициент ошибок и дефектов. Как правило, выражается как число ошибок, приходящееся на KLOS (тысячу строк кода), или число ошибок, приходящихся на отдельную функцию.

• Доля ошибок или дефектов различных типов. Обычно ошибки разбиваются на следующие категории: незначительные, серьезные и критические. Требования должны определять, что понимается под "критической" ошибкой (такой, как полная потеря данных или невозможность использовать определенную часть функциональных возможностей системы).

4.2.3. Производительность

Здесь описываются характеристики производительности системы. Следует указать время ответа для различных ситуаций. Если требуется, указываются названия соответствующих вариантов использования.

• Время ответа для транзакции (среднее, максимальное)

• Пропускная способность (транзакций в секунду)

• Емкость (число пользователей или транзакций, которые может об¬служить система)

• Режимы снижения производительности (допустимые режимы ра¬боты при ухудшении параметров системы)

• Использование ресурсов (память, диск, каналы связи)

4.2.4. Возможность сопровождения

Данный раздел содержит требования, способствующие улучшению возможности сопровождения и обслуживания создаваемой системы, в том числе стандарты кодирования, определенные соглашения, библиотеки классов, доступ для обслуживания и вспомогательные обслуживающие программы.

5. Требования к интерактивной документации пользователя и системе подсказок

Здесь описываются требования (если таковые имеются) к интерактивной документации пользователя, системе подсказок и т.д.

6. Ограничения проектирования

В данном разделе следует описать все ограничения проектирования создаваемой системы. Ограничения проектирования представляют решения по проектированию, которые являются обязательными и должны быть выполнены. Например, может задаваться язык программирования, требования к программным процессам, а также может предписываться использование определенных средств разработки, архитектурных и проектных ограничений, закупаемых компонентов и библиотек классов.

7. Закупаемые компоненты

В этом разделе описываются все используемые в системе закупаемые компоненты и соответствующие ограничения лицензирования или использования, а также все связанные с ними стандарты совместимости/взаимодействия или интерфейсов.

8. Интерфейсы

В данном разделе определяются интерфейсы, которые должны поддерживаться приложением. Раздел должен содержать достаточно подробное описание протоколов, портов, логических адресов и т.п., чтобы можно было разработать программное обеспечение и проверить его соответствие налагаемым на интерфейсы требованиям.

8.1. Интерфейсы пользователя

Описываются интерфейсы пользователя, которые должны быть реализованы программным обеспечением.

8.2. Аппаратные интерфейсы

Определяются все аппаратные интерфейсы, поддержку которых должно осуществлять программное обеспечение, в том числе логическая структура, физические адреса и ожидаемое поведение.

8.3. Интерфейсы программного обеспечения

Описываются программные интерфейсы с другими компонентами системы программного обеспечения. Это могут быть закупаемые компоненты, повторно используемые компоненты другого приложения или компоненты, разработанные для подсистем, не описываемых данной SRS, но с которыми данное программное приложение должно взаимодействовать:

8.4. Коммуникационные интерфейсы

Описываются все коммуникационные интерфейсы с другими системами или устройствами, такими как локальные сети или удаленные последовательные порты.

9. Требования лицензирования

Определяются все требования лицензирования или другие ограничивающие использование требования, которые оказывают влияние на программное обеспечение.

10. Замечания, касающиеся законности, авторских прав и т.д.

Описываются все необходимые гарантии, все отказы от ответственности, отметки об авторском праве, торговой марке или вопросы соответствия логотипу для программного обеспечения.

11. Применяемые стандарты

Посредством ссылок указываются все стандарты (а также конкретные их разделы), которые применяются к описываемой системе. Например, это могут быть стандарты качества, некие законы или инструкции, а также отраслевые стандарты практичности, взаимодействия, интернационализации, соответствия операционной системы и т.д.

Индекс

Наличие индекса помогает читателю определять местонахождение в документе ключевых понятий и тем.

Глоссарий

Здесь описываются все термины данного приложения, а также все определения и принятые в проекте или компании сокращения, которые необходимы для понимания данного документа и приложения.

Приложения

Методические рекомендации

Разработать спецификации системы по указанному шаблону.

Пункт 4.1 нужно писать так, чтобы если бы эту задачу поручили вам, то вам будет понятно, что нужно сделать. Если пунктов в 4.1 много, можно сделать подробную спецификацию только пяти из них. В остальных указать …

В пункте 4.2 делайте реалистичные предположения.

Неиспользуемые пункты верхнего уровня (с номером) оставьте, но укажите "нет" или "не требуется".

Разделите пункты спецификации между участниками команды для ускорения работы.

Примеры контрольных вопросов

1. Что такое трассируемость требований?
2. Чем отличается функциональное требование от нефункцонального?

Критерии оценивания

1. Структура спецификации соответствует шаблону - 2 балла, иначе 0 баллов
2. Выполняется трассируемость функций из документа концепции - 1 балл
3. Имеется подробная спецификация не менее 5 функциональных требований - 2 балла, 3-4 требования - 1 балл, менее 3 - 0 баллов
4. Имеется подробная спецификация не менее 3 нефункциональных требований - 2 балла, 2 требования - 1 балл, , менее 2 - 0 баллов
5. Указана спецификация для интерфейсов пользователя (не менее 3 диалогов) - 2 балла, 2 диалога - 1 балл, , менее 2 - 0 баллов
6. Указана спецификация для хотя бы 1 программного интерфейса (API) - 1 балл

Итого 10 баллов

4.8 Задание 8

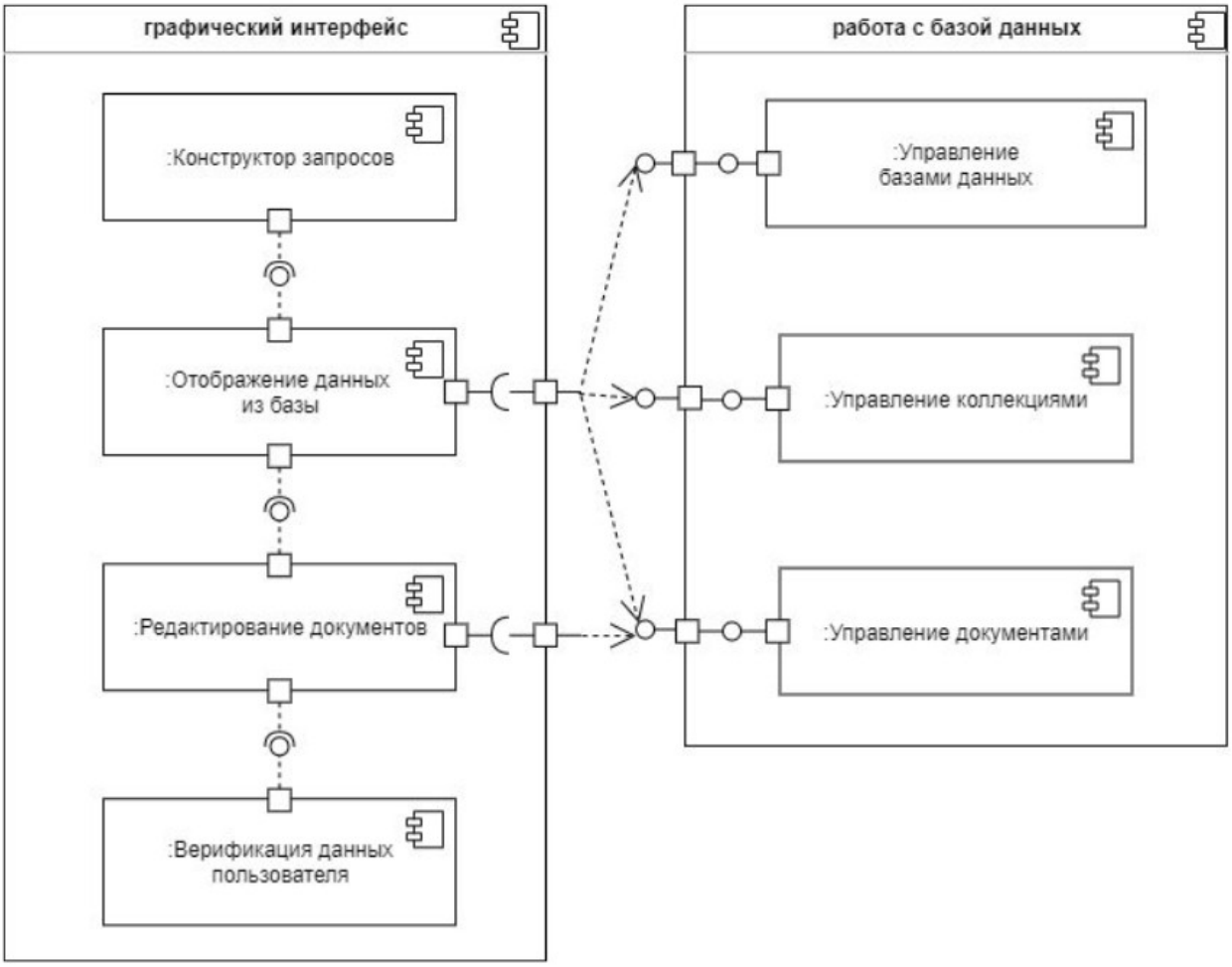
Цель лабораторной работы – выполнить проектирование архитектуры приложения, использующего ИИ.

Методические рекомендации

1. Разработать архитектуру систему с использованием ИИ
2. Выбрать язык программирования
3. Выбрать фреймворк, библиотеки для реализации интерфейса пользователя Выбрать СУБД
4. Нарисовать диаграмму компонентов (подсистем)

Пример

1 Был выбран язык программирования C++, так как:
● C++ позволяет создавать кроссплатформенные приложения;
● Программы, созданные на C++ имеют низкие системные требования;
● Является одним из самых популярных языков, что положительно сказывается на поддержке ПО;
● Имеется библиотека для взаимодействия с MongoDB.
2 Для реализации интерфейса пользователя используется фреймворк Qt
● Один из самых популярных UI фреймворков;
● QtCreator позволяет создавать интерфейс в графическом режиме;
● Имеет большое сообщество и множество готовых библиотек.
3. В качестве СУБД была выбрана MongoDB так как:
● Позволяет создавать коллекции документов с гибкой структурой;
● Поддерживает массивы, бинарные данные, строки, числа и другие типы данных;
● Реализует принципы ACID;
● Поддержка многопоточности;
● Поддерживает потоки событий об изменении данных в базе данных;
● Близка к представлению данных в реальном мире.
4 Подсистемы:
1. Графический интерфейс
2. Работа с базой данных
5. Выделить компоненты
● Графический интерфейс: отображение данных из базы, конструктор запросов, редактирование документов, верификация данных пользователя.
● Работа с базой данных: управление базами данных; управление коллекциями; управление документами.



Примеры контрольных вопросов

1. Какие критерии были лля выбора фреймворка (библиотеки)?
2. Как обозначается компонент в диаграмме UML?

Критерии оценивания

1. Выбран язык программирования - 1 балл
2. Выбран фреймворк, библиотеки для реализации интерфейса пользователя - 2 балла, устаревший фреймворк - 1 балл, не выбран - 0 баллов
3. Выбрана СУБД - 1 балл
4. Сделано обоснование выбора - 1 балл
5. Нарисована диаграмма компонентов (подсистем) - 1 балл
6. Обозначения на диаграмме соответствуют стандарту UML - 2 балла, оценка снижается на 1 балл за каждую ошибку
7. Подсистемы и компоненты обеспечивают выполнение требований (функций) ПО - 2 балла, , оценка снижается на 1 балл за каждую ошибку

Итого 10 баллов

4.9 Задание 9

Цель лабораторной работы – разработка диаграммы классов проекта .

Теория

Традиционно принцип единственной ответственности (Single Responsibility Principle, SRP) описывался так: модуль должен иметь одну и только одну причину для изменения.

Программное обеспечение изменяется для удовлетворения нужд пользователей и заинтересованных лиц. Пользователи и заинтересованные лица как раз и есть та самая «причина для изменения», о которой говорит принцип .

Фактически принцип можно перефразировать так: Модуль должен отвечать за одного и только за одного пользователя или заинтересованное лицо.

Более правильным выглядит понятие группы, состоящей из одного или нескольких лиц, желающих данного изменения . Мы будем называть такие группы акторами (actor).

Соответственно, окончательная версия принципа единственной ответственности выглядит так: модуль должен отвечать за одного и только за одного актора.

Принцип открытости/закрытости (Open-Closed Principle; OCP) гласит: программные сущности должны быть открыты для расширения и закрыты для изменения.

Иными словами, должна иметься возможность расширять поведение программных сущностей без их изменения.

Мы должны избегать зависимости от неустойчивых конкретных элементов системы. То есть от модулей, которые продолжают активно разрабатываться и претерпевают частые изменения.

Как следствие, стабильными называются такие архитектуры, в которых вместо зависимостей от переменчивых конкретных реализаций используются зависимости от стабильных абстрактных интерфейсов. Это следствие сводится к набору очень простых правил:

- Не ссылайтесь на изменчивые конкретные классы. Ссылайтесь на абстрактные интерфейсы. Это правило применимо во всех языках, независимо от устройства системы типов. Оно также накладывает важные ограничения на создание объектов и определяет преимущественное использование шаблона «Абстрактная фабрика».

- Не наследуйте изменчивые конкретные классы.

- Не переопределяйте конкретные функции. Конкретные функции часто требуют зависимостей в исходном коде. Переопределяя такие функции, вы не устраняете эти зависимости — фактически вы наследуете их. Для управления подобными зависимостями нужно сделать функцию абстрактной и создать несколько ее реализаций.

- Никогда не ссылайтесь на имена конкретных и изменчивых сущностей.

Принцип ацикличности зависимостей (ADP): циклы в графе зависимостей компонентов недопустимы.

Принцип устойчивых зависимостей (SDP): зависимости должны быть направлены в сторону устойчивости.

Принцип устойчивости абстракций: устойчивость компонента пропорциональна его абстрактности.

Принцип устойчивости абстракций (SAP) устанавливает связь между устойчивостью и абстрактностью. С одной стороны, он говорит, что устойчивый компонент также должен быть абстрактным, чтобы его устойчивость не препятствовала расширению, с другой — он говорит, что неустойчивый компонент должен быть конкретным, потому что неустойчивость позволяет легко изменять его код. То есть стабильный компонент должен состоять из интерфейсов и абстрактных классов, чтобы его легко было расширять.

Методические рекомендации

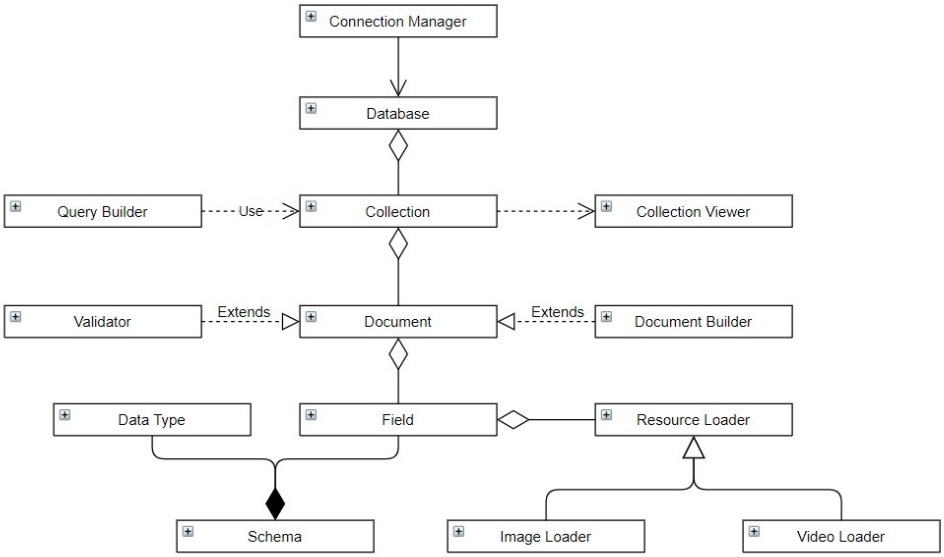
Задание выполняется в группе из 3-4 человек

Разработайте иерархии классов проекта, используя принципы проектирования.

Не делайте детализацию, только название классов.

Для выполнения задания можно использовать https://app.diagrams.net/ (aka draw.io)

Пример



Примеры контрольных вопросов

1. Как обозначается агрегация?
2. Как обозначается абстрактный класс?

Критерии оценивания

1. Разработана диаграмма классов - 3 балла, иначе 0 баллов
2. Правильно использованы обозначения UML - 3 балла, оценка снижается на 1 балл за каждую ошибку

3. Каждому компоненту (задание 5) соответствует хотя бы один класс - 2 балла, отсутствует 1 компонент - 1 балл, отсутствует более 1 компонента - 0 баллов
4. Направление связей в иерархии классах обеспечивают устойчивость компонент - 2 балла, иначе 0 баллов

Итого 10 баллов

4.10 Задание 10

Цель лабораторной работы – разработка детализированной диаграммы классов и спецификации модулей.

Теория

Диаграмма классов демонстрирует общую структуру иерархии классов системы, их атрибутов (полей), методов, интерфейсов и взаимосвязей между ними.

Для классов и связей между ними используются обозначения на рисунке 3.

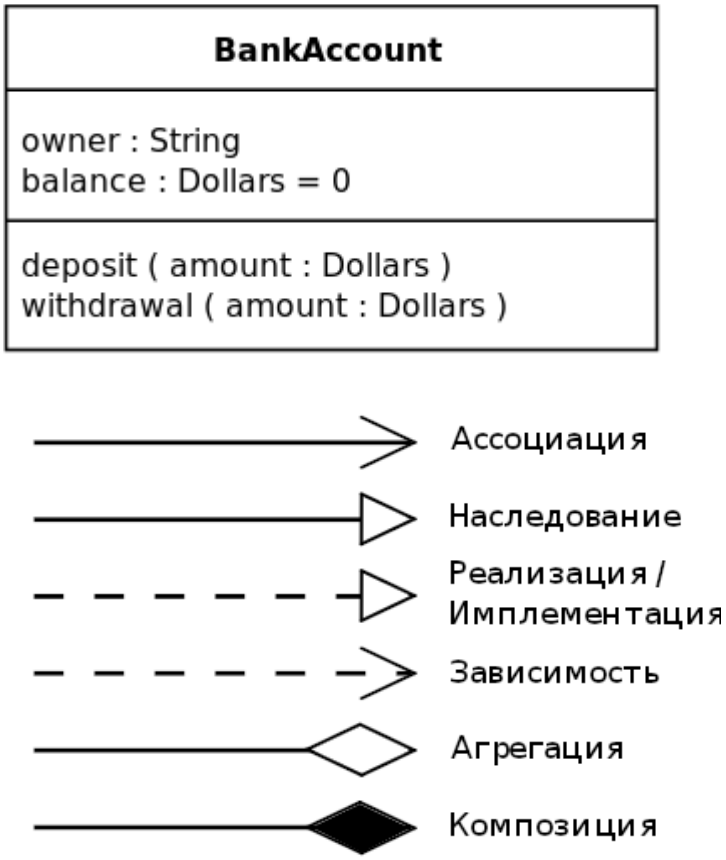


Рисунок 3 — Обозначения UML для классов и связей.

Формат описания поля:
- имя : тип [кратность] = нач_значение {ограничения}
ограничения для полей:

readOnly

id

для полей с кратностью ordered

unique nonunique
подчеркивание имени поле является static
Формат описания метода:
видимость имя(параметры):тип_результата {ограничения}
видимость (+ - #) может комбинироваться с / если метод переопределен
ограничения для методов: query - не изменяет состояния объекта
курсив - чисто виртуальный метод

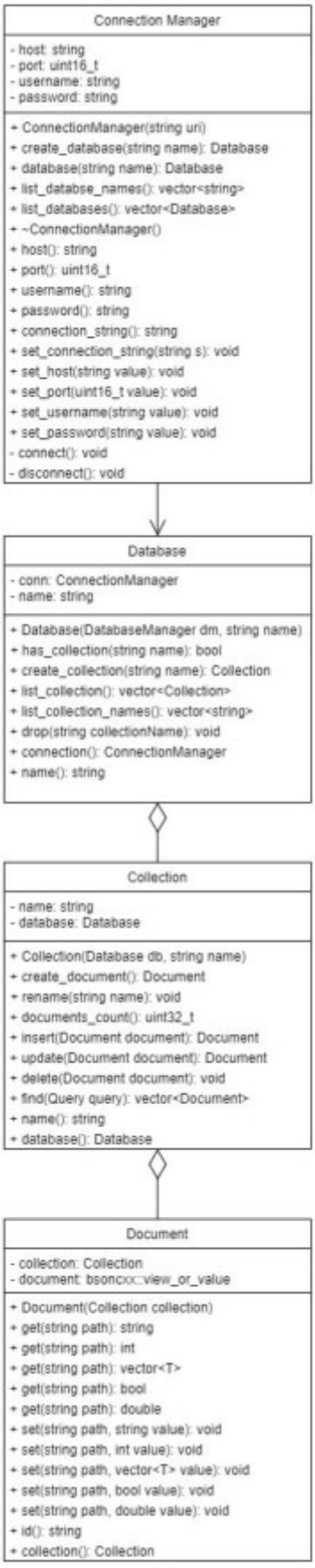
Методические рекомендации
Задание выполняется в группе из 3-4 человек
1. Выбрать классы одной компоненты, нарисовать диаграмму классов для них.
2. Для нескольких методов написать спецификацию (вход, выход, исключения)

Пример
public Document Collection::insert(Document document) – добавляет документ “document” в коллекцию и возвращает тот же документ с установленным базой данных “id”.
Параметры:
document – документ для добавления в коллекцию
Исключения: DocumentAlreadyExistsException

public void Document::set(string path, string value) – устанавливает значение “value” по указанному пути “path” в документе.
Параметры:
path - разделяемый "." путь до поля. Например: “user.address.city”
value - значение поля
Исключения: нет

private void ConnectionManager::connect() - устанавливает подключение к базе данных по заданным параметрам.
Исключения: NoSuchHostException, AuthenticationFailedException

public vector<Database> ConnectionManager::list_databases() - возвращает список баз данных, либо пустой список, если их нет.
Исключения: NoConnectionException



Примеры контрольных вопросов

1. Какой формат описания метода?

2. Как указать видимость элемента?

Критерии оценивания

1. Нарисована диаграмма классов (не менее 2 классов) - 3 балла, иначе 0 баллов
2. Использованы корректные графические обозначения UML для классов и связей - 2 балла, оценка снижается на 1 балл за каждую ошибку
3. В диаграмме классов поля имеют правильные описания - 1 балл
4. В диаграмме классов методы имеют правильные описания - 1 балл
5. Указаны спецификации не менее 3 методов - 3 балла (по 1 баллу за каждую спецификацию, но не более 3 баллов)

Итого 10 баллов

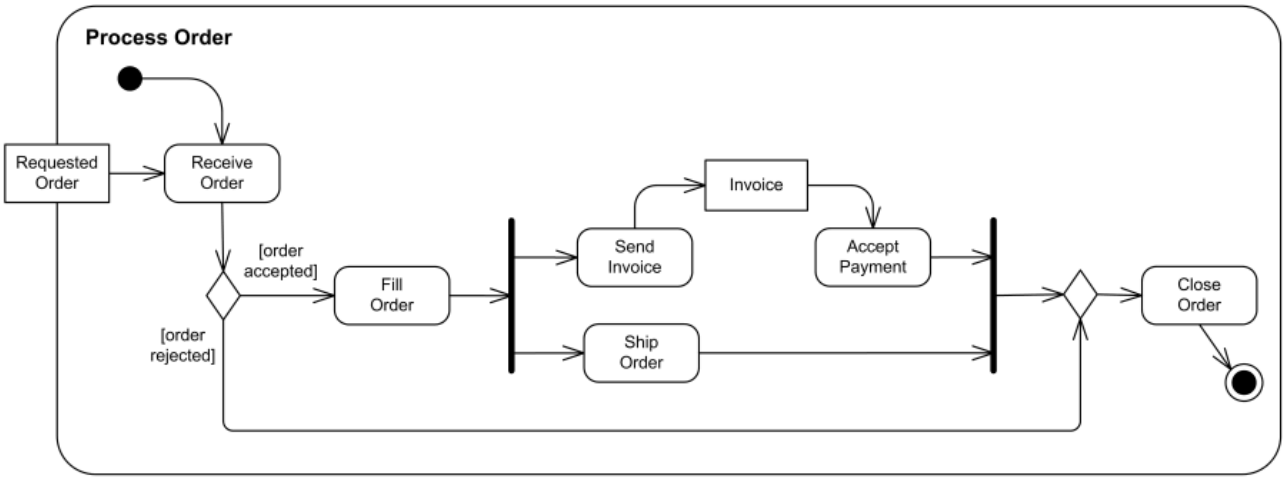
4.11 Задание 11

Цель лабораторной работы – разработка диаграммы, описывающей поведение.

Методические рекомендации

Для учебного проекта нарисовать диаграмму деятельности или диаграмму последовательности

Пример



Примеры контрольных вопросов

1. Как обозначается ветвление?
2. Как обозначается распараллеливание?

Критерии оценивания

1. Нарисована диаграмма - 3 балла, иначе 0 баллов
2. Использованы правильные обозначения UML - 4 балла, оценка снижается на 1 балл за каждую ошибку
3. Соответствует выбранному процессу - 3 балла, оценка снижается на 1 балл за каждую ошибку

Итого 10 баллов

4.12 Задание 12

Цель лабораторной работы – получение практических навыков по работе с системой контроля версий.

Теория

Глава	Паттерн	Решаемая задача	Решение
Паттерны для представления данных	Хешированный признак (Hashed Feature)	Задачи, связанные с категориальными признаками, такими как неполный словарь, размер модели вследствие кардинальности и холодный пуск	Группировать детерминированный и переносимый хеш строкового значения на корзины и делать уступку в пользу наличия коллизий в представлении данных
	Векторные вложения (Embeddings)	Признаки высокой кардинальности, в которых важно сохранить отношения близости	Усваивать представление данных, которое отображает данные высокой кардинальности, в низкоразмерное пространство таким образом, что сохранится информация, относящаяся к задаче обучения
	Синтетический признак (Feature Cross)	Сложность модели недостаточна для обучения связей между признаками	Помогать моделям быстрее усваивать связи между входными переменными, в явной форме делая каждую комбинацию входных значений отдельным признаком
	Мультимодальный вход (Multimodal Input)	Как выбрать между несколькими потенциальными представлениями данных	Конкатенировать все имеющиеся представления данных
Паттерны для представления задачи	Переформулировка (Reframing)	Несколько задач, включая уверенность в численном предсказании, порядковых категориях, ограничении диапазона предсказания и мультиметочном обучении	Менять представление выходных данных в задаче машинного обучения; например, представлять регрессионную задачу в качестве классификационной (и наоборот)
	Мультиметка (Multilabel)	К данному тренировочному примеру применяется более одной метки	Кодировать метку, используя массив значений с несколькими активными состояниями, и использовать *k* сигмоид в качестве выходного слоя
	Ансамбли (Ensembles)	Компромисс между смещенностью и дисперсией в задачах малого и среднего масштаба	Совмещать несколько моделей машинного обучения и агрегировать их результаты в целях генерирования предсказаний
	Каскад (Cascade)	Проблемы технической сопроводимости или дрейфа, когда ML-задача разбивается на ряд ML-подзадач	Трактовать систему машинного обучения как единый рабочий поток для процессов тренировки, оценивания и предсказания
	Нейтральный класс (Neutral Class)	Метка класса для некоторого подмножества примеров по существу является произвольной	Вводить дополнительную метку для классификационной модели, не пересекающуюся с текущими метками
	Перебалансировка (Rebalancing)	Сильно несбалансированные данные	Выполнять понижающий отбор, повышающий отбор либо использовать функцию взвешенной потери в зависимости от разных соображений

Паттерны для тренировки моделей	Полезное переобучение (Useful Overfitting)	Использование методов машинного обучения для обучения физической модели или динамической системы	Отказываться от обычных технических приемов обобщения ради намеренной переподгонки на тренировочном наборе данных
	Контрольные точки (Checkpoints)	Потеря несохраненных результатов работы во время длительных тренировочных заданий из-за аварийного отказа машины	Периодически сохранять полное состояние модели, чтобы частично натренированные модели были доступны и могли использоваться для возобновления тренировки не с нуля, а с промежуточной точки
	Трансферное обучение (Transfer Learning)	Отсутствие крупных наборов данных, необходимых для тренировки сложных моделей машинного обучения	Брать часть ранее натренированной модели, замораживать веса и использовать эти нетренируемые слои в новой модели, которая решает аналогичную задачу
	Распределительная стратегия (Distribution Strategy)	Тренировка крупных нейронных сетей может занимать очень много времени, что замедляет экспериментирование	Выполнять цикл тренировки в требуемом масштабе на нескольких воркерах, используя преимущества кэширования, аппаратного ускорения и параллелизации
	Гиперпараметрическая настройка (Hyperparameter Tuning)	Как определять оптимальные гиперпараметры модели машинного обучения	Вставлять цикл тренировки в метод оптимизации с целью отыскания оптимального набора модельных гиперпараметров
Паттерны обеспечения воспроизводимости	Преобразователь (Transform)	Данные на входе в модель должны быть преобразованы для создания ожидаемых моделью признаков, и этот процесс должен быть согласованным между тренировкой модели и обработкой запросов	В явной форме захватывать и сохранять преобразования, применяемые для конвертирования входных модельных данных в признаки
	Повторяемая разбивка (Repeatable Splitting)	При создании срезов данных важно иметь метод, который остается легковесным и воспроизводимым независимо от языка программирования или случайных начальных чисел	Выявлять столбец, который улавливает корреляционную связь между строками, и использовать алгоритм хеширования Farm Fingerprint для разбивки имеющихся данных на тренировочный, валидационный и тестовый наборы данных
	Мостовая схема (Bridged Schema)	По мере появления новых данных любые изменения в схеме данных могут помешать использованию как новых, так и старых данных для перетренировки	Адаптировать данные из старой, изначальной схемы данных в соответствии со схемой новых, более совершенных данных
Ответственный искусственный интеллект	Эвристический эталон (Heuristic Benchmark)	Объяснение результативности модели с помощью сложных метрик оценивания не дает интуитивного понимания, необходимого лицам, принимающим бизнес- решения	Сравнивать ML-модель с простой и понятной эвристикой
	Объяснимые предсказания (Explainable Predictions)	Иногда для отладки либо в целях регулирования и надзора за соблюдением стандартов необходимо знать причину, почему модель делает те или иные предсказания	Применять технические приемы обеспечения объяснимости моделей, чтобы понимать, как и почему модели делают предсказания, и повышать доверие пользователей к системам машинного обучения
	Призма объективности (Fairness Lens)	Искаженность может приводить к тому, что ML-модели не будут одинаково относиться ко всем пользователям и могут иметь неблагоприятные последствия для некоторых групп населения	Использовать инструменты выявления искаженностей наборов данных перед тренировкой и оценивать натренированные модели через призму объективности с целью обеспечения объективности модельных предсказаний для разных групп пользователей и разных сценариев

Методические рекомендации

Выполнить детальное проектирование компонента интеллектуальной системы с использованием паттернов проектирования

Варианты

1. Выбор тарифного плана данного оператора сотовой связи
2. Выбор специальности обучения в университете
3. Выбор блюд в ресторане
4. Рекомендательная система для фильмов

Примеры контрольных вопросов

1. Какой паттерн проектирования был использован?
2. Почему?

Критерии оценивания

1. Определен метод для получения результата — 2 балла, иначе 0 баллов
2. Определен способ интеграции компонента в систему— 2 балла, иначе 0 баллов
3. Интерфейс модулей соответствует задаче – 3 балла, оценка снижается на 1 балл за каждую ошибку в описании
4. Использованы паттерны проектирования – 3 балла (1 балл за каждый паттерн, но не более 3 баллов)

Итого 10 баллов

5. Учебно-методическое и информационное обеспечение дисциплины

5.1 Основная литература

1. Новиков, Ф.А. Учебно-методическое пособие по дисциплине «Анализ и проектирование на UML». [Электронный ресурс] — Электрон. дан. — СПб. : НИУ ИТМО, 2007. — 286 с. http://e.lanbook.com/book/43540

2. Джесутасан, Р. Реинжиниринг бизнеса: Как грамотно внедрить автоматизацию и искусственный интеллект / Р. Джесутасан ; перевод с английского Е. Милицкая. — Москва : Альпина Паблишер, 2019. — 278 с. — ISBN 978-5-9614-2634-2. — https://e.lanbook.com/book/140499

3. Бурков, А. Инженерия машинного обучения / А. Бурков ; перевод с английского А. А. Слинкина. — Москва : ДМК Пресс, 2022. — 306 с. — ISBN 978-5-93700-125-2. — https://e.lanbook.com/book/314834

4. Косяков, А. Системная инженерия. Принципы и практика. [Электронный ресурс] / А. Косяков, У. Свит. — Электрон. дан. — М. : ДМК Пресс, 2014. — 624 с. http://e.lanbook.com/book/66484

5.2 Дополнительная литература

1. Станкевич, Л. А. Интеллектуальные системы и технологии : учебник и практикум для вузов / Л. А. Станкевич. — 2-е изд., перераб. и доп. — Москва :

Издательство Юрайт, 2023. — 495 с. — (Высшее образование). — ISBN 978-5-534-16238-7. — https://urait.ru/bcode/530657

2. Буч, Г. Язык UML. Руководство пользователя. [Электронный ресурс] / Г. Буч, Д. Рамбо, И. Якобсон. — Электрон. дан. — М. : ДМК Пресс, 2008. — 496 с. http://e.lanbook.com/book/1246