МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г.Локтионова
«26» 04 2024г.

Управление проектами разработки систем искусственного интеллекта

Методические указания по выполнению лабораторных работ по дисциплине «Управление проектами разработки систем искусственного интеллекта»
для обучающихся по направлению подготовки 09.04.01 Информатика и вычислительная техника программа "Киберфизические системы и искусственный интеллект" направленность (профиль, специализация) "Облачная и сетевая инфраструктура систем искусственного интеллекта "

Курск 2024

УДК 004.82 (075.8)
Составитель Т.И.Лапина

Рецензент
Кандидат технических наук, доцент Е.А.Петрик

Управление проектами разработки систем искусственного интеллекта: методические указания по выполнению лабораторных работ по дисциплине «Управление проектами разработки систем искусственного интеллекта» / Юго-Зап. гос. ун-т; сост.: Т. И. Лапина, Курск, 2022. 64с. ил. 44, табл.4, Билиогр.: с.64.

Содержат краткие теоретические сведения о методах разработки требований к проекту информационных систем. Большое внимание уделено проблемам стандартов и профилей информационных систем. Подробно описаны различные методологические подходы к проектированию ИС и соответствующие этим подходам инструментальные средства (Bpwin, Rational Rose)

Предназначены для студентов направления подготовки 09.04.01 Информатика и вычислительная техника программа "Киберфизические системы и искусственный интеллект" направленность (профиль, специализация) "Облачная и сетевая инфраструктура систем искусственного интеллекта "

Текст печатается в авторской редакции
Подписано в печать 17.01.24 Формат 60x84 1/16.
Усл. печ. л.0,9 . Уч. – изд. л.0.8 .Тираж 100 экз. Заказ. 41
Бесплатно.
Юго - Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Лабораторная работа №1

Разработка требований к проекту СИИ

Цель работы: исследование предметной области разрабатываемой системы искусственного интеллекта и разработка требований к проекту.

1. Теоретические сведения

Проводится исследование предметной области разрабатываемой системы. Составляется словесное описание действующих лиц, их ха-рактеристик и процессов взаимодействия, а также других объектов и их свойств. Взаимодействие действующих лиц описывается поверхностно, без серьёзной детализации. Результатом исследования является нефор-мальное описание предметной области.

Словарь предметной области – документ, в котором содержатся определения всех терминов, использующихся в рамках данной пред-метной области. Словарь предметной области составляется на основе её неформального описания.

На основе неформального описания предметной области и словаря предметной области определяются концептуальные классы системы. Концептуальные классы описывают объекты реального мира и их свой-ства, которые описываются атрибутами классов.

Объекты, моделями которых являются определённые концепту-альные классы, в реальном мире взаимодействуют между собой. По-этому после определения концептуальных классов необходимо при по-мощи неформального описания предметной

области определить зави-симости между ними. Совокупность концептуальных классов и зависи-мостей между ними составляет статическую модель предметной облати, которая визуально представляется в виде диаграммы концептуальных классов.

На основе статической модели предметной области формируются требования к информационной части системы. Они определяют модель данных, которая будет использоваться в системе. Требования следует записывать в декларативной форме, то есть каждое требование определяет, что должна делать система или другие действующие лица. Все требования собираются в документ, на основе которого в дальнейшем происходит разработка программного продукта.

2. Пример разработки требований

Тема: «Программа для моделирования работы лифта».

Неформальное описание предметной области

Моделирование работы лифта является достаточно популярной задачей в программировании. Система управления лифтом представляет собой сложную взаимосвязь многих элементов: пассажир, этаж от-правления, направление, этаж назначения и, собственно, сам лифт, который в свою очередь можно разбить на множество мелких деталей: кабина лифта, двигатель, двери, противовесы, с помощью чего лифт может опускаться и подниматься. Кроме того, управление лифтом может осуществляться как извне пассажирами на этажах, так и изнутри с помощью панели управления.

Рисунок 1 - Кнопочная панель управления Лифт - разновидность грузоподъёмной машины, предназначенная для вертикального перемещения грузов.

При работе лифта нужно учитывать количество этажей в здании, их высоту, грузоподъемность лифта. Перемещение в реальном времени можно отследить с помощью дисплея, отображающего состояние лифта.

Рисунок 2 - Изображение лифта

Словарь объектов предметной области приложения

На основе анализа описания предметной области были определены основные понятия и разработан словарь объектов предметной области, представленный в таблице 1

Таблица 1 – Словарь предметной области

Понятие	Описание
Лифт	Лифт - разновидность грузоподъёмной машины, предназначенная для вертикального или наклонного перемещения грузов.
Этаж	Этаж - часть пространства здания между двумя горизонтальными перекрытиями (между полом и потолком).
Запрос	Запрос - обращение на предмет выполнения устанавливаемых требований.
Кабина	Кабина - закрытое грузонесущее устройство, предназначенное для транспортировки пассажиров и грузов
Грузоподъёмность	Грузоподъёмность — масса груза, на перевозку которого рассчитано транспортное средство.
Направление	Направление движения — направление (возможно, усреднённого) вектора скорости объекта
Пассажир	Пассажир – человек, совершающий поездку.
Панель управления	Панель управления лифта – это специальное устройство, которое расположено внутри кабины лифта и которое предназначено для управления и контроля движения кабины лифта.

Моделирование предметной области с помощью диаграммы классов

На основе анализа словаря предметной области приложения и описания вариантов использования были определены классы, атрибуты и методы классовотношения классов, представленные на рисунке 3 в виде диаграммы концептуальных классов.

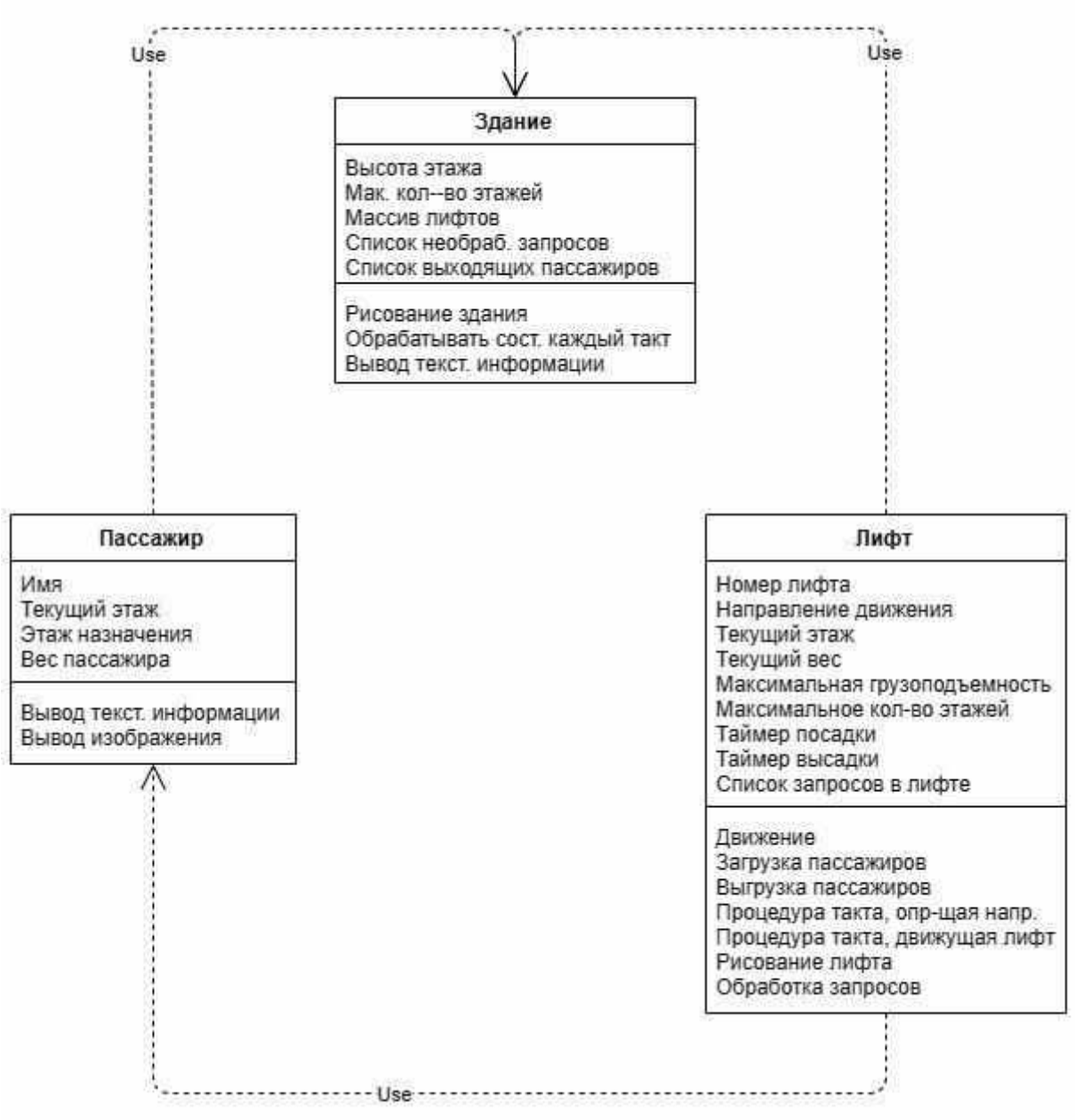

Рисунок 3 – Диаграмма концептуальных классов предметной области программы для моделирования работы лифта

Описание класса Здание:

– Общедоступные поля класса:

int height_floor – высота этажа.

int max_floor – максимальное количество этажей.

Lift[] mas_lift – массив лифтов.

List<Request> List_Request0 – список необработанных запросов (пассажиры ждут лифт на своих этажах).

List<Request> List_Request_2 – список запросов, отработанных лифтом, пассажиры на выходе из лифта.

– Общедоступные методы класса:

Draw(Graphics g) – рисует здание, пассажиров, информацию о них. Метод ничего не возвращает.

Master_tick() – процедура обработки лифтов каждый такт, использует методы Lift_tick1() и Lift_tick2(). Метод ничего не озвращает.

String_To() – метод, который позволяет преобразовать в строку и количество необработанных запросов. Возвращает строку s.

Описание класса Пассажир:

– Внутренние поля класса:

string name – имя пассажира.

– Общедоступные поля класса:

int first_floor – Текущий этаж – местоположение вызова.

int last_floor – этаж назначения.

int cur_weight – вес пассажира.

int time_counter – счетчик времени.

– Общедоступные методы класса:

String_To() – метод, который преобразует в строку параметры запросы с возможностью дальнейшего вывода на экран. Возвращает строку s.

Draw(Graphics g, Zdanie zdanie, int k) – отрисовка пассажиров с их параметрами, основываясь на списке запросов. Метод ничего не возвращает.

Draw_2(Graphics g, Zdanie zdanie, int k, int time) – метод, позволяющий нарисовать динамическое отображение пассажира. Метод ничего не возвращает.

Описание класса Лифт:

– Внутренние поля класса:

int number – номер лифта.

enum Direction – направление движения лифта.

int cur_floor – текущий этаж лифта.

byte cur_direction – текущее направление движения.

int cur_weight – текущий вес груза.

int max_weight – максимальная грузоподъемность.

int max_floor – максимальное количество этажей.

int loading_timer – таймер во время посадки.

int unloading_timer – таймер во время высадки.

List<Request> List_Request - список запросов в лифте (лифт их забрал,но еще не отвез).

– Общедоступные методы класса:

Move1() – Метод, изменяющий текущий этаж лифта на 1 единицу. Метод ничего не возвращает

Load_pass(Request req) – Загрузить пассажиров, при этом увеличить вес груза, добавить в список запросов в лифте. Возвращает переменную логического типа. Если «true» пассажир заходит, в противном случае остается на этаже.

Unload_pass(Request req) – Выгрузить пассажира, снизить вес груза, добавить в список пассажиров, выходящих из лифта. Метод ничего не возвращает.

String_To() – метод, который преобразует в строку параметры лифта с возможностью дальнейшего вывода на экран. Возвращает строку

s.Draw(Graphics g, Zdanie zdanie) – рисовать лифт. Метод ничего не возвращает.

Lift_tick1() - Процедура такта, которая использует метод Work().Метод ничего не возвращает.

Lift_tick2() – Процедура такта, которая использует метод Move1(), т.е. двигает лифт. Метод ничего не возвращает.

Work() – Метод, в котором заключается основная логика программы, по которой лифты принимают то или иное решение о движении.

3. Задание

По описанной задаче разработать требования к информационной части программно-информационной системы ИИ (отчёт в виде документа MS Word):

4. Контрольные вопросы

1. В чем заключается методика предпроектного обследования?
2. Какие существуют универсальные методы, пригодные для обследования всех функциональных звеньев предприятия?
3. Какие существуют характеристики документа?
4. Каким образом производится кодирование полученной документации?
5. На каких уровнях проводится обследование аспектов деятельности предприятий?
6. Что включает информационная база данных?
7. В каких направлениях выполняется информационный анализ предметной области?
8. Цель анализа полученной информации.

Лабораторная работа №2

Управление проектами СИИ на основе гибкой методологии AGILE

Цель работы: **О**свршение приемов использования гибкой методологии AGILE **для** азрабатываемой системы искусственного интеллекта и разработка требований к проекту.

2. Теоретические сведения

Гибкая методология разработки (от англ. - Agile software development) - манифест, определяющий способ мышления и содержащий основные ценности и принципы, на которых базируется несколько подходов (фреймворков, от англ. framework — каркас, структура) к разработке программного обеспечения (хотя в последнее время идет тенденция и попытки применения гибкой методологии разработки к иным направлениям деятельности, не только в части информационных технологий), подразумевающих под собой интерактивную разработку, периодического (динамического) предоставления (обновления) требований от Заказчика и их реализацию посредством самоорганизующихся рабочих групп, сформированных из

экспертов различного профиля (разработчики, тестировщики, внедренцы и т.д.). Такой перевод Agile, как "гибкая методология разработки" не совсем корректен т.к. обычно Agile не называют методологией, а вот подходы на основе данного манифеста и есть

методологии, но с точки зрения Agile их называют - фреймворки. На данный момент существует множество фреймворков (методологий),

подходы которых базируются на гибкой методологии разработки, например такие, как: Scrum, Extreme programming, FDD, DSDM и т.д.

Это набор итеративных подходов к разработке программного обеспечения, в которых требования и решения возникают в результате сотрудничества между самоорганизующимися межфункциональными командами. Agile-методы или Agile-процедуры часто поддерживают дисциплинированную методологию управления проектами, которая

поощряет регулярные проверки и адаптацию, а также философию лидерства, поощряющую командную работу, самоорганизацию и подотчетность. Также требуется набор передовых инженерных методов для обеспечения быстрой доставки высококачественного программного обеспечения и бизнес-стратегии, связывающей разработку с потребностями клиентов и целями компании. Гибкая разработка требует, чтобы все процессы разработки были согласованы с концепциями **Agile Manifesto.**

Agile можно определить как сотрудничество заинтересованных лиц (stakeholders) для поставки ценности заказчику с частыми инкрементами, при постоянном размышлении (reflection) и адаптации. Это определение фокусируется на характеристиках, присущих любому agile окружению, а именно:

- Сотрудничество – как люди работают вместе, включая ко-

манду, занимающуюся разработкой, и заинтересованных лиц (stakeholders)

- Поставка ценности – цель прилагаемых усилий – это постав-

ка ценности заказчикам, что бы это ни было: программное обеспечение, более эффективные процессы или новые продукты.

- Частые инкременты – команда поставляет ценность каждые несколько дней, недель или месяцев, а не единожды, в конце проекта.
- Постоянное размышление и адаптация – проектная команда размышляет над подходами и проектом на регулярной основе, и настраивает свою работу в соответствии со сделанными выводами.

Agile – одна из методологий итеративной и пошаговой разработки ПО, в противоположность традиционной линейной методологии «водопад». Методология гибкой разработки определяет систему методов проектирования, разработки и тестирования на протяжении всего жизненного цикла ПО. Методы гибкой разработки (например, SCRUM) основаны на оперативном реагировании на изменения за счет применения адаптивного планирования, совместной выработки требований, рационализации самоорганизующихся кросс-функциональных групп разработчиков, а также пошаговой разработки ПО с четкими временными рамками. Этот подход используется во многих современных проектах разработки коммерческого ПО. В основе гибкой методологии разработки лежит либерально-демократический подход к управлению и организации труда команд, члены которой сконцентрированы на разработке конкретного программного обеспечения.

За счет того, что разработка программного обеспечения с применением гибкой методологии определяет серии коротких циклов (итераций), с длительностью 2-3 недели, достигается минимизация рисков т.к. по завершению каждой итерации Заказчик принимает результаты и выдает новые или корректирующие требования т.е.

контролирует разработку и может на неё сразу влиять. Каждая итерация включает в себя этапы планирования, анализа требований, проектирование, разработку, тестирование и документирование. Обычно одной итерации не достаточно для выпуска полноценного программного продукта, но при этом по окончании каждого этапа разработки должен появляться "осязаемый" продукт или часть функционала, которую можно посмотреть, протестировать и выдать дополнительные или корректирующие меры. На основе проделанной работы, после каждого этапа, команда подводит итоги и собирает новые требования, на основании чего вносит корректировки в план разработки программного обеспечения.

Одной из основных идей Agile, является взаимодействие внутри команды и с заказчиком лицом к лицу, что позволяет быстро принимать решения и минимизирует риски разработки программного обеспечения, поэтому команду размещают в одном месте, с географической точки зрения. Причем в команду входит представитель заказчика (англ. product owner - полномочный представитель заказчика или сам заказчик, представляющий требования к продукту; такую роль выполняет менеджер проекта от заказчика или бизнес-аналитик).

Agile-манифест разработки программного обеспечения

«Манифест гибкой методологии разработки программного обеспечения» был выпущен и принят в феврале 2001 года (штат ЮТА США, лыжный курорт The Lodge at Snowbird) группой экспертов. Данный манифест определяет 4 основные ценности и 12 принципов для методологий, базирующихся на нем, а также дает альтернативное

видение подхода к разработке программного обеспечения в отличие от крупных и известных методов и методологий, но не является сам по себе методологией. Обычно Agile сравнивают в первую очередь с "методом водопада" ("waterfall"), т.к. на момент выхода манифеста, именно "метод водопада" являлся основным при планировании разработки программного обеспечения. В разработке и выпуске Agile манифеста принимали участие представители следующих методологий: Adaptive software development (ASD)

Crystal Clear

Dynamic Systems Development Method (DSDM)

Extreme Programming (XP)

Feature driven development (FDD)

Pragmatic Programming

Scrum

Собственно, данные методологии гибкой разработки существовали и до выпуска манифеста. Сам же выпуск манифеста дал новый толчок к развитию гибких методологий, заложил основы, можно сказать конституцию гибкого подхода к разработке программного обеспечения.

Основной метрикой agile-методов является рабочий продукт. Отдавая предпочтение непосредственному общению, agile-методы уменьшают объём письменной документации по сравнению с другими методами.

Это привело к критике этих методов как недисциплинированных. Мы постоянно открываем для себя более совершенные методы

разработки программного обеспечения, занимаясь разработкой непосредственно и помогая в этом другим. Благодаря проделанной работе мы смогли осознать, что:

- Люди и взаимодействие важнее процессов и инструментов
- Работающий продукт важнее исчерпывающей документации
- Сотрудничество с заказчиком важнее согласования условий контракта
- Готовность к изменениям важнее следования первоначальному плану.

То есть, не отрицая важности того, что справа, мы всё-таки больше ценим то, что слева.

Основополагающие принципы Agile-манифеста

1) Наивысшим приоритетом для нас является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения.

2) Изменение требований приветствуется, даже на поздних стадиях разработки. Agile-процессы позволяют использовать изменения для обеспечения заказчику конкурентного преимущества.

3) Работающий продукт следует выпускать как можно чаще, с периодичностью от пары недель до пары месяцев.

4) На протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе.

5) Над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте под-держку и полностью доверьтесь им.

6) Непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды.

7) Работающий продукт — основной показатель прогресса.

8) Инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно. Agile помогает наладить такой устойчивый процесс разработки.

9) Постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта.

10) Простота — искусство минимизации лишней работы — крайне необходима.

11) Самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд.

12) Команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

В agile подходах фокус делается на командную работу и взаимодействие, поэтому роли там более общие по своей природе, чем те, что выделяют в традиционных подходах. В agile окружении не описаны задачи, специфичные для бизнес аналитика, и поэтому практикующему бизнес аналитику нужно знать, какие методики бизнес анализа нужно применять в таких проектах. Вот четыре основные роли, присутствующие в agile проектах.

Product Owner основной специалист, принимающий решения по проекту. Эта роль отвечает за видение продукта (product vision), приоритезацию фич, в соответствии с бизнес ценностью, и за ответы на вопросы команды.

Scrum Master отвечает за процесс. Эта роль отвечает за окружение, подходящее для достижения успеха, за устранение препятствий и за получение тесного сотрудничества между всеми ролями.

Team – это группа из 5-9 человек, выделенных для проекта на полное время, они отвечают за самоорганизацию и поставку ценности заказчику в каждой итерации. Команда определяет, как разрабатывать продукт и как распределить работы в условиях выделенного на проект времени.

Stakeholder (заинтересованное лицо) – это любой, кто может повлиять на проект и внести вклад в определение бизнес целей продукта. Заинтересованные лица, активно вовлеченные в проект, это часть команды. Заинтересованные лица, которые не вовлечены активно в проект, могут взаимодействовать с product owner-ом, участвуя таким образом в пополнении баклога (backlog) продукта.

Гибкая модель — это тип инкрементной модели, которая разрабатывается в виде быстрых циклов. Гибкий процесс разбивает задачи на более мелкие итерации или добавочные выпуски, которые не требуют долгосрочного планирования. Каждая итерация обычно длится от одной до четырех недель и проверяется на качество.

Планирование и масштаб проекта заранее включают количество итераций и продолжительность каждой итерации. Каждую итерацию обрабатывает команда профессионалов, которые управляют всем циклом разработки и обеспечивают своевременную реализацию проекта.

Основные этапы гибкой модели включают:

Этап сбора требований. Этот этап включает в себя определение требований для надлежащего планирования проекта заблаговременно.

Осуществимость проекта и технические требования могут быть оценены на основе этих знаний.

Этап проектирования. После определения требований к проекту поработайте с заинтересованными сторонами, чтобы показать им объем проекта и новые функции с помощью блок-схемы высокого уровня.

Этап итерации. Также известный как этап строительства, этап итерации означает начало проекта, который включает в себя различные

этапы разработки. Разработчики начинают работать над развертыванием работающего проекта.

Этап тестирования. Этап тестирования включает в себя проверку качества продукта и его производительности.

Этап развертывания. На этом этапе команда развертывает продукт для рабочей среды пользователя.

Фаза обратной связи. Это последняя фаза процесса гибкой модели, на которой команда получает обратную связь. Руководитель группы гарантирует, что они работают с этой обратной связью.

Четыре идеала применяются по-разному в каждом подходе Agile, но все разработчики программного обеспечения полагаются на них при создании и поставке высококачественного работающего программного обеспечения.

Люди и взаимодействия важнее процессов и инструментов

Легко понять, почему люди ценятся выше, чем процессы или инструменты, потому что именно люди реагируют на потребности компании и управляют процессом разработки. Когда разработка управляется процедурами или инструментами, команда менее приспособлена к изменениям и с меньшей вероятностью оправдает ожидания клиента. Различие между оценкой людей и процессов можно ясно увидеть в общении. Индивидуальное общение является гибким и может происходить всякий раз, когда есть необходимость, тогда как общение должно быть запланировано в случае процессов и требует определенного контента.

Рабочее программное обеспечение важнее полной документации

Исторически сложилось так, что значительное количество времени уходило на документирование продукта при подготовке к разработке и конечной поставке. Для каждого из них требуется собственный набор технических спецификаций, технических требований, технических проспектов, документации по дизайну интерфейса, планов тестирования, планов документации и утверждений. Список был обширным, что часто приводило к длительным задержкам разработки. Agile не устраняет документацию; скорее, это упрощает ее, чтобы у разработчика было все необходимое для выполнения задачи, не увязая в деталях. Agile использует пользовательские истории для документирования требований, и их достаточно, чтобы разработчик программного обеспечения начал работать над новой функцией.

Agile Manifesto уделяет большое внимание документации, но еще большее значение придается работающему программному обеспечению.

Сотрудничество с заказчиком при заключении контракта Переговоры — это когда заказчик и продакт-менеджер прорабатывают детали доставки, с возможностью повторного согласования деталей по маршруту. Сотрудничество — это совсем другая игра. Клиенты обсуждают требования к продукту, часто в мельчайших деталях, до начала какой-либо работы с помощью методологий разработки, таких как Waterfall. Это означало, что заказчик участвовал в процессе разработки до его начала и после его завершения, но не на протяжении большей его части. Манифест Agile описывает клиента, который участвует и взаимодействует с командой разработчиков на протяжении всего процесса. Это значительно облегчает разработчикам выполнение требований заказчика. Хотя

гибкие методологии могут включать клиента через регулярные промежутки времени для периодических демонстраций, проект может так же легко включать конечного пользователя, как и члена команды, который ежедневно посещает все встречи и обеспечивает соответствие продукта бизнес-потребностям клиента.

Реакция на изменение вместо следования плану

Это изменение было расценено как расходы в традиционной разработке программного обеспечения. Таким образом удалось избежать. Цель состояла в том, чтобы создать точные, сложные планы с определенным набором функций и всем, имеющим тот же приоритет, что и все остальное, и огромным количеством различных зависимостей от доставки в определенном порядке, чтобы команда могла перейти к следующему кусочку головоломки.

Поскольку agile-итерации короткие, приоритеты можно корректировать от итерации к итерации, а новые функции можно добавлять на следующей итерации. Изменения, согласно Agile, всегда улучшают проект, т. е. придают новую ценность.

Концепция адаптации метода, возможно, является лучшим примером конструктивного подхода Agile к изменениям. Agile-подходы позволяют Agile-команде изменять процесс так, чтобы он подходил команде, а не наоборот.

Частая поставка рабочего программного обеспечения — Scrum допускает это, поскольку команда работает в спринтах или итерациях программного обеспечения, что обеспечивает постоянную доставку рабочего программного обеспечения.

Сотрудничество между заинтересованными сторонами бизнеса и разработчиками на протяжении всего проекта. Более эффективные решения принимаются, когда бизнес и техническая команда согласованы. Это также уменьшает количество изменений в программном обеспечении.

Поддерживайте, доверяйте и мотивируйте вовлеченных людей. Недовольные команды с меньшей вероятностью сделают свою работу лучше, чем мотивированные.

Обеспечьте взаимодействие лицом к лицу. Эффективное общение может быть достигнуто, когда команды разработчиков находятся рядом друг с другом.

Работающее программное обеспечение является основным мерилом прогресса. Предоставление функционального программного обеспечения заказчику является основным фактором, измеряющим прогресс и успех.

Agile-процессы для поддержки постоянного темпа разработки. Команды устанавливают темп, в котором они могут создавать функционирующее программное обеспечение, которое можно

повторять и поддерживать, и они придерживаются его с каждым выпуском.

Внимание к техническим деталям и дизайну повышает гибкость. Надлежащие таланты и продуманный дизайн гарантируют, что команда сможет идти в ногу со временем, регулярно развивать продукт и адаптироваться к изменениям.

Простота — разрабатывайте ровно столько, чтобы выполнить работу прямо сейчас, чтобы ее можно было выполнить быстро и эффективно.

Самоорганизующиеся команды поощряют отличные архитектуры, требования и проекты. Качественные товары производятся квалифицированными и мотивированными членами команды, которые имеют право принимать решения, берут на себя ответственность, регулярно взаимодействуют с другими членами команды и обмениваются идеями.

Регулярные размышления о том, как стать более эффективными. Члены команды могут работать более эффективно, улучшая свое самосознание, улучшая свои процессы и изучая новые навыки и методы.

Цель Agile — связать разработку с потребностями бизнеса, и она доказала свою эффективность. Клиенты находятся в центре гибких инициатив, которые способствуют вкладу и вовлечению потребителей.

В результате Agile превратился в всеобъемлющее видение разработки программного обеспечения для всей индустрии программного обеспечения и отдельного бизнеса.

Поскольку Agile — это итеративный подход к разработке программного обеспечения, в отличие от модели линейного водопада, гибкие проекты состоят из небольших циклов, называемых спринтами.

Каждый спринт представляет собой мини-проект с бэклогом и состоит из всех этапов Agile, таких как определение, проектирование, разработка, демонстрация и доставка. После завершения цикла спринта небольшой модуль всего программного продукта готов к доставке, и описанный выше процесс повторяется, что приводит к постепенному росту продукта. Следование этому подходу в значительной степени снижает вероятность сбоя продукта, поэтому большинство организаций по всему миру используют agile-подход в своей практике.

Ниже приведены следующие аспекты гибкого процесса:

а.) Гибкость: требования и объем работ меняются в зависимости от потребностей бизнеса.

б.) Разбивка работы: продукт разрабатывается по модулям небольшими циклами (спринтами).

в) Улучшения: изучение прошлых ошибок для улучшения конечного продукта.

d.) Сотрудничество с клиентами: предоставление подробной информации о любых изменениях в требованиях или принятие предложений команды на протяжении спринтов.

3. Задание

Изучить методику управления проекта разработки СИИ **Agile.**

Подготовить ртеферат и презентацию окладв об оснолвных особенностях методологии **Agile.**

4. Контрольные вопросы

1. Что такое гибкая разработка программного обеспечения ?
2. Включает ли в себя разработка проекта как организации планируют, разрабатывают, тестируют и выпускают программное обеспечение?
3. Что включает agile-подход в своей практике реализации проектов СИИ?

Список использованных источников

1. «Agile: практическое руководство» / Олимп–Бизнес: Москва, 2018. – 110 стр. - ISBN 978-5-9693-0403-1, 978-1-62825-418-1.
2. Agile Methodology. Great Learning Tutorials [Электронный ресурс] // Режим доступа: https://www.mygreatlearning.com/blog/ agile-methodology/ (дата обращения: 16.02.2023).
3. Мартин, Р. Чистый Agile. Основы гибкости // Р. Мартин. - СПб.: Питер, 2020. - 352 с. ISBN 978-5-4461-1552-5.

Лабораторная работа №3

Управление проектами СИИ на базе методологии SCRUM

Цель работы: **О**свршение приемов использования методологии SCRUM для азрабатываемой системы искусственного интеллекта и разработка требований к проекту.

1. Теоретические сведения

Scrum — это набор правил, благодаря которым команда налаживает гибкий рабочий процесс, разработка ведется итерациями, четко обозначаются цели каждой итерации и задачи каждого члена команды. Благодаря фреймворку компании могут применять принципы и ценности методологии управления проектами по Agile [1].

Scrum (как, собственно, и Agile) зародился для упрощения рабочих процессов в компаниях, которые занимаются разработкой программного обеспечения и управлением продуктов. В наше время методика Scrum используется в сферах маркетинга, брендинга, дизайна и многих других. Это отличный фреймворк для работы над динамично развивающимися проектами. Scrum направлен на самостоятельную работу над проектом, а не на решение данных «сверху» задач.

Эти два понятия регулярно путают, считая, что Agile и Scrum одно и то же. Обе методологии фокусируются на постоянном совершенствовании продукта, а не на его выпуске. Это гибкие структуры, суть которых в постоянном изменении, адаптивности, направленности на самостоятельную работу участников, нестандартных подходах к работе.

Разница кроется в масштабе двух подходов.

Agile — это особый образ мышления. Идея, стоящая за тем, к чему вы стремитесь — например, к адаптивности, самоконтролю или скорости выполнения заданий.

Scrum — это инструкция по применению. Четкий план, описывающий каждый шаг по внедрению Agile в разработку продукта. Можно сказать, что Scrum — это методология управления проектами с конкретными этапами, в которой четко определены роли и события.

Система управления проектами Scrum основана на пяти ценностях:

- Преданность (Commitment);
- Сфокусированность (Focus);
- Открытость (Openness);
- Уважение (Respect);
- Смелость (Courage).

В контексте Scrum все, что делают работники должно быть направлено на усиление этих ценностей, и ни в коем случае не подрывать их.

И это рабочая методика, так как 58% Agile-команд используют фреймворк Скрам. Благодаря ему члены Scrum-команды могут учитывать нужды клиентов на протяжении всей работы над проектом.

В Scrum нет стандартов идеального долгосрочного планирования, на которую опираются в традиционных рабочих подходах. Фреймворк сосредотачивается на выполнении задач на короткой дистанции.

Состав Scrum команды.

Прежде чем говорить о структуре фреймворка, рассмотрим, кто обычно входит в состав Scrum-команды.

Владелец продукта — тот, кто налаживает связь между командой и заинтересованными лицами. Он понимает, что нужно клиентам, контролирует общее видение проекта и его цели.

Scrum-мастер — один из членов команды, в задачи которого входит внедрение и укрепление ценностей Scrum на командных митингах и поддержка участников во время выполнения задач.

Члены команды — остальные участники Scrum-команды. Все они равноправны и каждый выполняют свою задачу.

Заинтересованные лица, упомянутые выше — не члены команды. Это все те, кто инвестирует в результат проекта. Например, особые клиенты, внутренние пользователи продукта, руководители высшего звена и прочие. Ключевые заинтересованные лица присутствуют на важных встречах и рассматривают ключевые решения по модернизации продукта, а также предоставляют обратную связь после каждой итерации.

Scrum-команда обязательно кросс-функциональна. Например, в команде по созданию мобильных приложений должны быть UX-дизайнеры, разработчики, специалисты по API и прочие. Каждый участник обязан располагать соответствующими инструментами для завершения итерации. Поэтому у них не должно возникать необходимости передавать часть работы на аутсорс. Это один из основных принципов управления проектами по Scrum.

Этапы Scrum.

В фреймворке Scrum можно выделить пять основных этапов:

1. Предварительное планирование.

Постановка целей, определение видения продукта. Лидер проекта обозначает задачи, намечает дорожную карту проекта. Создание и доработка бэклога продукта — списка функций, требований и исправлений ошибок, где для команды прописываются все этапы работы над продуктом. Обычно к этапу предварительного планирования объема работы присоединяются заинтересованные лица.

2. Планирование.

На этом этапе участники команды вместе занимаются планированием спринта и выбором функций для включения в его бэклог. Поскольку их обычно определяет точка зрения пользователя, они называются пользовательскими историями. Необходимо разбить большие требования (которые обычно называют «эпиками») на простые задачи с приблизительной оценкой времени выполнения. Стоит убедиться, что бэклог спринта достаточно небольшой, его получится выполнить в рамках планируемого времени, распределить задачи и назначить ответственных за пользовательские истории.

3. Спринт, этап реализации.

Работа идет над итерацией или инкрементом продукта (ощутимый результат работы одного спринта), который реализуется в конце спринта. Необходимо проводить ежедневные митинги или Scrum-собрания, на которых будет обсуждаться прогресс, задачи, потенциальные трудности.

4. Тестирование и проверка.

По окончанию спринта клиенты и пользователи продукта (заинтересованные лица) тестируют новые функции или улучшения продукта. Если все работает как надо, итерация считается завершенной.

5. Ретроспектива.

Анализ итогов спринта вместе со Scrum-командой, во время которого разбираются ошибки и выдвигаются предложения по улучшению работы. Общий бэклог продукта актуализируется в зависимости от результатов работы над обновлениями и смены приоритетов у заинтересованных лиц.

Ценность фреймворка Scrum для управления проектами

Scrum популярен за счет ряда преимуществ для команд, которые решили использовать его для организации работы [2]:

- Наглядность процесса. Намеченные задачи к выполнению можно представить в удобном виде на доске в таск-трекере. Визуализируя задачи на Канбан-доске, Scrum-команда всегда видит, как продвигается работа, к кому обращаться по тем или иным вопросам и какие задачи в работе сегодня.

- Концентрация на важном. Предварительное планирование целей спринта помогает не распыляться на другие задачи, оставаться сосредоточенным и собранным.

- Конкретные результаты. Итогом итерации по Scrum всегда является какое-то улучшение, определенное достижение. Его можно оценить и однозначно ответить, достигла ли команда поставленных целей в полной мере или нет. Для быстрой и удобной оценки используются разнообразные Agile-метрики: упомянутый выше график

сгорания задач (Burndown chart), накопительная диаграмма потока (Cumulative Flow Diagram) и другие.

- Все участники процесса поддерживают связь. Все участники команды, Scrum-мастер, владелец продукта, заказчик и заинтересованные лица всегда поддерживают коммуникацию. Любые уточнения всегда можно получить быстро, чтобы не только в короткие сроки выпускать продукт, но и поддерживать его актуальность, вовремя реагируя на изменения рынка и нужды клиентов.

- Высокая гибкость и адаптивность. Несмотря на точное планирование целей, методика Scrum все же не подразумевает обязательного следования одним и тем же правилам. Вы можете адаптировать подход к потребностям именно вашей команды и для достижения ваших целей, организовав работу максимально удобно.

Отличия Scrum как подхода для организации работы

Среди множества различных подходов и методологий организации работы, Scrum выделяется следующими особенностями:

- Четко зафиксированные роли сотрудников, цели и этапы спринтов. Во время итерации Scrum-команда всегда знает, кто, над чем и для чего работает в любой момент времени.

- Кросс-функциональность команды. Команда включает в себя разных специалистов, которые работают в связке. Например, для создания видеоигры необходима команда из разработчиков, графических дизайнеров, тестировщиков, сценаристов и др. Полностью укомплектованная команда для проекта самодостаточна и не требует сторонних экспертов для выполнения задач.

- Отсутствие долгосрочного планирования. Scrum не подходит для построения долгосрочных планов. При данном подходе приоритеты и цели постоянно меняются между итерациями, гибко адаптируясь к текущим требованиям к продукту. Краткосрочные спринты помогают Scrum-команде единым рывком выполнять поставленные цели, при этом держится фокус на обозначенных задачах без траты ресурсов на остальные дела.

- Предварительное планирование задач для краткосрочных спринтов. Scrum требует обязательного составления подробного бэклога и выделения целей на каждый цикл.

- Есть только одно лицо для коммуникации между командой и заинтересованными лицами — владелец продукта. Так устраняется риск противоречивости полученной информации по задачам, все запросы и ответы исходят от одного человека.

- Регулярное общение с командой. Участники проекта говорят о прогрессе и проблемах в работе на ежедневных собраниях, встречах по пополнению очереди задач и других видах собраний для обмена информацией и получением обратной связи. Обсуждение текущих сложностей и способов их решения — важная часть работы по Scrum.

- Обязательная оценка результата и получение обратной связи от заинтересованных лиц после окончания спринта через владельца продукта. Без получения одобрения от заинтересованных лиц результата работы цель не может считаться достигнутой и следующий спринт не может быть начат.

5. Задание

Изучить методику управления проекта разработки СИИ **Scrum.**

Подготовить ртеферат и презентацию окладв об оснолвных особенностях методологии **Scrum.**

6. Контрольные вопросы

4. Что такое гибкая разработка программного обеспечения ?
5. Включает ли в себя разработка проекта как организации планируют, разрабатывают, тестируют и выпускают программное обеспечение?
6. Что включает Scrum- подход в своей практике реализации проектов СИИ?

Список использованных источников

1. Сазерленд Д. Scrum. Революционный метод управления проектами [Текст] / Сазерленд Д. — 1. — Москва: Манн, Иванов и Фербер, 2022 — 272 с.

2. Кон М. Scrum: гибкая разработка ПО [Текст] / Кон М. — 1. — Москва: Вильямс, 2017 — 576 с.

Лабораторная работа №4

Системы управления проектами СИИ нв основе YOUTRACK

Цель работы: **О**свршение приемов использования методологии для азрабатываемой системы искусственного интеллекта и разработка требований к проекту.

2. Теоретические сведения

YouTrack — это инструмент управления проектами, который адаптируется под потребности различных команд в компании. В YouTrack можно планировать проекты и отслеживать задачи, использовать Agile-доски, организовывать спринты и релизы, вести базу знаний, использовать диаграмму Ганта, отслеживать время выполнения работы, создавать отчёты и панели мониторинга, настраивать рабочие процессы. YouTrack полностью подстраивается под бизнес-процессы различных команд — от небольших стартапов до корпораций. 70 тысяч команд во всём мире используют YouTrack.

Сервис YouTrack был запущен компанией JetBrains в 2009 году.

YouTrack был разработан в соответствии с парадигмой языково-ориентированного программирования, использует JavaScript и Kotlin.

Система использует встроенную базу данных Xodus для записи и хранения данных. Для удалённых вызовов процедур использует REST-стиль.

YouTrack используется не только для отслеживания ошибок, но и для управления проектами, благодаря широкому набору функций, включая Agile доски, диаграммы Gantt и слежение за временем.

YouTrack поддерживает интеграцию с большим количеством других инструментов, таких как Jira, Redmine, GitHub, GitLab, TFS и другие.

Система полнотекстового поиска YouTrack позволяет пользователям быстро находить задачи, используя при этом естественный язык.

YouTrack предоставляет возможность конфигурировать рабочее пространство пользователя, чтобы каждый член команды мог настроить свой интерфейс по своему усмотрению.

В YouTrack есть возможность создания автоматических команд, которые позволяют пользователям автоматизировать рутинные действия с задачами.

Существует функция «Тайм-трекинг» в YouTrack, которая помогает отслеживать и контролировать затраченное время на определенные задачи.

YouTrack предлагает гибкую систему лицензирования, включая бесплатный вариант для небольших команд до 10 пользователей.

7. Задание

Изучить сервис управления проекта разработки СИИ **YouTrack.**

Подготовить ртеферат и презентацию окладв об оснолвных особенностях методологии **YouTrack.**

8. Контрольные вопросы

7. Что такое гибкая разработка программного обеспечения ?
8. Включает ли в себя разработка проекта как организации планируют, разрабатывают, тестируют и выпускают программное обеспечение?
9. Что **включает YouTrack** - подход в своей практике реализации проектов СИИ?

Список литературы

1. Аньшин, В. М. Управление проектами: фундаментальный курс [Электронный ресурс] : учебник / В. М. Аньшин, А. Алешин, К. Багратиони. - Москва : Высшая школа экономики, 2013. - 624 с. – Режим доступа : biblioclub.ru.
2. Ипатова, Э. Р. Методологии и технологии системного проектирования информационных систем : учебник / Э. Р. Ипатова, Ю. В. Ипатов. – 3-е изд., стер. – Москва : ФЛИНТА, 2021. – 256 с. : табл., схем. – (Информационные технологии). – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=79551 (дата обращения: 22.01.2024). – Библиогр.: с. 95-96. – ISBN 978-5-89349-978-0. – Текст : электронный.
3. Шуваев, А. В. Программная инженерия : учебное пособие для магистрантов направления подготовки 09.04.02 – Информационные системы и технологии : [16+] / А. В. Шуваев ; Ставропольский государственный аграрный университет, Кафедра информационных

систем. – Ставрополь : Ветеран, 2020. – 84 с. : ил., табл. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=700960 (дата обращения: 22.01.2024). – Библиогр. в кн. – Текст : электронный.

4. Кугаевских, А. В. Проектирование информационных систем. Системная и бизнес-аналитика : учебное пособие : [16+] / А. В. Кугаевских ; Новосибирский государственный технический университет. – Новосибирск : Новосибирский государственный технический университет, 2018. – 256 с. : табл., схем., ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=573827 (дата обращения: 22.01.2024). – Библиогр.: с. 247-251. – ISBN 978-5-7782-3608-0. – Текст : электронный.

5. Антонов, В. Ф. Методы и средства проектирования информационных систем : учебное пособие / В. Ф. Антонов, А. А. Москвитин ; Северо-Кавказский федеральный университет. – Ставрополь : Северо-Кавказский Федеральный университет (СКФУ), 2016. – 342 с. - URL: http://biblioclub.ru/index.php?page=book&id=458663. - Режим доступа: по подписке. - Текст : электронный.