



Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ

«Робототехника и комплексная автоматизация»

КАФЕДРА

«Системы автоматизированного проектирования (РК-6)»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
НА ТЕМУ:
«Автоматизация обработки отчётов об ошибках в
программном обеспечении с помощью больших языковых
моделей»

Студент РК6-42М
(Группа)

(Подпись, дата)

Гунько Н.М.
(Фамилия И.О.)

Руководитель ВКР

(Подпись, дата)

Витюков Ф.А.
(Фамилия И.О.)

Нормоконтролёр

(Подпись, дата)

Грошев С.В.
(Фамилия И.О.)

2025 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой РК6
(Индекс)
Карпенко А.П.
(Подпись) (Фамилия И.О.)
«15» февраля 2025 г.
(Дата)

ЗАДАНИЕ
на выполнение выпускной квалификационной работы

Студент группы: РК6-42М

Гунько Никита Макарович
(фамилия, имя, отчество)

Тема выпускной квалификационной работы «Автоматизация обработки отчётов об ошибках в программном обеспечении с помощью больших языковых моделей»

При выполнении ВКР:

	Используются / Не используются	Да / Нет
1)	Литературные источники и документы, имеющие гриф секретности	Нет
2)	Литературные источники и документы, имеющие пометку «Для служебного пользования», иных пометок, запрещающих открытое опубликование	Нет
3)	Служебные материалы других организаций	Нет
4)	Результаты НИР (ОКР), выполняемой в МГТУ им. Н.Э. Баумана	Нет
5)	Материалы по незавершённым исследованиям или материалы по завершённым исследованиям, но ещё не опубликованные в открытой печати	Нет

Тема квалификационной работы утверждена распоряжением по факультету:		
Название факультета:		
Дата	и	рег. номер
распоряжения:		

Техническое задание

Часть 1. Аналитическая часть

Исследовать современные подходы к автоматизации анализа обращений в сфере технической поддержки программного обеспечения. Проанализировать архитектурные особенности крупных языковых моделей, включая GPT-4, Claude, DeepSeek и Grok. Изучить методы их применения для интерпретации баг-репортов и формирования рекомендаций. Выявить

ключевые метрики оценки интеллектуальных возможностей LLM и сопоставить их применимость к задачам технической диагностики.

Часть 2. Практическая часть 1

Разработать архитектуру программного решения, предназначенного для автоматизированного анализа обращений пользователей с помощью LLM. Спроектировать модуль взаимодействия с языковой моделью. Определить сценарии интеграции с системой Intradesc.

Часть 3. Практическая часть 2

Реализовать прототип программного средства, обеспечивающего автоматизированную обработку баг-репортов с использованием выбранной языковой модели. Настроить обмен данными с LLM через API. Выполнить тестирование системы на наборе типовых обращений и оценить качество генерации ответов.

Оформление квалификационной работы:

Расчетно-пояснительная записка на ____ листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

Работа содержит 8 графических листов формата А1:

1. Структура баг-репорта;
2. Схема взаимодействия отделов;
3. Схема взаимодействия отделов после внедрения;
4. Архитектура программного решения;
5. Пример баг-репорта в формате JSON;
6. Пример запроса и ответа языковой модели в формате JSON;
7. Промпт к языковой модели;
8. Сравнение стоимости при разной нагрузке.

Дата выдачи задания: «10» февраля 2025 г.

В соответствии с учебным планом выпускную квалификационную работу выполнить в полном объеме в срок до «1» июня 2025 г.

Руководитель квалификационной работы

(Подпись, дата)

Витюков Ф.А.

(Фамилия И.О.)

Студент

(Подпись, дата)

Гунько Н.М.

(Фамилия И.О.)

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ РК
КАФЕДРА РК6
ГРУППА РК6-42М

УТВЕРЖДАЮ
Заведующий кафедрой РК6
(Индекс)
Карпенко А.П.
(Подпись) (Фамилия И.О.)
«15» февраля 2025 г.
(Дата)

КАЛЕНДАРНЫЙ ПЛАН
выполнения выпускной квалификационной работы

Студента Гуныко Никиты Макаровича
(фамилия, имя, отчество)

Тема квалификационной работы «Автоматизация обработки отчётов об ошибках в программном обеспечении с помощью больших языковых моделей»

№ п/п	Наименование этапов выпускной квалификационной работы	Сроки выполнения этапов		Отметка о выполнении	
		план	факт	Должность	ФИО, подпись
1.	Задание на выполнение работы. Формулирование проблемы, цели и задач работы	10.02.2025	_____	Руководитель ВКР	Витюков Ф.А.
2.	1 часть. Аналитическая часть	18.02.2025	_____	Руководитель ВКР	Витюков Ф.А.
3.	Утверждение окончательных формулировок решаемой проблемы, цели работы и перечня задач	28.02.2025	_____	Заведующий кафедрой	Карпенко А.П.
4.	2 часть. Практическая часть 1	15.04.2025	_____	Руководитель ВКР	Витюков Ф.А.
5.	3 часть. Практическая часть 2	05.05.2025	_____	Руководитель ВКР	Витюков Ф.А.
6.	1-я редакция работы	12.05.2025	_____	Руководитель ВКР	Витюков Ф.А.
7.	Подготовка доклада и презентации	23.05.2025	_____	Студент	Гуныко Н.М.
8.	Заключение руководителя	01.06.2025	_____	Руководитель ВКР	Витюков Ф.А.
9.	Допуск работы к защите на ГЭК (нормоконтроль)	04.06.2025	_____	Нормоконтролер	Грошев С.В.
10.	Внешняя рецензия	05.06.2025	_____		
11.	Защита работы на ГЭК	09.06.2025	_____		

Студент Гуныко Н.М. (подпись, дата) (Фамилия И.О.)
Руководитель ВКР Витюков Ф.А. (подпись, дата) (Фамилия И.О.)

АННОТАЦИЯ

Работа посвящена исследованию возможностей автоматизации обработки отчётов об ошибках с применением современных крупных языковых моделей, а также разработке программного решения, интегрированного с корпоративной системой управления обращениями Intradesc. В рамках исследования проведён обзор и сравнительный анализ моделей GPT-4, Claude, DeepSeek и Grok с точки зрения архитектуры, вычислительных требований, уровня интеллектуальных способностей, а также их практической пригодности к задачам смысловой обработки пользовательских обращений. Разработанное программное решение осуществляет интеграцию с внешними языковыми моделями через API и обеспечивает обработку обращений в рамках внутренней среды компании. Также исследована зависимость производительности моделей от вычислительных ресурсов и проанализированы факторы, влияющие на выбор решений при интеграции в инфраструктуру предприятия.

Тип работы: выпускная квалификационная работа.

Тема работы: «Автоматизация работы с отчётами об ошибках с применением крупных языковых моделей».

Объекты исследований: крупные языковые модели, автоматизация обработки баг-репортов, интеграция LLM в корпоративные системы, взаимодействие с API языковых моделей, производительность моделей в условиях ограниченных ресурсов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	10
1. Теоретические и прикладные аспекты баг-репортов	13
1.1. Основные принципы и структура	13
1.2. Проблемы традиционного подхода к анализу баг-репортов.....	15
1.3. Языковые модели для автоматизации обработки	16
1.4. Процессы сопровождения программных решений.....	17
1.5. Типы обращений и прикладные сценарии	18
2. Обзор и сравнение крупных языковых моделей	20
2.1 Архитектурные особенности GPT-4, Claude, DeepSeek и Grok	20
2.2 Сравнение моделей по интеллектуальным метрикам.....	23
2.2.1. Общие интеллектуальные способности и логика	23
2.2.2. Задачи программирования.....	24
2.2.3. Творческие задачи	25
2.3. Анализ вычислительных требований	27
2.4. Оценка применимости к задачам обработки баг-репортов.....	31
2.4.1. Понимание технического языка и логики	31
2.4.2. Точность и глубина анализа	32
2.4.3. Учет контекста и длинных диалогов	33
2.4.4. Эмоциональный интеллект и тон ответов	34
2.4.5. Конфиденциальность и интеграция.....	35
3. Разработка программного решения.....	37
3.1. Архитектура системы обработки баг-репортов	38
3.2. Интеграция с системой Intradesc.....	41
3.3. Очистка обращений от конфиденциальных данных	45
3.4. Взаимодействие с внешними языковыми моделями	46
3.5. Пример обращения и результат анализа	51
3.6. Организация среды развертывания	56
3.7. Оценка стоимости использования языковых моделей	58
ЗАКЛЮЧЕНИЕ	62
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	64
ПРИЛОЖЕНИЕ А	69

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

Bug-report (bug report) – сообщение об ошибке в работе программного обеспечения, составленное пользователем или тестировщиком.

API (Application Programming Interface) – программный интерфейс, предоставляющий доступ к функциям внешней системы или сервиса.\

Искусственный интеллект (ИИ; Artificial Intelligence, AI) – направление в информатике и информационных технологиях, задачей которого является воссоздание разумных рассуждений и действий с помощью вычислительных систем и иных искусственных устройств.

LLM (Large Language Model) – крупная языковая модель, обученная на больших массивах текстовых данных и способная выполнять задачи генерации, интерпретации и трансформации текста на естественном языке.

GPT-4, Claude, DeepSeek, Grok – современные крупные языковые модели, использующие различные архитектурные подходы и методы генерации текста.

CRM (Customer Relationship Management) – система управления взаимоотношениями с клиентами, предназначенная для автоматизации процессов продаж, поддержки, маркетинга и анализа взаимодействий с клиентами.

Intradesc – корпоративная система учёта заявок и управления обращениями с веб-интерфейсом, предназначенная для организации службы поддержки, постановки задач, обработки клиентских обращений, а также автоматизации сервисных и CRM-процессов.

Автоматизирующая программа-прослойка (АПП) – программное средство, предназначенное для автоматизации предварительной обработки обращений, анализа их полноты и маршрутизации внутри организации.

Мультимодальность – способность модели обрабатывать различные типы входных данных (например, текст, изображение) одновременно.

Инференс (inference) – этап использования обученной модели для получения результата по входным данным (например, генерации ответа по баг-репорту).

Маршрутизация обращений – процесс автоматического определения ответственного исполнителя или отдела на основе содержимого обращения.

Метрики интеллекта LLM – стандартизированные количественные показатели, используемые для оценки когнитивных возможностей языковой модели (например, MMLU, BIG-Bench, HumanEval и др.).

Корпоративная инфраструктура – совокупность аппаратных, программных и организационных ресурсов внутри предприятия, обеспечивающих выполнение ИТ-процессов.

NLP (Natural Language Processing) – это направление в машинном обучении, посвященное распознаванию, генерации и обработке устной и письменной человеческой речи.

ML (англ. Machine Learning – Машинное обучение) – это подраздел ИИ о разработке алгоритмов и моделей, способных решать задачи через обобщение множества схожих примеров. В рамках машинного обучения система собирает информацию, учится на ней, а затем использует то, чему научилась, для принятия решений.

Hugging Face – это платформа с коллекцией готовых современных предварительно обученных Deep Learning моделей. А библиотека Transformers предоставляет инструменты и интерфейсы для их простой загрузки и использования. Это позволяет вам экономить время и ресурсы, необходимые для обучения моделей с нуля.

Предобработка текста – процесс очистки и подготовки текстовых данных перед их дальнейшим анализом и использованием языковыми моделями.

Очистка конфиденциальных данных – автоматическое удаление или замена персональной и чувствительной информации (например, ИНН, номера счетов) в текстовых сообщениях перед их дальнейшей обработкой.

Модуль анализа полноты – компонент системы, предназначенный для оценки наличия всех необходимых элементов в баг-репорте и генерации уточняющих вопросов при необходимости.

Сервис-деск (Service Desk) – программный комплекс для управления инцидентами, запросами пользователей и внутренними задачами в организации, поддерживающий ITIL-процессы.

Оценка стоимости использования моделей – анализ финансовых затрат, связанных с эксплуатацией языковых моделей в зависимости от объемов обработки обращений и используемых вычислительных ресурсов.

Конституциональный ИИ (Constitutional AI) – подход к обучению языковых моделей, основанный на соблюдении заранее определенных этических принципов.

ВВЕДЕНИЕ

С ростом масштабов и сложности программных решений возрастает и объём технической информации, сопровождающей жизненный цикл программного обеспечения. Одной из ключевых составляющих процессов сопровождения и эксплуатации является работа с баг-репортами – пользовательскими или системными сообщениями об ошибках в работе программных продуктов. Обработка этих сообщений представляет собой трудоёмкую и многосоставную задачу: необходимо понять суть обращения, выделить ключевые данные, проверить наличие дубликатов, оценить корректность и полноту описания, а также определить дальнейшие действия – маршрутизация задачи на исполнителя или запрос дополнительной информации.

В условиях увеличения количества обращений и ограниченных ресурсов команд поддержки возрастает потребность в интеллектуальной автоматизации анализа и обработки баг-репортов. Использование жёстко заданных правил и шаблонов эффективно лишь при высокой стандартизации входных данных, однако на практике подавляющее большинство сообщений об ошибках составляются в свободной форме, с неструктурированной и разной по качеству информацией.

Среди типичных проблем, возникающих при ручной обработке баг-репортов, можно выделить:

- необходимость анализа неструктурированных текстов с определенным количеством фоновой или противоречивой информации;
- отсутствие стандартизированного представления информации о проблеме, включая описание произошедшего, ожидаемого поведения системы и предполагаемой причины сбоя

Решение этих задач требует применения интеллектуальных технологий обработки естественного языка. С развитием крупных языковых моделей – таких как GPT-4, Claude, DeepSeek и Grok – появилась возможность задействовать

современные инструменты машинного обучения для автоматизации анализа обращений, извлечения смысловой структуры, распознавания повторяющихся проблем и формирования обоснованных технических заключений.

Настоящая работа посвящена исследованию возможностей применения таких моделей для автоматизации обработки баг-репортов, а также разработке программного решения, интегрированного с системой управления обращениями Intradesc, используемой в корпоративной среде для регистрации, маршрутизации и анализа заявок. Решение должно обеспечивать автоматическую интерпретацию обращений с участием крупной языковой модели, а также соответствовать требованиям безопасности, производительности и совместимости с инфраструктурой предприятия.

Цель работы – исследовать потенциал использования современных крупных языковых моделей для автоматизации обработки отчётов об ошибках, провести сравнительный анализ моделей GPT-4, Claude, DeepSeek и Grok с точки зрения их архитектурных особенностей, вычислительных требований и интеллектуальных способностей, а также разработать программное решение, интегрированное с корпоративной системой Intradesc и способное обеспечивать эффективную обработку обращений в условиях внутренней среды компании.

Для достижения поставленной цели выделены следующие задачи:

- Изучить современные подходы к автоматизации обработки баг-репортов, включая методы работы с неструктурированными пользовательскими текстами.
- Проанализировать архитектурные особенности и функциональность языковых моделей GPT-4, Claude, DeepSeek и Grok.
- Провести сравнительную оценку указанных моделей по интеллектуальным метрикам, вычислительным требованиям и прикладной пригодности к задачам анализа обращений.
- Исследовать влияние вычислительных ограничений на производительность моделей в различных режимах использования.

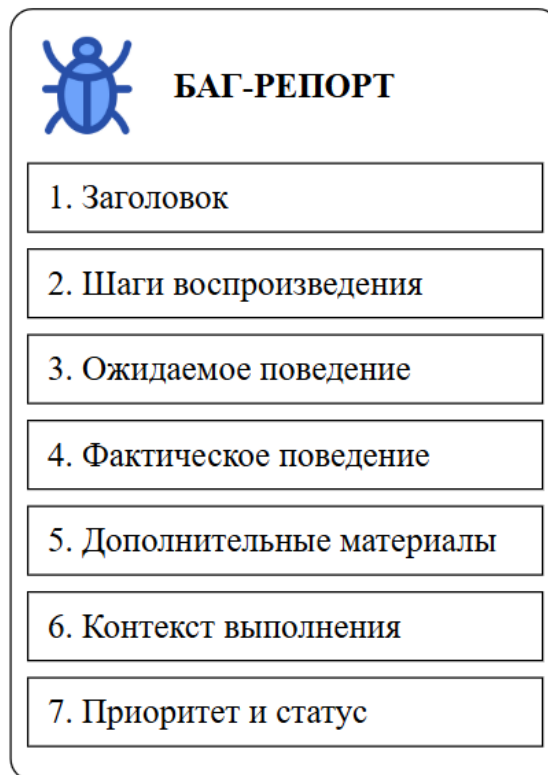
- Разработать программное решение, обеспечивающее автоматизированный сбор и обработку заявок из системы Intradesc с использованием языковых моделей через API.

1. Теоретические и прикладные аспекты баг-репортов

1.1. Основные принципы и структура

Баг-репорт представляет собой структурированную форму данных, используемую для фиксации и документирования информации об обнаруженных дефектах или ошибках в программном обеспечении (ПО). Грамотно составленный баг-репорт является важным инструментом в процессах тестирования, сопровождения и исправления программных продуктов, так как существенно повышает эффективность выявления, анализа и устранения ошибок.

На рисунке 1 представлена стандартная структура баг-репорта.




 БАГ-РЕПОРТ	
1.	Заголовок
2.	Шаги воспроизведения
3.	Ожидаемое поведение
4.	Фактическое поведение
5.	Дополнительные материалы
6.	Контекст выполнения
7.	Приоритет и статус

Рисунок 1 – Основные части стандартной структуры баг-репорта

Наличие стандартизированных элементов в структуре баг-репорта обеспечивает полноту информации и минимизирует риск упущения важных

деталей, непосредственно влияющих на скорость и точность исправления дефектов [1].

Описание основных элементов, входящих в состав баг-репорта:

1. **Заголовок.** Должен лаконично и точно отображать сущность проблемы. Заголовок помогает быстро понять характер дефекта без необходимости подробного изучения описания, что ускоряет процесс обработки и анализа.

2. **Шаги воспроизведения.** Являются обязательным элементом и должны быть описаны чётко, однозначно и последовательно. Подробность шагов воспроизведения позволяет любому специалисту повторить ошибку в идентичных условиях, что критически важно для последующего устранения и тестирования исправлений.

3. **Ожидаемое поведение.** Представляет собой описание корректного и предусмотренного спецификациями поведения программного продукта. Этот элемент необходим для точного понимания требований и выявления отклонений при тестировании.

4. **Фактическое поведение.** Включает подробное и объективное описание фактического состояния или реакции системы, полученного после выполнения указанных шагов. Чёткое документирование фактического поведения облегчает диагностику и локализацию источника проблемы.

5. **Дополнительные материалы.** Включают в себя различные типы информационных артефактов, такие как логи, скриншоты, видеозаписи и иные материалы, позволяющие детализировать и визуализировать проблему. Данные материалы значительно ускоряют выявление точной причины и облегчение её устранения.

6. **Контекст выполнения.** Содержит технические детали среды, в которой была обнаружена ошибка: версия программного обеспечения, тип операционной системы, используемый браузер, аппаратная платформа и другие релевантные параметры окружения. Это особенно важно для ошибок, проявляющихся только в специфических конфигурациях.

7. Приоритет и статус. Категоризация по степени критичности проблемы (блокирующая, высокая, средняя, низкая) и текущее состояние обработки бага (новый, в работе, исправлен, закрыт). Чётко установленный приоритет и статус помогают команде правильно распределять ресурсы и организовать рабочий процесс.

Отсутствие или неполное заполнение любого из элементов баг-репорта приводит к снижению эффективности его анализа, увеличению сроков исправления дефектов и потенциальному увеличению стоимости разработки и поддержки программного обеспечения [2].

1.2. Проблемы традиционного подхода к анализу баг-репортов

Традиционный подход к обработке баг-репортов, преимущественно основывается на ручном анализе, что порождает ряд существенных проблем. Одной из наиболее значимых является влияние человеческого фактора. Даже опытные специалисты могут допускать ошибки или упускать важные детали из-за усталости, невнимательности или ограниченных временных ресурсов.

Ещё одной существенной проблемой традиционного подхода является сложность масштабирования процессов анализа обращений. По мере роста сложности программных продуктов и увеличения количества пользователей объём обращений стремительно возрастает, достигая тысяч и десятков тысяч баг-репортов. Ручной подход в таких условиях становится неэффективным и экономически неоправданным, так как требует значительных трудозатрат и замедляет реагирование на критические ошибки.

Также серьёзным недостатком ручного анализа является проблема идентификации и группировки схожих обращений, особенно когда разные пользователи или тестировщики описывают одну и ту же проблему разными словами и терминами. Это усложняет выявление дублирующих отчётов, затрудняя эффективное управление ошибками и значительно увеличивая временные затраты на диагностику и устранение дефектов.

Дополнительным ограничением традиционного подхода является недостаточная интеграция с современными технологиями анализа данных. Многие распространённые системы баг-трекинга не поддерживают автоматизированную классификацию, обработку мультимодальной информации (например, текст и изображения), а также не применяют инструменты машинного обучения, способные существенно повысить скорость и точность обработки обращений.

Таким образом, указанные проблемы традиционного подхода приводят к снижению общей эффективности процесса обработки баг-репортов, увеличению трудозатрат и временных издержек, а также ухудшению качества программных продуктов и уровня удовлетворённости пользователей [3].

1.3. Языковые модели для автоматизации обработки

В последние годы большие языковые модели (БЯМ) продемонстрировали значительный потенциал в автоматизации процессов анализа и обработки текстовых данных. Благодаря способности к глубокому пониманию контекста и генерации связного текста, БЯМ способны не только классифицировать и структурировать информацию, но и активно участвовать в улучшении качества баг-репортов.

Ключевые направления применения БЯМ в обработке баг-репортов:

- **Анализ полноты и структурированности баг-репорта.** БЯМ могут автоматически проверять наличие всех необходимых компонентов в баг-репорте, таких как шаги воспроизведения, ожидаемое и фактическое поведение, а также контекст выполнения. При обнаружении отсутствующих или неполных разделов модель способна сформулировать запросы к автору отчета для предоставления недостающей информации, тем самым повышая качество и полноту данных.
- **Выявление дубликатов и схожих багов.** Используя семантический анализ, БЯМ способны обнаруживать баг-репорты, описывающие одну и ту же

проблему разными словами. Это позволяет сократить количество повторяющихся записей и оптимизировать процесс устранения дефектов.

- **Генерация рекомендаций по устранению ошибок.** На основе анализа текста баг-репорта и сопоставления с документацией или предыдущими случаями, БЯМ могут предлагать возможные причины возникновения ошибки и рекомендации по ее устранению.

- **Обработка мультимодальных данных.** Современные БЯМ способны анализировать не только текстовую информацию, но и визуальные данные, такие как скриншоты или видеозаписи. Это расширяет возможности диагностики, позволяя выявлять проблемы, которые сложно описать словами, например, визуальные артефакты или некорректное отображение интерфейса.

Применение БЯМ в обработке баг-репортов позволяет значительно повысить эффективность процессов тестирования и сопровождения программного обеспечения, снижая нагрузку на специалистов и ускоряя выявление и устранение дефектов [4].

1.4. Процессы сопровождения программных решений

В рамках производственной деятельности компании ООО «ЮБС», разрабатывающей программные решения для банковского сектора, сопровождение программного обеспечения включает в себя широкий спектр задач, связанных с реакцией на обращения клиентов, анализом ошибок, консультированием и взаимодействием между подразделениями.

Типовой цикл сопровождения начинается с поступления обращения от клиента. Оно может быть связано с технической ошибкой, некорректным поведением функциональности, либо представлять собой запрос на разъяснение или настройку. Обращения фиксируются в системе Service Desk (в данном случае – Intradesk) и требуют дополнительного уточнения информации. Ключевая задача сопровождающего специалиста – **добыть из клиента необходимые сведения**, включая:

- что пользователь делал;
- какое поведение ожидалось;
- какое поведение наблюдается;
- каковы условия среды.

Эти сведения критичны для постановки корректной задачи в систему разработки и последующего устранения неисправности.

Кроме технических аспектов, сопровождение также включает координацию между командами разработки, тестирования и поддержки. Отправной точкой всегда является корректно составленное и верифицированное обращение, преобразованное в техническое задание с использованием внутренних шаблонов.

Таким образом, эффективное сопровождение – это не только техническая экспертиза, но и коммуникационная работа по уточнению контекста проблемы. Именно здесь автоматизация анализа баг-репортов на основе языковых моделей может значительно ускорить процесс уточнения информации, автоматизируя интерпретацию и структурирование неформализованных обращений.

1.5. Типы обращений и прикладные сценарии

Внутри производственной среды обращения, поступающие в систему Service Desk, условно делятся на три ключевые категории:

- **Ошибки (инциденты)** – это обращения, содержащие информацию о некорректной работе ПО. Их дальнейшая обработка включает уточнение всех параметров, постановку задачи в разработку и контроль исправления.
- **Консультации** – запросы, не содержащие признаков ошибки, но требующие пояснений, настройки, анализа поведения продукта в конкретных условиях. Эти обращения, как правило, закрываются силами специалистов линии сопровождения.
- **Запросы на улучшение (инициативы)** – обращения, в которых клиент выражает потребность в доработке, новой функциональности или изменении

поведения системы. Они подлежат приоритизации и передаче в продуктовую команду.

Эти обращения, независимо от типа, приводят к постановке задач: на разработку, на тестирование, либо на консультационное сопровождение. Вся информация по ним аккумулируется в едином хранилище, и дальнейшая автоматизация с применением языковых моделей помогает:

- автоматизировать распознавание типа обращения;
- извлекать ключевые параметры для задач;
- проверять полноту сообщения;
- связывать с ранее решёнными случаями.

Таким образом, введение классификации и формализация обработки обращений являются важной предпосылкой для внедрения систем автоматизации на основе ИИ.

2. Обзор и сравнение крупных языковых моделей

2.1 Архитектурные особенности GPT-4, Claude, DeepSeek и Grok

Модель **GPT-4** от компании OpenAI представляет собой большую трансформерную нейронную сеть, обученную предсказывать следующий токен в тексте. Точные архитектурные параметры GPT-4 не раскрыты OpenAI из соображений безопасности и конкуренции, однако известно, что модель является мультимодальной – она способна обрабатывать не только текст, но и изображения в качестве входных данных.

По оценкам, GPT-4 содержит сотни миллиардов параметров (возможно, более триллиона), что делает ее одной из крупнейших моделей своего времени. Для повышения качества ответов GPT-4 подвергалась пост-обучению с подкреплением от обратной связи людей (RLHF), что улучшило ее фактическую точность и следование инструкциям пользователя. Модель доступна только через облачный API и недоступна для локального развертывания пользователями, что обусловлено как ее закрытым исходным кодом, так и колоссальными вычислительными требованиями к инфраструктуре [5].

Claude от компании Anthropic – семейство моделей, разработанных с упором на безопасность и управляемость ответов. Архитектурно Claude, подобно GPT-4, является трансформерной моделью высокой размерности (точное число параметров не опубликовано). Особенностью является применение подхода *Constitutional AI* – специального метода обучения, при котором модель ориентируется на заложенные этические принципы вместо жестких запретов. Это позволяет Claude генерировать осмысленные и принципиальные ответы на сложные вопросы, реже отвечая отказом.

Модель Claude известна самым большим окном контекста среди публичных LLM на 2024 год: до **100 000 токенов** ввода за один сеанс. Такой объём контекста (эквивалент ~75 тыс. слов) позволяет Claude анализировать очень длинные документы – например, полный текст книги или технического

отчёта – и вести глубокие многошаговые рассуждения без утраты ранее упомянутых деталей.

Архитектура Claude не предусматривает собственный механизм веб-поиска или подключения плагинов; модель опирается только на данные своего обучения и предоставленный пользователем контекст. Также в версии Claude 2/3 реализована обработка изображений: пользователь может загрузить картинку (например, график или фотографию), и модель опишет или проанализирует её. При этом генерации изображений (как у некоторых конкурентов) Claude не умеет. Доступ к Claude предоставляется через веб-интерфейс и API Anthropic, модель закрытая и работает на серверах разработчика [6, 7].

DeepSeek – новая LLM, разработанная в 2025 году китайской исследовательской лабораторией, привлекающая внимание благодаря сильным возможностям и свободному доступу для пользователей. Архитектура DeepSeek основана на подходе *Mixture-of-Experts* (смешение экспертов) – модель состоит из набора экспертов-моделей, между которыми специальный механизм маршрутизирует запросы. Суммарный размер модели колоссален – сообщается о **671 млрд параметров** в совокупности по всем экспертам, однако в процессе ответа активна лишь часть экспертов (не все параметры одновременно), что повышает эффективность. Таким образом, эффективное число активных параметров оценивается на уровне ~37 млрд на токен, что сокращает требуемые вычисления без заметной потери качества.

DeepSeek предоставляет две основные версии модели: **V3** – универсальный многозадачный вариант (силён в программировании, математике, языковых задачах), и **R1** – специализированный на логическом рассуждении и автономном анализе вариант. Пользователь может выбирать режим работы (V3 или R1) под свой запрос.

Благодаря современной архитектуре DeepSeek демонстрирует высокую производительность при решении широкого круга задач, оставаясь при этом относительно быстрой. Отдельно стоит отметить, что DeepSeek изначально была доступна бесплатно через веб-интерфейс (без регистрации и ограничений по

сообщениям), что резко контрастирует с закрытыми платными GPT-4 и Claude. Алгоритмы DeepSeek позволяют обрабатывать длинные диалоги и большие тексты (многостраничные документы), хотя официально максимальное окно контекста не указано – оценки говорят о порядке **~32 тыс. токенов** ввода без деградации качества.

В отличие от некоторых конкурентов, DeepSeek фокусируется только на работе с текстом: по состоянию на 2025 год модель не поддерживает генерацию изображений или прямой веб-поиск, ограничиваясь информацией из обучающих данных. Тем не менее, благодаря эффективной реализации и открытости, DeepSeek стала привлекательной платформой для разработчиков: исходные коды и веса модели DeepSeek-R1 были выпущены в открытый доступ под лицензией MIT, а для сообщества подготовлены облегченные дистилляции модели (с вместимостью 32B и 70B параметров), упрощающие локальный запуск [8, 9, 10].

Grok – крупная языковая модель, созданная компанией **xAI** (основанной Илоном Маском). Grok выделяется своей концепцией *“truth-seeking”* – стремлением давать максимально правдивые и актуальные ответы, а также более раскованной, «остроумной» персональностью в общении. Архитектурно Grok во многом близок к DeepSeek, применяя смесь экспертов: первая версия Grok-1 имела **314 млрд параметров** и реализована как MoE-модель с активацией ~25% параметров на каждый токен. Эти базовые веса Grok-1 были открыты xAI в 2024 году для исследователей. Далее модель активно дорабатывалась: по некоторым данным, текущая версия **Grok-3** масштабирована до trillions-параметрного диапазона (возможно >2 трлн параметров) и оснащена очень большим контекстным окном (до **128 000 токенов**) [11].

Важным отличием Grok является интеграция модулей рассуждения и поиска: в режиме “Think” модель может явно показывать пользователю свои пошаговые размышления (цепочку рассуждений) при решении задачи. А режим “DeepSearch” выполняет автоматический интернет-поиск по запросу, извлекая свежую информацию в реальном времени. Таким образом, Grok обладает

встроенным доступом к актуальным данным (например, через поиск по постам в X/Twitter) и может цитировать найденные источники в своих ответах. Это уникальное на 2025 год свойство среди рассматриваемых моделей, позволяющее Grok отвечать на вопросы о самых последних событиях.

Кроме того, Grok – мультимодальная система: помимо текста она умеет генерировать изображения (имеет встроенный генератор *Aurora*) и анализировать загруженные картинки. Пользователь может попросить Grok создать иллюстрацию или мем; также доступна отправка изображения на анализ (например, описание содержимого фото). Такая широкая функциональность требует значительных вычислительных ресурсов, поэтому Grok доступен преимущественно через облачные сервисы X. Модель была интегрирована в платформу Twitter (X) для подписчиков сервиса X Premium/Premium+ и доступна через отдельное веб-приложение grok.com. Grok – проприетарная модель (исходный код закрыт, за исключением базовой версии Grok-1); для доступа к полной версии Grok-3 требуется платная подписка, что отличает ее от полностью бесплатного DeepSeek [12, 13].

2.2 Сравнение моделей по интеллектуальным метрикам

2.2.1. Общие интеллектуальные способности и логика

Рассматриваемые модели принадлежат к новому поколению больших языковых моделей (LLM), демонстрирующих человеко-сопоставимую производительность на ряде интеллектуальных и профессиональных задач. Все четыре модели (GPT-4, Claude, DeepSeek и Grok) уверенно справляются с широким спектром знаниевых и логико-аналитических тестов – от школьных экзаменов до междисциплинарных олимпиад. Так, GPT-4 успешно проходит юридический барьерный экзамен, входя в топ-10% лучших результатов среди

сдающих, и стабильно демонстрирует высокие оценки в академических сценариях [14].

В известном междисциплинарном тесте MMLU (Massive Multitask Language Understanding), измеряющем уровень знаний по множеству предметов, модели GPT-4, Grok-3 и DeepSeek R1 достигают показателей выше 90%. Так, Grok-3 показывает ~92,7%, DeepSeek R1 – около 90,8%, что сопоставимо с результатами GPT-4. Claude, по внутренним оценкам, также находится на том же уровне, демонстрируя стабильные результаты в логических, языковых и аналитических тестах [15].

Отдельного внимания заслуживают результаты по математическим и логическим задачам. Например, на датасете GSM8K, включающем сложные многошаговые арифметические задачи, DeepSeek-R1 достигает точности около 90,2%, немного превосходя Grok-3 (~89,3%) и приближаясь к лучшим версиям GPT-4. В задачах здравого смысла (common sense QA) все модели стабильно набирают более 90%, с минимальными различиями между ними. Таким образом, все четыре модели входят в топ мировых LLM по уровню «аналитического интеллекта» на 2025 год, с отрывом между ними в пределах нескольких процентов [16].

2.2.2. Задачи программирования

Для оценки способностей к написанию и пониманию кода часто используют бенчмарк HumanEval (набор задач по написанию небольших функций). Модель GPT-4 установила новый стандарт качества в программировании, верно решая ~80–85% заданий HumanEval (для сравнения, GPT-3.5 решал лишь около 50%). Конкуренты близки к этому уровню. Так, **Grok-3** достигает ~86,5% по HumanEval, немного опережая GPT-4. Модель **DeepSeek** (режим V3/R1) показывает результат примерно на уровне GPT-4 – т.е. уверенно решает подавляющее большинство кодовых задач [17].

Производительность Claude в кодинге также высока: Claude генерирует корректные решения и объяснения, хотя из моделей Anthropic больше

акцентирован на диалог, чем на точность кода. По некоторым тестам Claude 2/3 немного уступает GPT-4 в сложных алгоритмических задачах, но разница не принципиальна. Следует отметить, что Claude часто требует меньше уточняющих вопросов для понимания задачи – благодаря большому контексту он может сразу воспринять длинное техническое описание или несколько файлов кода и дать ответ [18].

DeepSeek заслужил репутацию «мощной альтернативы Copilot» для разработчиков – пользователи отмечают, что модель отлично пишет код, объясняет алгоритмы и даже превосходит GPT-4 в некоторых соревнованиях по спортивному программированию. Быстродействие DeepSeek при генерации кода также является плюсом (модель менее склонна к “размышлениям” и выдает ответ очень оперативно). Grok, помимо генерации кода, обладает уникальными возможностями: он умеет читать диаграммы или схемы и генерировать код по ним (заявлена способность интерпретировать блок-схемы и на их основе писать готовый код). Grok также успешно оптимизирует и отлаживает предоставленный пользователем код, логически рассуждая о возможных ошибках [19].

Таким образом, все четыре модели подходят для задач программирования, обеспечивая высокий процент правильных решений. Небольшое превосходство Grok и GPT-4 в стандартных метриках компенсируется особыми преимуществами Claude (контекст на большие проекты) и DeepSeek (скорость и бесплатность), поэтому выбор конкретной модели может зависеть от поставленной задачи.

2.2.3. Творческие задачи

Под творческими способностями понимается умение модели генерировать развернутые тексты в художественном или креативном стиле, придумывать нестандартные решения, шутки, сценарии и т.п. Здесь наблюдаются более заметные различия в «личностях» моделей. GPT-4 зарекомендовал себя как мощный инструмент для генерации осмысленных и структурированных текстов

практически в любом стиле – от стихотворений до эссе. Пользователи отмечают, что GPT-4 хорошо справляется с генерацией оригинальных идей и продолжительных связных рассказов, показывая высокий уровень креативности [20].

Claude обладает наиболее «человечным» и эмпатичным тоном из всех: его ответы зачастую наиболее теплые, с пониманием эмоций, модель готова рассуждать на философские и этические темы. Это делает Claude отличным собеседником для “брейнсторминга” или обсуждения деликатных вопросов – там, где требуется творческое и вдумчивое участие ИИ. Claude генерирует развернутые ответы и может подробно разъяснять свои мысли, хотя иногда бывает излишне многословен. В креативном письме (напр. сочинение рассказов) Claude и GPT-4 обычно выступают лучше остальных, выдавая более глубокие и стилистически проработанные тексты [21].

DeepSeek, напротив, придерживается фактического и лаконичного стиля. Её ответы можно охарактеризовать как «без излишеств» – модель фокусируется на фактах и прямом решении задачи, не показывая ярко выраженной фантазии или юмора. Это приводит к тому, что при решении строго технических задач DeepSeek эффективен, а вот в генерации креативного контента (например, сюжетов, шуток) он может уступать GPT-4 и Claude, которые склонны к более богатым формулировкам. Тем не менее DeepSeek может создавать связные тексты и сценарии – просто стиль будет более сухим и академичным [22].

Модель Grok обладает уникальной «личностью»: по задумке разработчиков, Grok отвечает с дерзостью и юмором. В его ответах часто присутствуют шутливые или неформальные интонации, современные отсылки. Это делает Grok интересным в применении для неформальных творческих задач – например, генерации мемов, шуток, нестрогих текстов. Пользователь даже может настраивать степень «безумности» тона Grok – вплоть до режимов, где он будет отвечать весьма раскованно и неоднозначно. Однако подобный стиль может быть нежелателен в профессиональной или академической обстановке. С точки зрения именно *качества* творческого контента (богатство

образов, сложность сюжета), Grok пока не превосходит GPT-4 – независимые отзывы указывают, что по глубине и связности генерируемых рассказов Grok все еще «ищет своё место» и немного уступает более зрелым моделям [23].

В итоге по критерию творчества GPT-4 и Claude можно считать лидерами благодаря сочетанию развитого языка и сложного обучения на широком корпусе текстов, тогда как Grok берет свое оригинальностью стиля, а DeepSeek – прагматичной ясностью изложения (что ценнее в технических текстах, чем в художественных).

2.3. Анализ вычислительных требований

Анализ архитектур показывает, что все рассматриваемые модели являются чрезвычайно ресурсоемкими и требуют современных аппаратных мощностей для обучения и функционирования. **GPT-4** и **Claude** – закрытые проприетарные системы, разворачиваемые только на серверах OpenAI и Anthropic соответственно. Пользователи взаимодействуют с ними через облачные **API** или веб-интерфейсы (например, ChatGPT Plus для GPT-4, claude.ai для Claude). Локальный запуск этих моделей на оборудовании конечных пользователей невозможен: помимо юридических ограничений, их объем (сотни миллиардов параметров) требует десятков GPU с высокой памятью для инференса в приемлемое время. По оценкам экспертов, обучение GPT-4 стоило порядка **100 млн долларов** и задействовало специализированные суперкомпьютеры с тысячами графических процессоров. В самом техническом отчете OpenAI указано, что для достижения предсказуемой обучаемости пришлось разработать инфраструктуру, масштабируемую почти линейно на разностях до 1000× в объеме модели – то есть GPT-4 обучалась на несопоставимо больших вычислительных мощностях, чем предыдущие версии.

Claude аналогично обучалась на больших облачных кластерах; точные затраты не раскрывались, но известен факт, что контекст 100k токенов потребовал разработки эффективных алгоритмов управления памятью и

внимания, что усложнило модель. В результате API-вызовы GPT-4 и Claude относительно дорогие (у OpenAI тарификация ~\$0.03 за 1К токенов ввода для GPT-4), и доступ к ним часто лимитирован по скорости и количеству запросов для одного пользователя [24].

DeepSeek изначально разрабатывалась с прицелом на более доступную и открытую альтернативу. Благодаря архитектуре MoE и другим оптимизациям, создателям удалось значительно снизить затраты на обучение: сообщается, что обучение DeepSeek-V3 обошлось менее чем в **\$6 млн**, то есть в десятки раз дешевле, чем GPT-4. Кроме того, разработчики **открыли модель DeepSeek-R1** для сообщества, выложив весовые коэффициенты и исходный код под свободной лицензией. Это означает, что облегченные версии DeepSeek можно запустить локально при наличии достаточных ресурсов – например, 32-миллиардная или 70-миллиардная дистиллированная модели требуют одной или нескольких мощных GPU (A100/H100) и доступны на платформах типа HuggingFace.

Таким образом, **DeepSeek – единственная из сравниваемых моделей, имеющая полноценную опцию локального развёртывания**. Сама по себе полноразмерная версия V3 (671B параметров) вряд ли пригодна для персонального запуска, но можно задействовать ее через официальный API. DeepSeek предоставляет как бесплатный публичный API (ограниченный), так и коммерческий API с низкой ценой: около **\$0.0008 за 1К токенов** ввода в стандартном режиме, что существенно дешевле OpenAI.

Для корпоративных клиентов DeepSeek позиционируется как недорогое и гибко интегрируемое решение (открытый исходный код позволяет встраивать модель в свои продукты). Однако при использовании публичного облачного сервиса DeepSeek возникали вопросы безопасности: в 2025 году имел место инцидент утечки данных, вызвавший обеспокоенность по поводу приватности запросов. Разработчики заверили, что в бесплатном чате DeepSeek не сохраняет историю сообщений на сервере (сеансы анонимны), но все же при работе с конфиденциальной информацией некоторые компании предпочитают развернуть модель локально, чем отправлять данные на внешний сервис.

Grok занимает промежуточное положение. С одной стороны, xAI открыли исходники Grok-1 (базовой модели) и выложили ее веса в открытый доступ. Любой желающий может загрузить эти 314-миллиардные веса и экспериментировать с моделью – в научных целях или для разработки. Однако полноценная актуальная модель Grok-3 остается закрытой и развернута на инфраструктуре X (Twitter). Для доступа к ней требуется подписка X Premium (8 долл/мес) или Premium+ (~30 долл/мес). Подписчики получают через API или веб-интерфейс неограниченный доступ к Grok-3, тогда как бесплатный режим, введенный в конце 2024 года, позволяет лишь до 10 запросов каждые 2 часа.

Таким образом, для интенсивного использования Grok в приложениях нужно заключать коммерческое соглашение с xAI либо ограничиваться возможностями открытой версии Grok-1. В вычислительном плане Grok-3 – одна из самых мощных моделей, требующая огромных ресурсов. Компания xAI построила для ее обучения собственный суперкомпьютер Colossus, заявленный как самый мощный в 2024 году. Parametric-ускорение (смещение экспертов) даёт Grok преимущество: например, при 2.7 трлн общих параметров активными могут быть значительно меньше, снижая нагрузку.

Тем не менее, интеграция Grok с реальным временем (поиском в интернете, генерацией изображений) означает, что помимо самой модели задействованы и внешние сервисы (поисковые движки, рендеринг картинок), что усложняет инфраструктуру. Grok является облачным сервисом, не предлагающим возможности локального размещения (за исключением экспериментов с Grok-1). Ввиду связи Grok с аккаунтом X, для компаний могут возникать риски конфиденциальности: все запросы проходят через сервера X, а переписка с Grok потенциально привязана к профилю пользователя [25].

Таблица 1 ниже обобщает некоторые известные численные характеристики моделей – количество параметров и максимальный размер контекстного окна.

Таблица 1 – Сравнение архитектурных параметров моделей

Модель	Количество параметров	Архитектура	Максимальный контекст (токенов)	Мультимодальность	Доступность	Цена за 1 млн. токенов (ввод / вывод)
<i>GPT-4</i>	оценочно 1–1,5 трлн	Dense Transformer	8192 (обычно), до 32 768	Текст + изображение	API	\$30 / \$60
<i>Claude</i>	оценочно ~100 млрд	Dense Transformer	до 100 000	Текст + изображение	API	Opus: \$15 / \$75; Sonnet: \$3 / \$15
<i>DeepSeek V3</i>	671 млрд (MoE) ~37 млрд активных	Mixture-of-Experts	~32 000 (оценочно)	Текст + изображение	API + OpenSource	\$0.27 / \$1.10
<i>Grok-3</i>	> 2 трлн (MoE, активн. доля не разгл.)	Mixture-of-Experts	до 128 000 (по отчётам)	Текст + изображение	Нет	Без лимита (подписка \$16/мес.)

Как видно, Claude обеспечивает наибольшее окно контекста (100k), позволяя удерживать в памяти чрезвычайно длинные цепочки сообщений или документов. Grok-3 заявлен как имеющий еще больше (128k), но эти данные неофициальны; в открытых источниках подтверждено, что контекст Grok не менее 8–16k токенов, что уже достаточно для большинства диалогов. GPT-4 в версии API доступен с контекстом 8k или 32k токенов.

DeepSeek явно не афиширует размер окна, но практические испытания показывают уверенную работу с десятками страниц текста, т.е. не менее ~30k токенов ввода модель обрабатывает полноценно. Количество параметров указывает на общую сложность модели: GPT-4 и Claude – плотные (dense)

модели без разрежения экспертов, тогда как DeepSeek и Grok достигают своих огромных размеров за счет архитектуры MoE. Это затрудняет прямое сравнение «по числу параметров», но в целом можно заключить, что **GPT-4 и Grok-3 – наиболее крупные и требовательные модели** (терапараметрический масштаб), Claude – несколько менее емкий, а эффективная сложность DeepSeek сопоставима с моделями ~30–70 млрд параметров, что облегчает ее развёртывание.

2.4. Оценка применимости к задачам обработки баг-репортов

Одной из прикладных областей использования LLM является автоматизированный анализ текстовых обращений пользователей – например, заявок в службу поддержки или баг-репортов о неисправностях программ. Все рассматриваемые модели обладают развитым натурально-языковым интеллектом, достаточным для понимания описаний проблем, выделения ключевых симптомов и даже предложения возможных решений. Однако между моделями имеются различия, влияющие на их эффективность и удобство в данном прикладном сценарии.

2.4.1. Понимание технического языка и логики

Баг-репорты часто содержат техническую лексику, логи, стек-трейсы ошибок, куски кода. Модели GPT-4 и DeepSeek особенно сильны в таких условиях: GPT-4 обучен на огромном массиве программных и технических данных, а DeepSeek специально оптимизирован под задачи программирования и математики. Оба могут разобрать длинный лог-файл, понять, где произошло исключение, и предложить причину. Claude благодаря своему 100k контексту способен **прочитать целиком большой баг-репорт с вложенными файлами** (например, логи, конфигурации) и не упустить деталей. Это дает ему

преимущество при анализе сложных инцидентов, где информация распылена по многим источникам: Claude может загрузить несколько документов сразу и ответить на вопрос по совокупности сведений.

DeepSeek с контекстом ~32k тоже покрывает большинство кейсов, хотя крайне длинные отчеты (например, сборник из множества логов по разным модулям) ему, возможно, придется обрабатывать по частям. Grok при решении подобных задач может воспользоваться **встроенным интернет-поиском**: если баг связан с известной проблемой, модель автоматически найдет упоминания этой ошибки или похожие случаи онлайн. Например, Grok способен по сообщению об ошибке найти соответствующий тикет на StackOverflow или патч в репозитории и включить эту информацию в ответ. Это крайне ценная функция при разборе обращений: модель не ограничена своим тренировочным корпусом, а подтягивает актуальные данные и решения (с указанием источников) [26].

GPT-4 в стандартной конфигурации не выполняет веб-поиск, но при использовании с включенным режимом Browsing (в ChatGPT Plus) также может находить онлайн-ресурсы по описанию проблемы. Claude и DeepSeek, напротив, ориентированы на закрытый контекст и не дополняют ответы внешними данными – они полагаются на знания, полученные при обучении, и на сам текст обращения [27].

2.4.2. Точность и глубина анализа

При применении LLM для диагностики багов важно, чтобы модель не **галлюцинировала** – то есть не выдумывала несуществующих причин или решений. Здесь многое зависит от внутренней надежности модели. GPT-4 имеет высокие показатели фактической точности и в большинстве случаев дает верные объяснения (OpenAI специально обучали его неуверенно отвечать, если данных недостаточно). Claude, благодаря **Constitutional AI**, старается быть честным и полезным; он может указать, что требуется дополнительная информация, вместо того чтобы придумывать нечто необоснованное. DeepSeek, как более

«прямолинейная» модель, обычно придерживается фактологии: её стиль ответа скорее сухой и конкретный, что снижает риск вымысла [28].

Grok, обладая более раскованным стилем, теоретически мог бы отклониться в сторону юмора или неформальных комментариев, что нежелательно в разборе баг-репортов. Однако xAI предоставляет возможность настраивать тон Grok; в контексте технических обращений можно перевести его в серьезный стиль. В целом по достоверности выводов все четыре модели близки: при правильной постановке вопроса они способны развернуто и по существу проанализировать обращение. Исследования подтверждают высокую эффективность GPT-4 и Claude в задачах отслеживания и устранения ошибок. Например, в эксперименте по поиску конфигурационных багов умного дома **GPT-4 и Claude 3.5 успешно предложили верную стратегию исправления в 80% случаев**, анализируя описание проблемы и исходные скрипты. Это свидетельствует, что современные LLM уже достигают полезного уровня точности в реальных сценариях отладки [29, 30].

2.4.3. Учет контекста и длинных диалогов

Анализ обращений часто требует **многоходового диалога**: модель может задавать уточняющие вопросы, пользователь — отвечать, и по цепочке вырабатывается решение. Здесь важна долговременная память модели в рамках сессии. Claude специально разрабатывался для длительных обсуждений — он помнит детали до 100k токенов и потому идеален для **длинных тикетов и комментариев** (например, переписки разработчиков по багу). GPT-4 с 8k/32k токенами тоже удерживает существенный объем истории, но в очень длинных дискуссиях (сотни сообщений) может начать забывать детали, требуя от пользователя рекапитуляции.

DeepSeek, по отзывам, **обладает конкурентоспособной памятью** — в тестах на длинный контекст он близок к GPT-4. Однако официально DeepSeek не претендует на экстремально большое окно, поэтому при

многоэтапном диалоге свыше ~30k токенов могут потребоваться периодические резюме.

Grok, вероятно, имеет контекст порядка 16k или более (точные данные не публиковались); при этом его интеграция с X дает интересную опцию: Grok может связывать обсуждение багов с реальными репликами из Twitter, что потенциально полезно, если баги обсуждаются в соцсетях. Но обычно анализ обращений происходит в закрытых системах (трекерах), поэтому здесь важнее локальная память.

По непрерывности ведения темы лидирует Claude – он практически не теряет нить разговора даже после множества уточнений. GPT-4 и DeepSeek тоже достаточно устойчивы, а Grok иногда может отвлечься на посторонние детали или шутку, если не держать его стиль под контролем [31, 32].

2.4.4. Эмоциональный интеллект и тон ответов

Важный аспект – особенно для обращений пользователей – это **умение модели вежливо и эмпатично общаться**. Пользователь, сообщаящий о проблеме, может быть расстроен или раздражен; хорошая модель-ассистент должна распознать тон обращения и ответить уместно. Здесь сильной стороной Claude является его эмпатичность: Claude часто **выражает понимание чувств пользователя, приносит извинения за неудобства** и т.п. Такой человекоподобный стиль взаимодействия ценен при автоматическом ответе на обращения, повышая удовлетворенность пользователей [33].

GPT-4 обычно придерживается нейтрально-вежливого тона: он профессионален, но несколько сух, если явно не попросить об эмоциональной поддержке. Тем не менее GPT-4 можно направить через системные инструкции говорить более человечно, и он будет соблюдать стиль. DeepSeek, как отмечалось, отличается **прямым и фактическим стилем** – в ответах на баг-репорт он, вероятно, сразу перейдет к сути проблемы, опустив любезности. Внутри команды разработчиков это не проблема, однако для клиентской

поддержки такой «роботизм» может быть ощутим. Grok по умолчанию шутлив и может в неформальной манере отвечать даже на жалобы («snarky and casual» стиль). В ряде случаев это может разрядить обстановку, но есть риск показаться несерьезным или грубым. В корпоративном контексте обслуживания клиентов подобное поведение, скорее всего, нежелательно [34].

Тем не менее xAI заявляет, что **Grok менее цензурирован и более откровенен**, готов обсуждать то, что ChatGPT или Claude могут отклонить. Это двояко: с одной стороны, Grok может дать честный ответ на острый вопрос, с другой – выше риск, что ответ выйдет за рамки политик компании. Поэтому при применении Grok для анализа обращений (особенно внешних) потребуются дополнительные фильтры и настройки тона, чтобы гарантировать уместность ответов [35].

2.4.5. Конфиденциальность и интеграция

Работа с обращениями часто предполагает, что данные (описание бага, конфигурации, логи) являются внутренними и конфиденциальными. Отправка их во внешние сервисы AI может быть нежелательной по соображениям безопасности. Здесь явное преимущество у **открытого решения DeepSeek-R1** – компания может установить модель на своих серверах и обрабатывать чувствительные баг-репорты локально, без утечки данных за периметр.

Claude и GPT-4, будучи облачными, требуют доверия к OpenAI/Anthropic: эти компании заявляют, что не используют пользовательские данные из API для обучения и соблюдают меры защиты, однако фактически текст обращения передается третьей стороне. В некоторых индустриях (финансы, госорганы) подобное запрещено политиками, и потому закрытые API не подходят.

Grok в этом плане еще сложнее: взаимодействие с ним привязано к аккаунту X, и политика приватности X может быть менее прозрачна. Уже упоминалось, что у DeepSeek был инцидент утечки – вероятно, это произошло

на стороне бесплатного веб-клиента, где не требовалась авторизация. Для API DeepSeek предоставляет возможность **заключить соглашение и обрабатывать данные конфиденциально**, тем более что исходный код открыт для аудита. В интеграции с рабочими процессами тоже есть нюансы: OpenAI и Anthropic предлагают готовые SDK и распространены во множестве платформ (например, плагин ChatGPT уже встроен в инструменты разработчиков, а у Claude есть интеграция в Slack). DeepSeek как новое решение требует дополнительных усилий для интеграции, но благодаря открытому API и низкой стоимости уже используется стартапами для чатботов и анализа документов. Grok ориентирован на использование через X или мобильное приложение, что затрудняет его встраивание в сторонние системы обращения клиентов (нет публично задокументированного SDK кроме как для X) [36].

Таким образом, для задачи анализа баг-репортов на предприятии наиболее практически применимы GPT-4/Claude (как готовые сервисы с наивысшим качеством) или DeepSeek (как настраиваемое решение, выгодное по цене и приватности). Выбор будет зависеть от приоритетов: максимальное качество и поддержка – в пользу GPT-4/Claude, минимальные затраты и контроль данных – в пользу DeepSeek, а доступ к актуальной информации – уникальная черта Grok, которую можно рассматривать для специфических сценариев [37].

3. Разработка программного решения

Автоматизирующая программа-прослойка (АПП) предназначена для упрощения обработки обращений (баг-репортов) от банков и их эффективной маршрутизации внутри компании ООО «ЮБС». Данная система минимизирует ручные действия сотрудников, анализирует поступающие обращения, уточняет недостающие данные, передает их в API выбранной крупной языковой модели для генерации структурированных отчетов и направляет обработанные заявки в соответствующие отделы компании.

До внедрения АПП процесс обработки обращений от банков был полностью ручным (рис. 2).

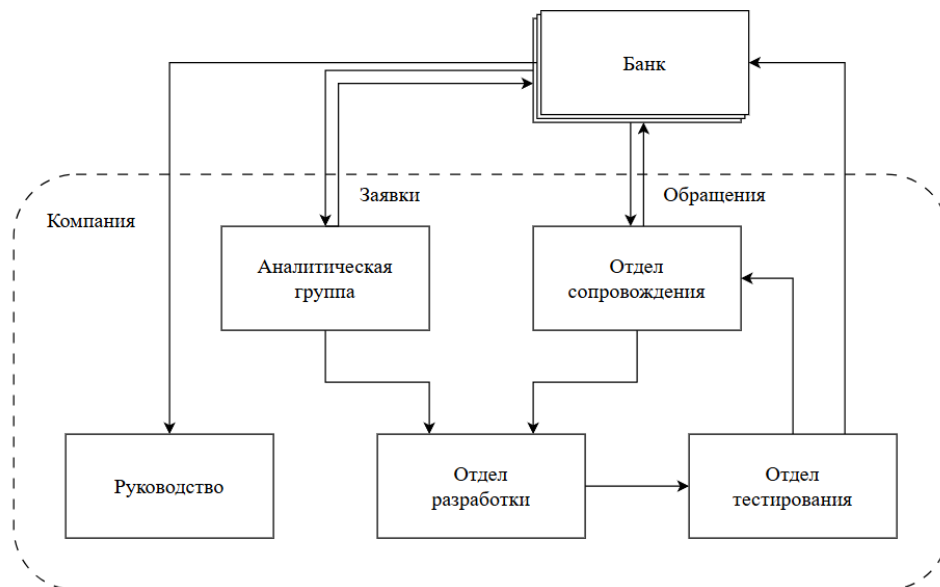


Рисунок 2 – Схема взаимодействия отделов компании до внедрения АПП

Банк отправляет обращение в отдел сопровождения, где сотрудники вручную проверяют описание проблемы, анализируют ее полноту и, если необходимо, запрашивают у банка дополнительную информацию. Затем обращение передается в аналитическую группу, которая оценивает его сложность и определяет, в какой отдел компании направить для решения. Если требуется доработка, обращение поступает в отдел разработки, а после реализации исправлений передается в отдел тестирования.

Основные проблемы процесса до автоматизации:

- Высокая нагрузка на отдел сопровождения – вручную анализировать и уточнять обращения занимает много времени.
- Задержки в обработке – если обращение неполное, его возврат на уточнение и повторная передача создают задержки.
- Отсутствие единого формата – описания ошибок могут быть разными, что затрудняет обработку.

После внедрения АПП в процесс обработки обращений был добавлен автоматический анализатор, который теперь обрабатывает все входящие обращения (рис. 3).

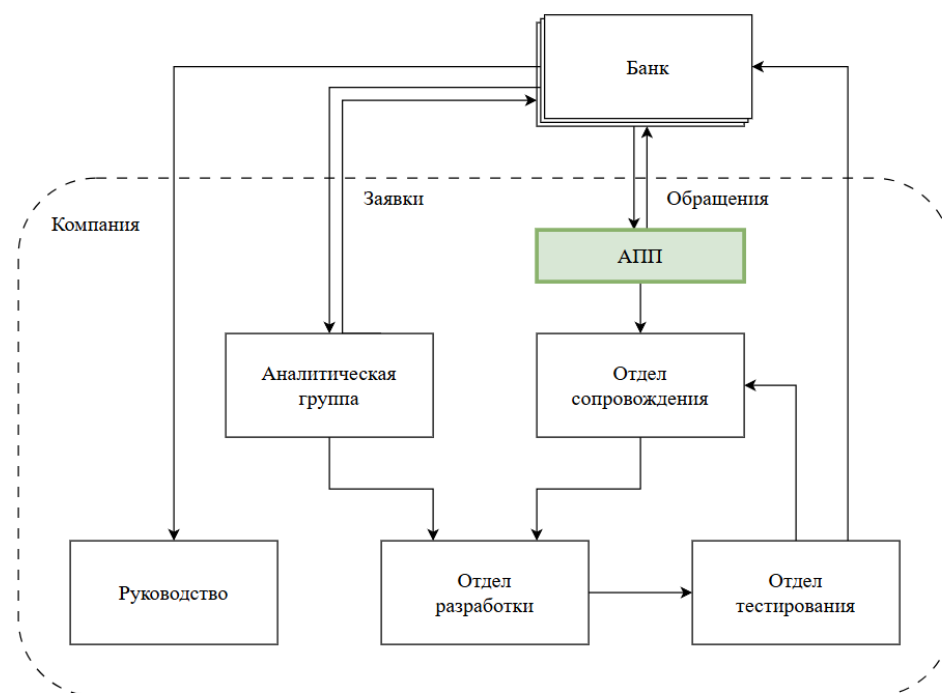


Рисунок 3 – Схема взаимодействия отделов компании после внедрения АПП

3.1. Архитектура системы обработки баг-репортов

Разработанное программное решение представляет собой систему автоматизированной проверки полноты баг-репортов, поступающих от пользователей через сервис-деск Intradesc. Основная цель внедрения данной системы – снижение количества неполных обращений в компании ООО «ЮБС»,

требующих ручной доработки, за счёт интеллектуального анализа текста и автоматического уточнения недостающей информации.

Программа функционирует в пакетном режиме: при запуске она последовательно обрабатывает все обращения, находящиеся в статусе «Принято» и может перевести задачу в одно из следующих состояний (рис. 4).



Рисунок 4 – Статусы обращений в компании

Для каждого обращения выполняется полная цепочка анализа, включающая извлечение текста, его предобработку, оценку содержательной полноты и, при необходимости, генерацию запроса на уточнение. Если по обращению был сформирован комментарий, система переводит заявку в состояние «Передано на уточнение», что сигнализирует о необходимости ответа от пользователя. Если комментарий не требуется, заявка сохраняет текущий статус.

Архитектура системы разделена на функциональные модули, каждый из которых реализует отдельный этап обработки обращения:

- Модуль интеграции с Intradesc – осуществляет подключение к REST API Intradesc и выборку всех заявок в статусе «Принято». Также через этот модуль

выполняется обновление обращений: добавление комментариев и изменение статуса.

- Модуль предобработки текста – применяет регулярные выражения для очистки текста обращения от неинформативных элементов (служебных подписей, повторов, шаблонных фраз) и приведения его к унифицированной структуре. Это значительно упрощает последующий анализ текста и повышает точность интерпретации.

- Модуль анализа полноты – взаимодействует с языковой моделью через внешний API (OpenAI). На вход передаётся очищенный текст обращения и инструкция по выявлению ключевых компонентов. Модель проверяет наличие следующих элементов:

- что делал пользователь до возникновения ошибки;
- какой результат он получил;
- какой результат ожидался;
- почему, по мнению пользователя, ошибка произошла (если указано).

- Модуль формирования уточняющего комментария – если модель обнаруживает отсутствие хотя бы одного из перечисленных компонентов, она формирует соответствующий вопрос. Сформулированный комментарий автоматически направляется в Intradesc в виде сообщения к заявке.

- Модуль изменения статуса – при добавлении уточняющего комментария программа обновляет статус обращения на «Передано на уточнение». Таким образом, технические специалисты могут оперативно отследить, какие обращения требуют дополнительной информации от пользователя.

Принципиальная схема архитектуры системы представлена на рисунке 5. В ней отобразён полный цикл обработки – от получения данных из Intradesc до возможной передачи задачи на доработку пользователю.

предоставляемый данной платформой. API Intradesc позволяет внешним приложениям запрашивать списки заявок, получать подробные данные по конкретной заявке, добавлять комментарии, менять статус и выполнять другие действия, аналогичные ручной работе через интерфейс. Доступ к API защищен с помощью API-ключа, передаваемого в запросах.

В ходе интеграции реализованы следующие основные операции: получение новых баг-репортов, отправка комментариев с запросами уточнений и обновление полей заявок (например, статуса или ответственного). Для осуществления этих операций выполняются HTTP-запросы к соответствующим ресурсам Intradesc.

Создание новой заявки происходит с помощью **POST-запроса**, направляемого на URL:

```
POST https://apigw.intradesk.ru/changes/v3/tasks?ApiKey=<ключ API>
```

Пример тела запроса для создания заявки, помеченной тегом «monitoring» (предварительно созданным и имеющим, например, ID=33290):

```
{
  "blocks": {
    "name": "{\"value\":\"Не работает сервер X1. Просьба починить.\"}",
    "description": "{\"value\":\"Даже не запускается. Лампочки не горят.\"}",
    "tags": "{\"value\":[33290]}"
  },
  "Channel": "web"
}
```

После отправки такого запроса Intradesc создаёт новую заявку, присваивая ей уникальный идентификатор и номер, которые возвращаются в ответном JSON.

Обновление существующей заявки (изменение статуса, назначения исполнителя, добавление комментариев) осуществляется методом **PUT** по аналогичному адресу:

```
PUT https://apigw.intradesk.ru/changes/v3/tasks?ApiKey=<ключ API>
```

Тело запроса содержит идентификатор заявки и блоки с обновляемыми данными:

```
{
  "number": 272,
  "blocks": {
    "status": "{\"value\":\"22058\"}",
    "service": "{\"value\":\"23097/\"}",
    "tasktype": "{\"value\":\"7805\"}",
    "executor": "{\"value\":{\"userid\":\"109747\",\"groupid\":\"31974\"}}",
    "resolutiondateplan": "{\"value\":\"2022-03-15T15:00:00.000000Z\"}",
    "tags": "{\"value\":[32887,32893]}",
    "comment": "{\"value\":\"Постараемся сделать до обеда.\"}",
    "evaluation":
    "{\"value\":{\"value\":3,\"text\":\"Удовлетворительно\"}}"
  },
  "Channel": "api"
}
```

При успешном выполнении запроса Intradesc возвращает код ответа **200 OK** и обновлённые данные заявки.

Для получения списка заявок используется **GET-запрос** с фильтрацией в формате OData. Пример получения первых пяти заявок в статусах «открыта» и «в работе», отсортированных по дате последнего изменения:

```
GET https://apigw.intradesk.ru/tasklist/odata/v3/tasks?ApiKey=<ключ API>&$filter=(status eq 22055 or status eq 22058)&$orderby=updatedat desc&$top=5
```

Подробные данные конкретной заявки можно получить аналогичным способом, фильтруя запрос по номеру заявки:

```
GET https://apigw.intradesk.ru/tasklist/odata/v3/tasks?ApiKey=<ключ API>&$filter=tasknumber eq 31
```

Каждая заявка в системе Intradesc имеет подробную JSON-структуру, включающую такие важные поля, как:

- **id** и **tasknumber** – уникальные идентификаторы заявки.
- **status** – текущий статус заявки (например, «Открыта», «В работе»).
- **priority** – приоритет обработки заявки.
- **name** и **description** – заголовок и описание проблемы.

- **executor** и **initiator** – исполнители и инициаторы заявки, с указанием пользователей или групп.
- **servicepath** – сервис или подсистема, к которой относится заявка.
- **tags** – теги, назначенные заявке для классификации и дальнейшей фильтрации.
- **resolutiondateplan** и **resolutiondatefact** – плановые и фактические сроки решения.
- **attachments** – прикрепленные файлы к заявке.
- **additionalfields** – пользовательские дополнительные поля.

Это позволяет интегрируемой системе эффективно работать с заявками, полноценно автоматизируя процессы обработки.

Intradesc поддерживает добавление комментариев и файлов через API. Для добавления комментария и прикрепления файла необходимо предварительно загрузить файл методом **POST** на сервер Intradesc:

<p style="text-align: center;">POST</p> <p><a href="https://apigw.intradesk.ru/files/api/tasks/{id}/files/target/Comment?ApiKey=<ключ API>">https://apigw.intradesk.ru/files/api/tasks/{id}/files/target/Comment?ApiKey=<ключ API></p>
--

Затем этот файл указывается при отправке комментария и обновлении заявки методом **PUT**:

<pre>{ "number": 130, "blocks": { "comment": "{\\"value\\":\\"Посмотрите приложенные изображения.\\"}", "attachments": "{\\"value\\":{\\"addFiles\\":[{\\"name\\":\\"image.jpg\\",\\"id\\":\\"<идентификатор файла>\",\\"contentType\\":\\"image/jpeg\\",\\"size\\":43558,\\"target\\":30,\\"uploadedAt\\":\\"2024-06-04T13:28:20.435Z\\",\\"uploadedBy\\":\\"система\\"}],\\"deleteFileIds\\":[]}}"} }</pre>

Таким образом, интеграция с Intradesc обеспечила двусторонний обмен данными: наш сервис может **получать входные данные** (новые баг-репорты в формате JSON) и **отправлять выходные данные** (комментарии, изменения

статусов, назначения ответственных лиц, также в JSON). Например, после анализа бага наше приложение может отправить в Intradesc JSON с обновлением поля статуса на значение “Требуется информация от пользователя” и добавлением комментария с запросом уточнений. Intradesc, в свою очередь, обработает этот запрос и отразит изменения в своей базе, сделав комментарий видимым пользователю и изменив статус заявки. Такая интеграция позволяет автоматически продолжить жизненный цикл обращения в системе учета, минимизируя ручной труд оператора.

3.3. Очистка обращений от конфиденциальных данных

В контексте обработки баг-репортов, получаемых из CRM-системы Intradesc, одной из критически важных задач является обеспечение конфиденциальности обрабатываемой информации. Обращения пользователей могут содержать персональные и чувствительные данные, такие как ИНН, ОГРН, номера счетов, паспортные данные, адреса электронной почты и иные идентификаторы. Их передача во внешние сервисы, включая API языковых моделей, недопустима без предварительной очистки.

Для автоматического удаления подобных сведений был реализован модуль парсинга, основанный на применении регулярных выражений. Он анализирует текст обращения и заменяет обнаруженные элементы, описанные в таблице 2, маркерами соответствующего типа. Это позволяет сохранить смысловую структуру обращения без риска утечки персональных данных.

Таблица 2 – Типы обрабатываемых данных и применяемые шаблоны

Тип данных	Пример	Регулярное выражение
ИНН	7707083893	(?<!\d) \d{10,12} (?!\d)
ОГРН	1027700132195	(?<!\d) \d{13,15} (?!\d)
Номер счёта	40817810099910004312	(?<!\d) \d{20} (?!\d)

Продолжение таблицы 2

Тип данных	Пример	Регулярное выражение
Телефон	+7 (495) 123-45-67 8 912 345 67 89 +375-29-1234567	(?:\+?\d{1,3})?[\s\-\]]?(?:\d{3,4}\s)?[\s\-\]]? \d{2,4} [\s\-\]]? \d{2} [\s\-\]? \d{2,4}
Email	user.name+test@example.co.uk	[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}
Паспорт	1234 567890 1234567890	\b\d{4}\s\d{6}\b
Банковская карта	1234 5678 9876 5432 1234567890123456	(?<!\d) (?:\d[-]]?) {13,19} (?!\d)

Все обнаруженные элементы подлежат замене на шаблонные маркеры: например, ИНН 7707083893 → ИНН [ИНН удален].

Этот этап обработки реализуется программно сразу после десериализации обращения и до его передачи в OpenAI API. Тем самым достигается полная защита персональных данных на всех этапах автоматизации.

3.4. Взаимодействие с внешними языковыми моделями

Для семантического анализа текста баг-репорта и генерации рекомендаций применяется внешняя облачная языковая модель. Выбор внешнего API обусловлен высокой точностью и мощностью современных LLM, таких как модели семейства GPT-3.5/GPT-4, при отсутствии необходимости размещать тяжелую модель на локальном сервере. Взаимодействие с моделью происходит через REST API: наше приложение формирует запрос с описанием задачи (промптом), отправляет его на сервер модели и получает сгенерированный ответ.

Технически, для обращения к языковой модели используется HTTP POST запрос к эндпоинту модели. В запросе в теле передается JSON с параметрами генерации. Например, для OpenAI GPT-4 это адрес:

```
POST https://api.openai.com/v1/chat/completions
Authorization: Bearer YOUR_API_KEY
Content-Type: application/json
```

В заголовках передается ключ API для аутентификации (выданный провайдером модели), а тело содержит указание модели, контекст диалога или сообщения с описанием баг-репорта, а также настройки генерации (температура, максимальная длина ответа и пр.). Ниже приведен пример JSON-запроса к API языковой модели:

```
{
  "model": "gpt-4",
  "messages": [
    {
      "role": "system",
      "content": "Ты – ассистент, оценивающий полноту баг-репортов. Ответь в формате JSON."
    },
    {
      "role": "user",
      "content": "Описание: При попытке сохранить профиль кнопка Save ничего не делает. Среда: Chrome 113, Windows 10."
    }
  ],
  "temperature": 0,
  "max_tokens": 200
}
```

В данном примере у модели запрашивается анализ указанного текста бага. Параметр `model` задает конкретную языковую модель (например, GPT-3.5-Turbo), а блок `messages` формирует диалог: системное сообщение может содержать инструкцию отвечать структурировано, а пользовательское – собственно содержимое баг-репорта и вопрос на естественном языке (например, *"проанализируй отчет об ошибке и сформируй вывод в JSON-формате"*).

При выполнении такого запроса языковая модель генерирует ответ, который возвращается также в виде JSON:

```
{
  "is_complete": false,
  "missing_parts": ["Шаги воспроизведения", "Ожидаемый результат"],
  "comment": "Не хватает информации о том, что делал пользователь и чего он ожидал.",
  "next_action": "request_additional_info"
}
```

Провайдер API обычно включает ответ модели в структуру, содержащую метаданные. Так, ответ OpenAI содержит поле `choices`, внутри которого находится сгенерированное сообщение модели.

Наше приложение обрабатывает ответ, извлекая контент, сформированный языковой моделью. Приложение парсит эту строку как JSON, получая структуру данных с необходимыми полями. Далее эти данные используются логическим модулем для принятия решения.

Для сокращения объема передаваемых данных и обеспечения согласованности, промпт для модели формируется компактно, включая только нужную информацию (заголовок и описание бага, без лишних метаданных) и четкие указания по формату ответа.

Применяемый промпт выглядит следующим образом:

Ты – ассистент по качеству баг-репортов в службе технической поддержки.

Твоя задача – проверить баг-репорт ниже и определить, все ли обязательные элементы в нём присутствуют. Если каких-то пунктов не хватает, необходимо сгенерировать вежливый, но деловой комментарий для автора обращения с просьбой добавить недостающие сведения.

Проверь обращение на наличие следующих пунктов:

1. Что делал пользователь?
2. Что он ожидал?
3. Что произошло на самом деле?
4. Почему это может происходить? (необязательный пункт, `is_complete` не зависит от наличия этого пункта)
5. Достаточно ли данных для воспроизведения ошибки?

Если пункт явно не сформулирован, но можно логически понять его суть из контекста – считай, что он присутствует.

Ответ верни в следующем JSON-формате:


```
{  
  "is_complete": true/false,  
  "missing_parts": ["<названия отсутствующих пунктов>"],  
  "comment_to_author": "<сформулированный текст комментария оператору>",  
  "next_action": "ok" или "request_additional_info"  
}
```

Правила для comment_to_author:

- Комментарий должен быть адресован автору баг-репорта.
- Просьбы о доработке должны быть чёткими и конкретными.
- Не пересказывай баг-репорт, не объясняй суть ошибки.
- Просто напиши, что именно нужно уточнить или добавить, чтобы можно было воспроизвести проблему.

Текст обращения:

""{ticketText}""

На стороне модели, благодаря ее обученным возможностям, происходит **анализ текста баг-репорта** с учетом контекста (например, модель может выявить упоминания шагов воспроизведения, окружения, ожидаемого поведения) и **генерация структурированного ответа**.

Конструкция промпта играет решающую роль в эффективности работы языковой модели. От точности формулировок и ясности требований напрямую зависит корректность, структурированность и релевантность выдаваемого ответа. При формировании промпта, используемого в процессе оценки баг-репортов, были учтены как технические ограничения модели, так и практические особенности взаимодействия со службой поддержки.

- **Формулировка роли модели.** Начальная установка «Ты – ассистент по качеству баг-репортов в службе технической поддержки» задаёт модели необходимый профессиональный контекст и стиль общения. Исследования показывают, что корректно заданная роль (role priming) положительно влияет на фокусировку модели на нужной задаче и повышает когнитивную последовательность её ответов.

- **Явное перечисление критериев полноты.** Список из пяти пунктов (действия пользователя, ожидаемый результат, фактическое поведение, гипотеза, воспроизводимость) отражает эмпирически обоснованные компоненты, присутствие которых необходимо для квалифицированного

анализа проблемы. Эти элементы соответствуют рекомендациям по написанию баг-репортов в промышленной практике (см. Bug Advocacy, Cem Kaner) и многократно подтверждены в процессе анализа обращений внутри компании.

Особо стоит отметить, что **пункт 4 – гипотеза причины** – явно обозначен как *необязательный*, что снижает риск ошибочной классификации обращения как неполного из-за отсутствия субъективной аналитики со стороны пользователя. Это сделано для повышения устойчивости метрики `is_complete` и уменьшения числа ложноположительных запросов на уточнение.

- **Указание на контекстную интерпретацию.** Фраза *«Если пункт явно не сформулирован, но можно логически понять его суть из контекста – считай, что он присутствует»* служит для имитации человеческой логики восприятия. Без неё модель склонна требовать дословного присутствия шаблонов («я нажал», «я ожидал» и т. д.), игнорируя естественные описания, которые можно интерпретировать. Такое поведение модели улучшает точность и снижает избыточную требовательность.

- **Задание точного формата вывода.** Формат JSON с чёткими полями (`is_complete`, `missing_parts`, `comment_to_author`, `next_action`) упрощает парсинг и дальнейшую обработку на стороне приложения. Подобная строгость формата минимизирует вариативность и облегчает интеграцию ответа модели в логический модуль системы.

- **Отдельные правила для комментария оператору.** Подсекция с названием «Правила для `comment_to_author`» введена для устранения неоднозначностей в генерации комментариев. В ранних тестах модели склонны были пересказывать содержание обращения или переформулировать его, тогда как задача заключалась в создании краткого и делового запроса о доработке. Четкое разграничение допустимых действий позволило стабилизировать стиль сообщений и адаптировать их к корпоративным стандартам коммуникации.

Применение крупных языковых моделей в области анализа баг-репортов оправдано, поскольку они способны «понимать» неструктурированные описания и извлекать из них смысловую информацию. В исследованиях показано, что

NLP-модели могут проверять наличие ключевых деталей отчета об ошибке (фактическое поведение, ожидаемое поведение, шаги воспроизведения) и оценивать полноту описания. Таким образом, интеграция с внешней языковой моделью через API позволила реализовать интеллектуальный анализ обращений без необходимости создавать с нуля собственную модель обработки естественного языка.

3.5. Пример обращения и результат анализа

В данном разделе представлен подробный анализ практических примеров обработки баг-репортов с использованием различных крупных языковых моделей (LLM). Основной целью анализа являлась оценка полноты описания ошибок и качества формулируемых моделью уточняющих комментариев.

В качестве примеров были выбраны два реальных обращения. Первый баг-репорт представлен на рисунке 6, а его вложение на рисунке 7.

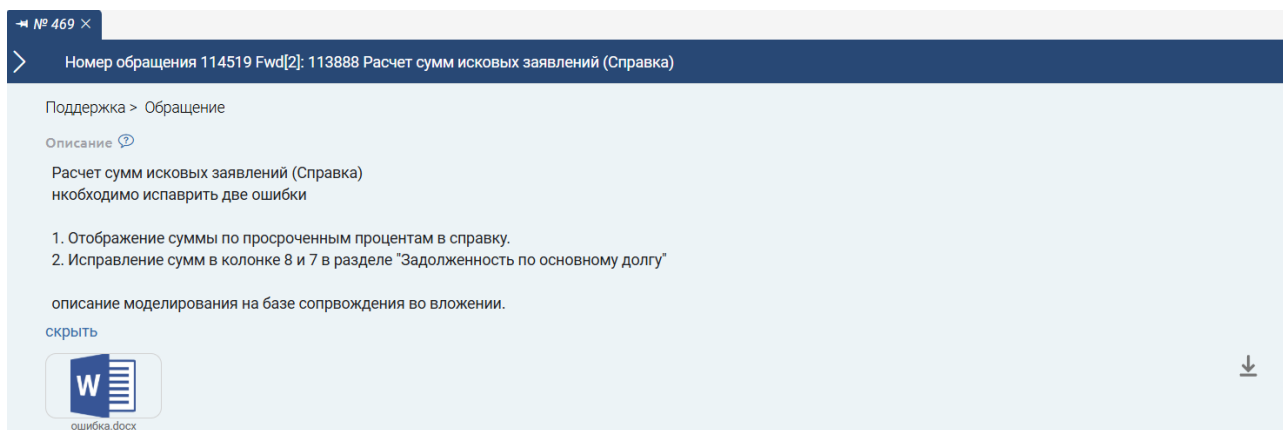


Рисунок 6 – Пример обращения с вложением
по расчетам исковых заявлений

Договор в нашу базу положил
Ид-р 46885
Отчет строим на 20.09.2022
Договор выдавался 25.05.2022 первый и второй платеж ушли и просрочку, потом чуть чуть погасили, потом опять просрочка, суть гашение и 13.08.2022 полный вынос и просрочку и после этого чуть гашений

1. К вопросу о процентах

субсчет по процентам по ОЗ — они только выносились на просрочку

Субсчет - Проценты

Дата	Дебет	Кредит	Остаток
			0,00
31.05.2022	30,14		30,14
30.06.2022	75,74		105,88
01.08.2022	71,99		177,87
13.08.2022	27,74		205,61
Итого:	205,61	0,00	205,61

из картинки видно что сумма там не только процент по ОЗ но и с добавками процент по ПЗ Во второй строк указано $82.19 = 75.74$ (проценты по ОЗ) + 6,45 (проценты по ПЗ)
во вложении файл с движениями средств по договору
Т.е. блок понимает что надо брать проценты по ПЗ, но не берет их полсе того как закончился график — 13.08.2022 А они там есть и точно также копятся, гасятся и выносятся на просрочку

Рисунок 7 – Вложение из обращения по расчетам исковых заявлений

Второе обращение представлено на рисунке 8.

№ 582 ×

В ДБО открываю вклад, ид. типового вклада 103.

Поддержка > Задача

Описание

В ДБО открываю вклад, ид. типового вклада 103.

Получаю ошибку: Открытие депозитного договора. Договор 'До востребования, для Мишиных договоров'. Ставка 12 договора не соответствует расчетной! В типовом сказано что должен открыться договор до востребования под счет уплаты % при закрытии.

Параметры которые передает ДБО в функцию Oper_Open_OutSystem такие:

OutSystem : True :

DateTrn : 30.09.2022 9:34:17 :

DateOpen : 30.09.2022 9:34:17 :

IdCurrency : 810 :

Балансовый номер : :

NumDivision : 6 :

Amount : 1000 :

IdGroupContract : 103 :

IdClient : 22365 :

Period : 1 :

Type : в годах :

OlContract : 66 :

OlAccounts : 44 :

Объявленная ставка процентов : 12 :

DESCRIPTION : Зачисление суммы для открытия вклада (безналично, со счета) :

Дополнительные свойства : lUbsParam :

ContractProperty : | Источник создания | RB |

Рисунок 8 – Пример обращения по открытию вклада в дистанционном банковском обслуживании

Ответ модели **GPT-4** на первое обращение:

```
{
  "is_complete": true,
  "missing_parts": [],
  "comment_to_author": "",
  "next_action": "ok"
}
```

Ответ GPT-4 свидетельствует о том, что модель полностью и корректно распознала все необходимые элементы обращения.

Ответ модели на второе обращение:

```
{
  "is_complete": false,
  "missing_parts": ["Что он ожидал?", "Почему это может происходить?"],
  "comment_to_author": "Пожалуйста, уточните, какого результата вы ожидали при открытии вклада. Также, если есть предположения о причинах возникновения ошибки, просьба их указать – это поможет в более быстром анализе ситуации.",
  "next_action": "request_additional_info"
}
```

Здесь GPT-4 ясно и конкретно указала на нехватку важных деталей и сформулировала полезный уточняющий комментарий.

Ответ модели **DeepSeek** на первое обращение:

```
{
  "is_complete": false,
  "missing_parts": ["Что делал пользователь?", "Что он ожидал?", "Что произошло на самом деле?", "Достаточно ли данных для воспроизведения ошибки?"],
  "comment_to_author": "Уточните, пожалуйста, какие действия вы выполняли перед возникновением ошибки. Опишите, какой результат ожидали получить и что произошло вместо этого. Также проверьте, достаточно ли данных для воспроизведения ошибки (например, указаны ли все необходимые шаги или параметры).",
  "next_action": "request_additional_info"
}
```

DeepSeek проявила излишнюю требовательность и указала недостающие элементы там, где они явно присутствуют или очевидны из контекста.

Ответ модели на второе обращение:

```
{
  "is_complete": true,
  "missing_parts": [],
  "comment_to_author": "",
  "next_action": "ok"
}
```

Здесь модель DeepSeek верно оценила полноту обращения, хотя и ранее продемонстрировала чрезмерную строгость.

Ответ модели **Grok** на первое обращение:

```
{
  "is_complete": false,
  "missing_parts": ["Что ожидал пользователь?", "Что произошло на самом деле?"],
  "comment_to_author": "Уважаемый автор, для дальнейшей обработки вашего обращения просим уточнить: 1) Какое поведение системы вы ожидали при отображении сумм просроченных процентов и остатков в колонках 7 и 8? 2) Что именно произошло в системе вместо ожидаемого результата? Пожалуйста, добавьте эти сведения, чтобы мы могли воспроизвести проблему.",
  "next_action": "request_additional_info"
}
```

Grok также продемонстрировала умеренную строгость, однако упустила контекстные детали, которые были достаточно очевидны.

Ответ модели на второе обращение:

```
{
  "is_complete": true,
  "missing_parts": [],
  "comment_to_author": "",
  "next_action": "ok"
}
```

Оценка второго обращения моделью Grok выполнена корректно, подтверждая её общую пригодность, хотя и с необходимостью более внимательного контроля. Результаты анализа ответов моделей приведены в таблице 3.

Таблица 3 – Результаты обработки обращений языковыми моделями

Критерий оценки	GPT-4	DeepSeek	Grok
Корректность определения полноты	Высокая	Средняя	Средняя
Точность выявления недостающих пунктов	Высокая	Средняя	Средняя
Ясность и конкретность комментария	Отличная	Средняя	Хорошая
Практическая применимость	Высокая	Средняя	Средняя
Общая эффективность	Отличная	Хорошая	Хорошая

Анализ результатов показывает, что модель GPT-4 обладает наиболее высоким уровнем эффективности и точности. В частности, в первом примере GPT-4 полностью и корректно распознала все компоненты баг-репорта как полные и не потребовала дополнительной информации. Во втором примере GPT-4 четко идентифицировала отсутствие ожиданий пользователя и причины ошибки, корректно сформулировав деловой и понятный комментарий:

“Пожалуйста, уточните, какого результата вы ожидали при открытии вклада. Также, если есть предположения о причинах возникновения ошибки, просьба их указать – это поможет в более быстром анализе ситуации.”

Модели DeepSeek и Grok показали меньшую точность, часто требуя избыточную информацию или неправильно интерпретируя имеющиеся данные. Например, DeepSeek в первом примере неверно указала несколько обязательных пунктов как отсутствующие, что привело к формированию слишком общего

запроса на доработку. Grok же частично упустила явные формулировки ожиданий и фактического поведения.

Отзыв руководителя:

“В результате тестирования моделей было выявлено, что GPT-4 показала себя наиболее эффективной в части анализа полноты баг-репортов. Эта модель максимально приближена к экспертной оценке, корректно выявляет недостающие элементы и формирует чёткие, конкретные и деловые комментарии, необходимые для оперативной работы технической поддержки. DeepSeek и Grok также продемонстрировали неплохие результаты, однако их ошибки и неточности требуют дополнительного контроля и могут привести к лишней нагрузке на службу поддержки. Рекомендуется использовать GPT-4 в качестве основной модели для автоматизации анализа обращений.”

Таким образом, результаты практического анализа подтверждают необходимость использования модели GPT-4 для автоматизации обработки баг-репортов с целью повышения качества данных и сокращения ручных операций.

3.6. Организация среды развертывания

Развертывание разработанного решения осуществлено с учетом ограничений по ресурсам вычислительной среды. Система должна функционировать на доступной инфраструктуре, не располагающей неограниченными мощностями (например, на обычном офисном сервере или пользовательском компьютере), и при этом оперативно обрабатывать обращения в реальном времени. Основные ограничения среды – это **аппаратные ресурсы** (оперативная память, число ядер CPU, отсутствие GPU) и **сетевые условия** (возможные задержки или ограниченная пропускная способность интернет-соединения, необходимого для работы с внешним API).

Учитывая, что применение крупных языковых моделей часто требует значительных вычислительных ресурсов, было принято архитектурное решение вынести эту нагрузку на внешний сервис (облачный API). В локальной среде

развертывается только само приложение (Windows Forms клиент), которое потребляет относительно мало памяти и CPU, делегируя тяжелые NLP-вычисления удаленному серверу. Такой подход позволяет системе работать даже на машинах со скромными характеристиками – фактически основные требования локально сводятся к возможности запустить .NET-приложение и обеспечить сетевое соединение. Тем не менее, сетевые вызовы к API вводят задержки на передачу данных и получение ответа. Были предприняты меры по минимизации влияния этих задержек: по возможности компактный объем передаваемых данных (только текст баг-репорта и необходимые параметры), а также параллельная обработка. Например, загрузка нескольких баг-репортов и вызовы к API для них могут выполняться асинхронно, не блокируя основной поток приложения. Благодаря этому даже при наличии сетевой задержки интерфейс остается отзывчивым.

Для обеспечения устойчивости в условиях ограничений были рассмотрены и альтернативные стратегии. В частных случаях, когда соединение с облачным сервисом недоступно или дорого, возможно использование локальных облегченных языковых моделей. Современные подходы предлагают развёртывание LLM на периферии (Edge AI) с применением оптимизаций, таких как квантование моделей, кэширование ключ-значение и т.д., чтобы уменьшить потребление памяти и вычислительных мощностей. В рамках нашего решения заложена возможность переключения на другую модель через абстракцию API, поэтому в перспективе допускается подключение локального NLP-ядра, если ресурсы это позволят. Однако полноценное развёртывание больших моделей в локальной среде с 2-8 ГБ ОЗУ без специализированного оборудования затруднительно, поэтому текущая конфигурация опирается на облачный сервис.

Приложение развернуто в среде Windows (настольная ОС на рабочей станции инженера). Для доступа к API Intradesc и внешнему API использована библиотека HttpClient, входящая в .NET, что не требует дополнительных внешних зависимостей. Это упрощает установку – достаточно исполняемого файла приложения и файла конфигурации для ключей API. С точки зрения

памяти, приложение потребляет порядка десятков мегабайт RAM во время работы, в основном для хранения списка заявок и результатов анализа. Сетевой трафик также относительно небольшой, так как передаются в основном текстовые данные (порядка нескольких килобайт на один запрос к LLM). Таким образом, даже в условиях ограниченных ресурсов (например, запуск на устаревшем ноутбуке либо на виртуальной машине с 1-2 CPU и 4 ГБ RAM) система способна функционировать без существенных тормозов.

В целом, организация среды развертывания сводится к балансированию между использованием облачных мощностей для трудоемких задач и экономным расходом локальных ресурсов для обеспечения плавной работы интерфейса. Такой гибридный подход позволяет воспользоваться преимуществами мощных LLM, не требуя собственного парка GPU-серверов, что особенно важно при ограничениях бюджета и инфраструктуры.

3.7. Оценка стоимости использования языковых моделей

Использование LLM в рамках обработки обращений позволяет существенно автоматизировать работу с баг-репортами. Однако при интеграции внешних API важно учитывать не только технические, но и экономические аспекты. В данном разделе производится количественная оценка стоимости регулярного использования языковых моделей в контексте ежедневной загрузки системы технической поддержки ООО «ЮБС».

Автоматизированный анализ баг-репортов через API языковых моделей включает платную генерацию текстов. Стоимость зависит от:

- модели (GPT-4, Claude, DeepSeek, Grok и др.),
- количества обрабатываемых обращений,
- длины текста обращения,
- объема сгенерированного ответа (в токенах).

Цель оценки – определить, насколько экономически оправдано внедрение LLM и возможен ли их регулярный вызов при текущих нагрузках. По данным мониторинга системы Intradesc, за месяц в компанию поступает:

- **313 обращений** со статусом «Принято» (в среднем, около 10–11 в день).
- Средний объем текста одного обращения (после очистки): **около 900 токенов** (включая заголовок, описание, комментарии).

Запрос к языковой модели обычно приводит к генерации ответа размером 100–200 токенов. Пусть в среднем один анализ включает:

- Входящий текст (prompt): 900 токенов;
- Выход (completion): 150 токенов;
- Итого: 1050 токенов на обращение.

Умножим на количество заявок:

- **313 обращений × 1050 токенов = 328 650 токенов в месяц.**

Для пересчета в миллионы токенов: **328 650 токенов ≈ 0,33 млн. токенов / мес.** В таблице 4 приведем ориентировочную стоимость использования LLM по состоянию на 2025 год.

Таблица 4 – Сравнительная таблица стоимости на 2025 год

Модель	Вход (руб. / 1 млн. токенов)	Выход (руб. / 1 млн. токенов)	Примерная стоимость в месяц (руб.)
GPT-4	~200	~800	~67
Claude	~300	~1500	~109
DeepSeek	~28	~112	~9
Grok	—	—	~1500-2700 (подписка)

Примечание – Оценка дана в рублях по среднему курсу и тарифам на 2025 год.

- **Даже при использовании самой дорогой модели с API Claude** ежемесячные затраты не превышают 110 рублей.

- **Оптимальные по соотношению цена/качество модели**, такие как **DeepSeek V3**, позволяют проводить анализ практически бесплатно – менее 10 рублей в месяц.

- Учитывая экономию времени сотрудников и повышение качества баг-репортов, **затраты окупаются многократно**.

При росте объема заявок (например, до 1000 обращений в месяц), расчетная стоимость возрастает пропорционально (рис. 9).

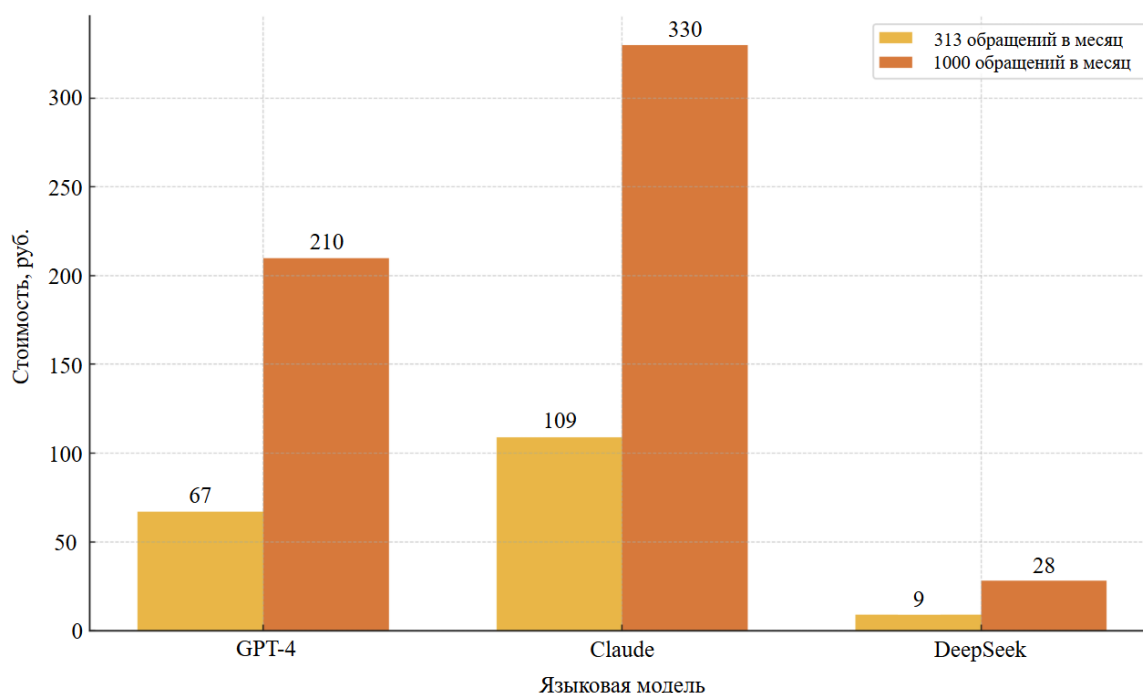


Рисунок 9 – Сравнение стоимости языковых моделей при разной нагрузке

Даже в этом случае:

- Для GPT-4: ~210 Р/мес;
- Для Claude: ~330 Р/мес;
- Для DeepSeek: ~28 Р/мес.

Для доступа к Grok-3 требуется подписка на платные планы X Premium, а стоимость использования не зависит от объёма токенов, что делает невозможным корректное сравнение с моделями, тарифицируемыми по токенам (GPT-4, Claude, DeepSeek) – поэтому он отсутствует на рисунке 6.

Таким образом, интеграция LLM полностью масштабируема без финансовых рисков. Экономическая оценка показывает, что внедрение языковых моделей для анализа баг-репортов является не только технически эффективным, но и крайне экономичным решением. При текущей загрузке поддержка даже платных API обходится в символическую сумму, при этом значительно снижая нагрузку на команду сопровождения. Такая архитектура легко масштабируется и может применяться в любых сервис-деск-системах с аналогичными сценариями.

ЗАКЛЮЧЕНИЕ

Выполненная в рамках данной квалификационной работы исследовательская и практическая деятельность позволила глубоко проанализировать потенциал современных больших языковых моделей в контексте их применения для автоматизации первичного анализа пользовательских обращений, связанных с выявлением и устранением ошибок в программном обеспечении. В ходе теоретического исследования была выполнена комплексная оценка архитектурных особенностей и функциональных возможностей моделей GPT-4, Claude, DeepSeek и Grok. Особое внимание уделялось таким критически важным аспектам, как способность к смысловому обобщению, устойчивость к галлюцинациям, поведение модели в условиях ограниченного или избыточного контекста, а также соответствие стилю общения, ожидаемому в профессиональной среде.

Анализ показал, что модели GPT-4 и Claude обладают наивысшим уровнем универсальности и качества генерации текстов, включая сложные аналитические и диалоговые сценарии. Модель DeepSeek выделяется среди остальных возможностью полноценного локального развёртывания и открытым исходным кодом, что обеспечивает большую степень контроля и адаптируемости, особенно в задачах с повышенными требованиями к конфиденциальности. Grok, в свою очередь, продемонстрировал интересные особенности по части интеграции с внешними источниками информации и неформального взаимодействия с пользователем, что делает его актуальным для отдельных типов обращений, но менее универсальным в корпоративной практике.

В практической части был реализован прототип программной системы, способной автоматически анализировать поступающие обращения с использованием LLM, извлекать ключевые элементы описания ошибки, классифицировать обращения по признакам и формировать первичные рекомендации по их обработке. Были реализованы пользовательский интерфейс и модуль взаимодействия с API модели, интегрированные с системой Intradesc.

Результаты тестирования показали высокую степень релевантности генерируемых ответов, особенно в типичных сценариях, включающих описание симптомов и логи с указанием параметров системы.

Также было подтверждено, что использование LLM существенно снижает когнитивную нагрузку на оператора службы поддержки, ускоряет процесс первичной диагностики и позволяет перейти от формального анализа к содержательной интерпретации обращения. Вместе с тем в рамках экспериментов были выявлены и ограничения, связанные с объёмом контекста, потенциальной уязвимостью к галлюцинациям при недостаточной информации, а также необходимостью предварительной настройки модели на формат обращений конкретной компании. Были предложены пути их компенсации, включая адаптацию подсказок, настройку режима общения и внедрение пользовательских фильтров.

Таким образом, в ходе работы достигнута поставленная цель – подтверждена техническая реализуемость и практическая эффективность использования LLM для автоматизации анализа обращений об ошибках. Разработанный прототип демонстрирует перспективность внедрения таких решений в практику сопровождения программного обеспечения, а полученные результаты могут служить основой для масштабирования и дальнейшего усовершенствования интеллектуальных сервисов технической поддержки в организациях.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Testlio. “What is a Bug Report? How to Write Your Own [+ Template]”. [Электронный ресурс]. Режим доступа: <https://testlio.com/blog/the-ideal-bug-report/> (Дата обращения: 21.03.2025).
2. Requestly. “The Anatomy of a Perfect Bug Report: What It Needs to Contain”. [Электронный ресурс]. Режим доступа: <https://requestly.com/blog/the-anatomy-of-a-perfect-bug-report/> (Дата обращения: 21.03.2025).
3. Atlassian. “Managing Bugs and Issues Effectively.” [Электронный ресурс]. Режим доступа: <https://www.atlassian.com/software/jira/guides/manage-bugs> (Дата обращения: 25.03.2025).
4. Большие языковые модели как инструмент для анализа документации и инцидентов // Habr [Электронный ресурс]. Режим доступа: <https://habr.com/ru/articles/895114/> (дата обращения: 25.03.2025).
5. OpenAI GPT-4 Technical Report // arXiv.org [Электронный ресурс]. Режим доступа: <https://arxiv.org/abs/2303.08774> (дата обращения: 05.04.2025).
6. What is Claude AI and how does it work? // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/what-is-claude-ai-and-how-does-it-work> (дата обращения: 05.04.2025).
7. Claude AI with 100K context window – capabilities and use cases // RedBlink [Электронный ресурс]. Режим доступа: <https://www.redblink.com/claude-ai-context-window-and-use-cases/> (дата обращения: 05.04.2025).
8. DeepSeek AI – возможности и архитектура новой модели // RedBlink [Электронный ресурс]. Режим доступа: <https://www.redblink.com/deepseek-ai-overview/> (дата обращения: 05.04.2025).
9. What is DeepSeek AI? Capabilities and Open-Source Availability // Appy Pie Automate [Электронный ресурс]. Режим доступа: <https://www.appypieautomate.ai/blog/deepseek-ai-overview> (дата обращения: 05.04.2025).

10. DeepSeek API Documentation // DeepSeek [Электронный ресурс]. Режим доступа: <https://api-docs.deepseek.com> (дата обращения: 05.04.2025).
11. Grok AI: особенности архитектуры и режимы работы // RedBlink [Электронный ресурс]. Режим доступа: <https://www.redblink.com/grok-ai-overview/> (дата обращения: 10.04.2025).
12. Grok-1 Technical Release // xAI [Электронный ресурс]. Режим доступа: <https://x.ai/blog/grok-1-open-release> (дата обращения: 10.04.2025).
13. What is Grok AI? Capabilities and Real-Time Search // Appy Pie Automate [Электронный ресурс]. Режим доступа: <https://www.appypieautomate.ai/blog/grok-ai-explained> (дата обращения: 10.04.2025).
14. GPT-4 Technical Report // arXiv.org [Электронный ресурс]. Режим доступа: <https://arxiv.org/abs/2303.08774> (дата обращения: 12.04.2025).
15. Grok AI Benchmarks — Reasoning and Accuracy // Appy Pie Automate [Электронный ресурс]. Режим доступа: <https://www.appypieautomate.ai/blog/grok-ai-benchmarks> (дата обращения: 12.04.2025).
16. LLM Benchmark Comparison: GPT-4 vs Claude vs DeepSeek vs Grok // Appy Pie Automate [Электронный ресурс]. Режим доступа: <https://www.appypieautomate.ai/blog/llm-benchmark-comparison-2025> (дата обращения: 12.04.2025).
17. GPT-4 vs Grok vs DeepSeek: Programming Benchmark Results // Appy Pie Automate [Электронный ресурс]. Режим доступа: <https://www.appypieautomate.ai/blog/llm-programming-benchmark> (дата обращения: 12.04.2025).
18. Claude vs GPT-4 for Coding: Which Is Better? // RedBlink [Электронный ресурс]. Режим доступа: <https://www.redblink.com/claude-vs-gpt4-coding/> (дата обращения: 12.04.2025).
19. DeepSeek: The Fastest LLM for Developers? // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/deepseek-ai-coding-capabilities> (дата обращения: 12.04.2025).

20. Как GPT-4 пишет стихи и рассказы: примеры творческого использования // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/gpt4-creative-writing-examples> (дата обращения: 15.04.2025).

21. Claude AI and Emotional Intelligence in Creative Writing // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/claude-ai-empathy-and-creativity> (дата обращения: 15.04.2025).

22. Почему DeepSeek не шутит: обзор креативных способностей модели // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/deepseek-llm-creativity-limits> (дата обращения: 15.04.2025).

23. Grok AI: юмор, мемы и неформальные тексты — насколько он креативен? // RedBlink [Электронный ресурс]. Режим доступа: <https://www.redblink.com/grok-ai-humor-and-creativity/> (дата обращения: 15.04.2025).

24. Облачные LLM и стоимость их запуска: OpenAI, Anthropic, xAI // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/cloud-llm-cost-comparison> (дата обращения: 20.04.2025).

25. Colossus: The World's Most Powerful AI Supercomputer // LifeArchitect.ai [Электронный ресурс]. Режим доступа: <https://www.lifearchitect.ai/colossus-xai/> (дата обращения: 20.04.2025).

26. Grok AI и автоматический веб-поиск: как работает режим DeepSearch // RedBlink [Электронный ресурс]. Режим доступа: <https://www.redblink.com/grok-ai-deepsearch-mode/> (дата обращения: 22.04.2025).

27. GPT-4 с Browsing: как использовать режим онлайн-поиска в ChatGPT // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/chatgpt-browsing-mode-guide> (дата обращения: 22.04.2025).

28. Как LLM-решения справляются с багами: точность, галлюцинации и честность // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/llm-debugging-hallucination-study> (дата обращения: 27.04.2025).

29. Constitutional AI в модели Claude: предотвращение вымышленных ответов // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/claude-constitutional-ai-explained> (дата обращения: 27.04.2025).

30. LLM for Smart Home Debugging: A Comparative Study // arXiv.org [Электронный ресурс]. Режим доступа: <https://arxiv.org/abs/2403.17492> (дата обращения: 27.04.2025).

31. Claude AI и диалог на 100 тысяч токенов: как устроена память модели // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/claude-ai-context-memory> (дата обращения: 27.04.2025).

32. Сравнение контекста GPT-4, DeepSeek и Grok: какие модели «помнят» лучше? // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/llm-context-window-comparison> (дата обращения: 27.04.2025).

33. Claude и эмпатия: почему он лучший LLM для вежливого общения // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/claude-ai-empathy-in-dialogue> (дата обращения: 28.04.2025).

34. Сравнение стилей общения LLM: GPT-4, Claude, DeepSeek, Grok // Cointelegraph [Электронный ресурс]. Режим доступа: <https://cointelegraph.com/news/llm-conversational-tone-comparison> (дата обращения: 28.04.2025).

35. Grok AI и политика общения: откровенность, юмор и границы дозволенного // Cointelegraph [Электронный ресурс].

Режим доступа: <https://cointelegraph.com/news/grok-ai-communication-policy> (дата обращения: 28.04.2025).

36. Grok AI и конфиденциальность: риски при использовании в X // Cointelegraph [Электронный ресурс].

Режим доступа: <https://cointelegraph.com/news/grok-ai-privacy-and-usage-policy> (дата обращения: 28.04.2025).

37. Интеграция LLM в корпоративные процессы: сравнение решений // RedBlink [Электронный ресурс]. Режим доступа: <https://www.redblink.com/llm-integration-enterprise-solutions> (дата обращения: 28.04.2025).

ПРИЛОЖЕНИЕ А

В графическую часть выпускной квалификационной работы входят:

1. Структура баг-репорта;
2. Схема взаимодействия отделов;
3. Схема взаимодействия отделов после внедрения;
4. Архитектура программного решения;
5. Пример баг-репорта в формате JSON;
6. Пример запроса и ответа языковой модели в формате JSON;
7. Промпт к языковой модели;
8. Сравнение стоимости при разной нагрузке.