УДК 004.4 DOI: 10.31799/2077-5687-2026-2-35-47

АРХИТЕКТУРА АВТОМАТИЗИРОВАННОЙ СИСТЕМЫ УПРАВЛЕНИЯ СВЯЗЬЮ С ПРИМЕНЕНИЕМ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

А. А. Олимпиев[1], Н. В. Шеремет[2]
[1]Санкт-Петербургский государственный университет аэрокосмического приборостроения
[2]Санкт-Петербургский государственный университет телекоммуникаций им. М.А. Бонч-Бруевича, ООО «Лаборатория инфокоммуникационных сетей»

Современные автоматизированные системы управления связью должны решать широкий спектр задач контроля, анализа и оптимизации, адаптироваться к изменениям аппаратного и программного обеспечения элементов сети, предоставлять оператору информативный интерфейс и обоснования своих действий. Для построения таких систем предлагается использование актуальных средств искусственного интеллекта и, в частности, больших языковых моделей как многоцелевой и адаптивной основы с дружественным интерфейсом.

В данной статье приведена информация о результатах, полученных в рамках работы по созданию автоматизированной системы управления связью с использованием технологий искусственного интеллекта. Приведены сведения о преимуществах применения больших языковых моделей с оценкой результата. Описана архитектура разработанной системы, результаты испытаний, использованные технологии. Сделаны выводы относительно успешности полученного опыта и перспективах дальнейшего применения и развития технологий.

Ключевые слова: большая языковая модель, искусственный интеллект, автоматизированная система управления связью, архитектура программного обеспечения.

Для цитирования:

Олимпиев, А. А. Архитектура автоматизированной системы управления связью с применением искусственного интеллекта / А. А. Олимпиев, Н. В. Шеремет // Системный анализ и логистика. – 2026. – № 2(50). – с. 35-47. DOI: 10.31799/2077-5687-2026-2-35-47.

ARCHITECTURE OF NETWORK MANAGEMENT SYSTEM USING ARTIFICIAL INTELLIGENCE

A. A. Olimpiev[1], N. V. Sheremet[2]
[1]St. Petersburg State University of Aerospace Instrumentation
[2]The Bonch-Bruevich Saint Petersburg State University of Telecommunications, Infocommunication Networks Laboratory LLC

Modern network management system must solve a wide range of tasks of monitoring, analysis and optimization, adapt to changes in hardware and software of network elements, provide the operator with an informative interface and justification of their actions. To build such systems, it is proposed to use advanced artificial intelligence tools and, in particular, large language models as a versatile and adaptive basis with a user-friendly interface.

This article provides information on the results obtained in the framework of work on the creation of network management system using artificial intelligence technologies. Information about the advantages of using large language models with an assessment of the result is given. The architecture of the developed system, test results, and technologies used are described. Conclusions are drawn about the success of the obtained experience and the prospects for further application and development of technologies.

Keywords: large language model, artificial intelligence, automated control system for communication, software architecture.

For citation:

Olimpiev A. A. Architecture of network management system using artificial intelligence / A. A. Olimpiev, N. V. Sheremet // System analysis and logistics. – 2026. – № 2(50). – p. 35-47. DOI: 10.31799/2077-5687-2026-2-35-47.

Введение

Искусственный интеллект (ИИ) с каждым все шире применяется в автоматизированных системах управления (АСУ). Одним из популярных направлений является включение инструментов ИИ в АСУ техническими объектами в качестве помощников оператору, что дает дополнительные качества и удобства. Это объясняется в первую очередь сложностью решаемых АСУ задач и их разнообразием.

Одним из наиболее распространенных компонентов АСУ, в которых присутствует ИИ, являются интеллектуальные агенты и интеллектуальные интерфейсы пользователя. Среди элементов интерфейсов пользователя наиболее распространенными являются чат-боты и голосовые помощники, которые увеличивают дружественность системы и ее возможности по обеспечению удобства поиска способов решения проблем, с которыми сталкивается человек во время своей работы, повышают обучаемость пользователя.

Технологически чат-боты и голосовые помощники похожи друг на друга и отличаются в основном способом ввода информации и команд. Их основной функцией является определение потребности пользователя, формализация этой потребности в виде набора алгоритмов (например, поиск сценария управления), выполнение этих алгоритмов и вывод результата (озвучивание или печать полезной информации). Для АСУ также характерна реализация функции оповещения об авариях.

На интеллектуального агента возлагаются функции выполнения сценария контроля и мониторинга, обработка сырых данных, оптимизация и выявление аномалий в работе объекта управления. Концептуальная модель АСУ содержит голосовой помощник и интеллектуальный агент (рис. 1). Представленная модель характерна для любой АСУ и соответствует общим принципам их построения.

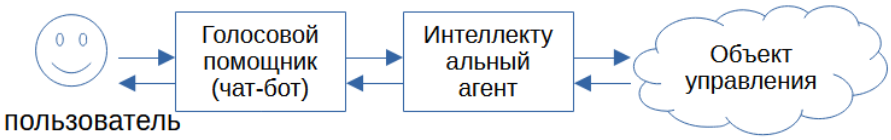

Рис.1. Концептуальная модель автоматизированной системы управления

Одна из основных целей разработки заключалась в проверке пригодности LLM для использования в составе АСУ связью и в особенности тех LLM, доступ к которым предоставляется сторонними провайдерами.

Разработка велась инкрементным способом, который предполагал получить минимальный жизнеспособный продукт за максимально короткое время. Па первом этапе была сформулирована следующая задача:

- разработать гибкую архитектуру, которая позволит интегрироваться с другими частями АСУ связью;
- интерфейс пользователя должен быть максимально удобным;
- должна быть возможность интеграции с облачными сервисами.

1. Большая языковая модель как инструмент решения типовых задач управления

Большие языковые модели или Large Language Model (LLM) — вид генеративного ИИ на основе нейросетей типа трансформер (Generative pre-trained transformer, GPT). LLM в основном используется для распознавания и генерации текста.

Генерация текста такими моделями сводится к предсказанию следующего слова, а точнее фрагмента, токена, в выходной текстовой последовательности. Распознавание текста LLM происходит посредством обработки естественного языка за счёт механизмов векторизации семантики (смыслов), внимания к контексту и обучения на больших объёмах данных.

Генеративные способности могут быть расширены интеграцией дополнительных инструментов: внутреннее исполнение кода позволяет LLM производить точные вычисления, а компьютерное зрение позволяет работать с визуальной информацией, что совокупно ограничивает применение таких моделей для автоматизации только техническими расходами и всё ещё достаточным риском ошибки.

Следующая ступень возможностей больших языковых моделей основана на методике рассуждения, которая в последних моделях становится всё эффективней. Процесс рассуждения в самом простом виде заключается в формировании моделью пошаговых подсказок для стабилизации процесса генерации через разбиение сложной задачи на цепочку маленьких. Данная методика в совокупности с интегрируемыми инструментами превращает LLM в агентную систему на основе целеполагания.

В сфере информационных коммуникаций растёт необходимость в гибких и масштабируемых методах автоматизации управления сетями и сервисами.

Актуальные обзоры и анализ [1] показывают перспективность применения систем ИИ [2] и, в частности, больших языковых моделей [3] в направлениях мониторинга, планирования, развёртывания и менеджмента сетей, в том числе сетей подвижной связи пятого и шестого поколений.

Тенденция к увеличению числа связей типа Машина-Машина (M2M) в сетях Интернета вещей, значительное увеличение плотности размещения устройств в сетях и реализации критически важных структур в рамках концепции Умных городов усложняют мониторинг при неструктурированных данных, и “мыслительные способности” LLM находят применение в глубоком анализе, поиске закономерностей и прогнозировании. Агенты ИИ с доступом к интерфейсам управления способны осуществлять непрерывную поддержку, исправление неисправностей и менеджмент ресурсов при оптимизации сетей.

В [4] авторы предлагают систему для упрощения и автоматизации конфигурации сетевых устройств на основе LLM, демонстрируется способность модели к адаптации под требования задачи. В [5] большая языковая модель используется для определения общих намерений пользователя и разложения их в контролируемый и безопасный набор выполняемых виртуальной сетью действий. Работа [6] рассматривает применение LLM для прогнозирования временных рядов параметров диаграммы направленности при диаграммообразовании в диапазоне миллиметровых волн; авторы делают вывод о перспективности использования больших языковых моделей в данной области.

Проведённый анализ этих и других источников показывает, что применение больших языковых моделей для решения задач сетей радиодоступа современных и перспективных стандартов является актуальной и исследуемой.

2. Архитектура универсального интеллектуального модуля управления

Детальное рассмотрение наиболее распространенных подходов к реализации АСУ объектами разного назначения позволил сделать вывод, что представленная модель (рис. 1), применима для организации схемы управления любыми объектами, начиная от автомобиля, умного дома и заканчивая системами управления технологическими процессами и телекоммуникациями. Разница между ними может быть сведена к предъявляемым критериям на время отклика и реакцию на сценарии управления. Это позволяет создать универсальный модуль управления, который может быть интегрирован как элемент в любую систему управления.

Архитектура такого модуля показана на рисунке 2.

При разработке архитектуры был применен подход, использующий принципы предметно-ориентированного проектирования [7], которые позволяют эффективно разделять систему на части за счет совмещения приемов разделения предметной области на домены и качественного распределения ответственности между частями программной системы.

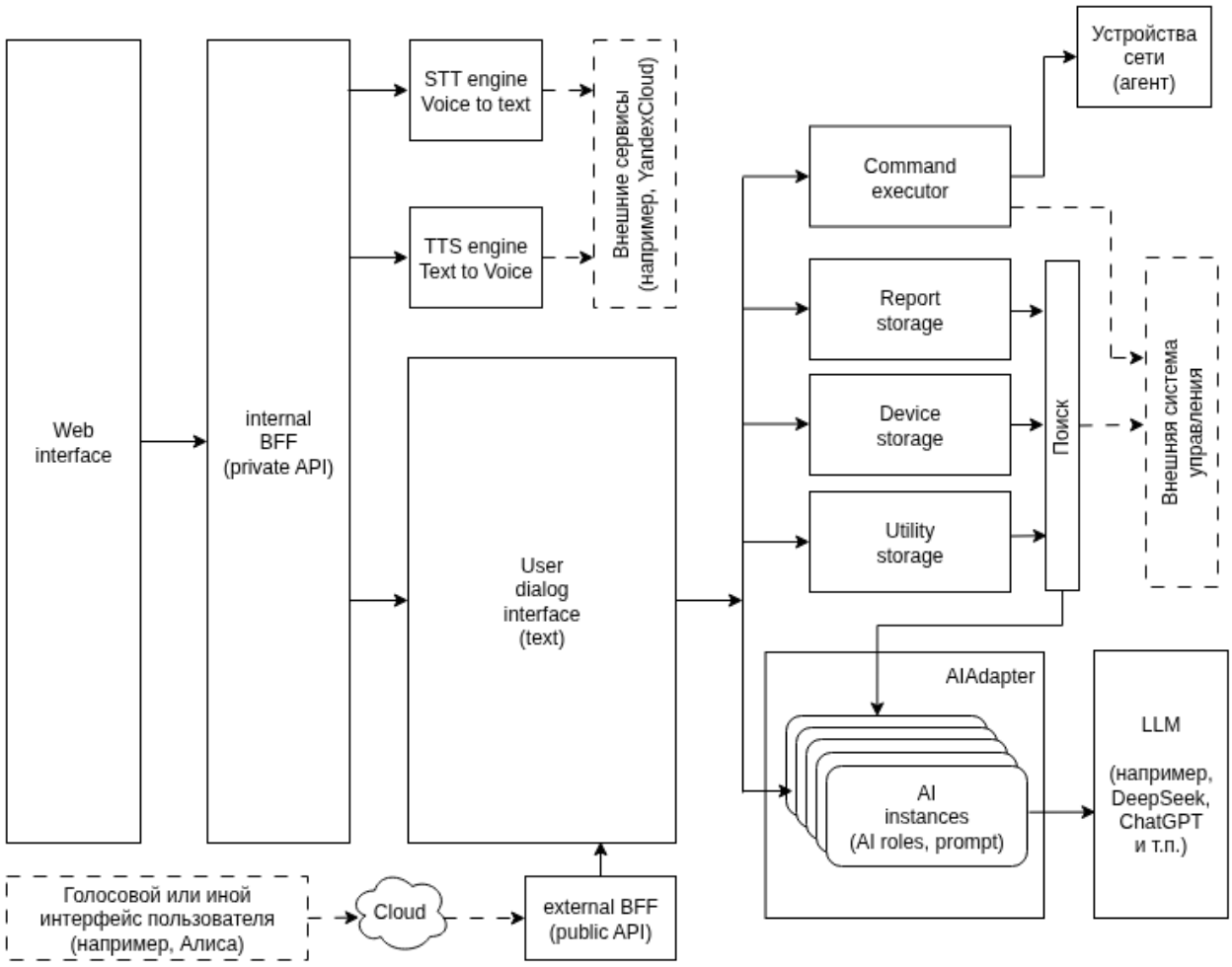

Рис. 2. Архитектура универсального модуля управления

Представленная архитектура (рис. 2) заложена в разработанный опытный образец программного средства, обладающего следующими особенностями:

- применение гибкого подхода для подключения любой LLM;
- возможность подключения модуля к любым хранилищам данных за счет применения механизмов маппинга;
- возможность интеграции с любым интерфейсом управления за счет применения открытого REST API;
- возможность добавления новых функций и настройки уже имеющихся с целью адаптации к конкретным потребностям пользователя и системе управления;
- возможность интеграции с внешними интерфейсами, такими как Алиса.

Для повышения результативности процесса разработки при создании опытного образца применялся подход «разработка через тестирование» [8], который позволил на самом начальном этапе формализовать критерии оценки ожидаемого результата и повысить скорость разработки.

Основные технологии, которые использованы при разработке: LLM, системы контейнеризации Docker и Kubernetes [9, 10], фреймворк Django [11, 12, 13], языки программирования Python, JavaScript, технология создания брокеров RabbitMQ [14]. Для достижения максимальной эффективности полученного результата изучены рекомендации по применению данных технологий, современный взгляд ведущих специалистов на архитектуру

программного обеспечения и особенности реализации соответствующих паттернов проектирования, в частности [15-19].

При разработке в качестве LLM использованы DeepSeek, YandexGPT, Qwen и др.

Применение языка Python, методологии «разработка через тестирование» и фреймворка Django, позволило достаточно быстро (менее, чем за два месяца) создать минимальный жизнеспособный продукт (опытный образец) с высокими показателями сопровождаемости программного кода.

Применение технологии REST для реализации API межмодульного взаимодействия позволило получить гибкую архитектуру на базе асинхронных вызовов и медиации потоков сообщений [15], что дало существенный выигрыш в производительности в том числе при росте количества пользователей, а применение технологии контейнеризации Docker — показателей масштабируемости.

Разработанный модуль состоит из следующих элементов:

- WEB-интерфейс реализован как компонент, который может быть встроен как всплывающее окно, либо как элемент на дашборде, позволяющий вводить сообщения с помощью клавиатуры или голоса, а также получать ответ, который может быть озвучен (пример компонента показан на рисунке 3);

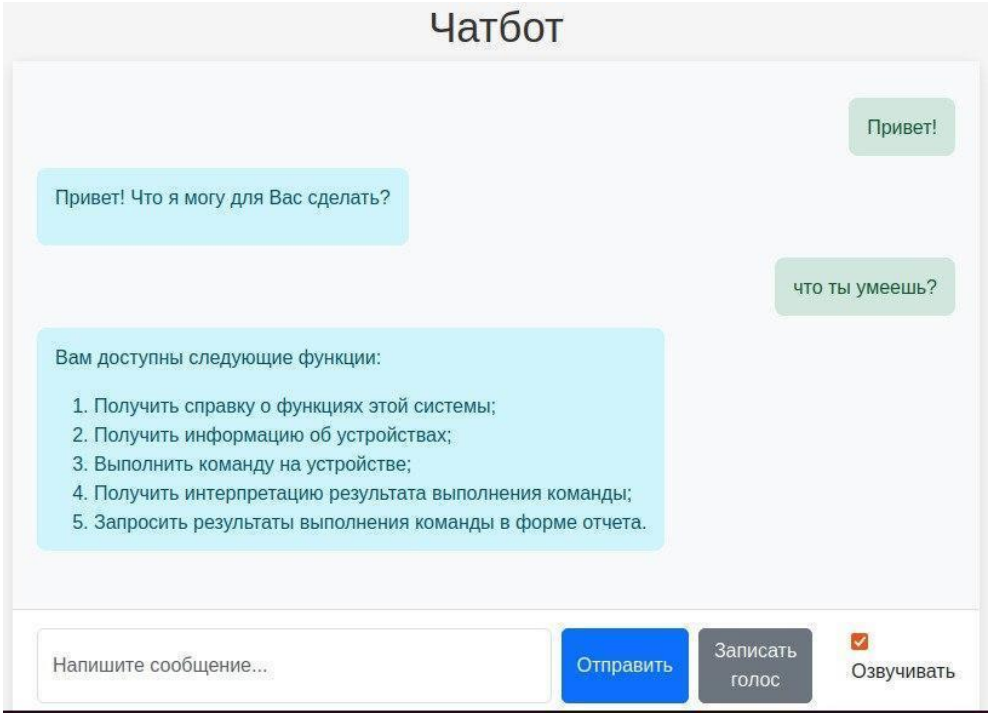

Рис. 3. Экранная форма с примером диалога пользователя

- internal и extenal BFF – паттерны реализации принципа фасада API, агрегирующие взаимодействие с различными контейнерами, в которых реализованы сервисы диалога и работы с голосом, и расширенные такими функциями обеспечения безопасности, как авторизация, шифрование, распределение полномочий, защита от подделки и т. п.;
- модули TTSEngine и STTEngine предназначены для предоставления услуг преобразования текста в голос (озвучивания) и преобразования голоса в текст (транскрибации) и реализованы с использованием паттерна «стратегия», за счет чего имеется возможность подключать внешние сервисы с требуемым качеством обработки звука и текста, а также управлять тарификацией использования этих услуг;

- модуль CommandExecuter предназначен для подключения к агентам управления, которые работают на подконтрольном оборудовании, передачи им команды управления или команды сбора данных. При подключении к агентам CommandExecuter определяет, по какому протоколу необходимо подключиться к агенту и как преобразовать обобщенную команду в вид, допустимый для этого протокола, для этого ему доступна вся необходимая информация, которая может при необходимости дополняться и уточняться;
- модули DeviceStorage, UtilityStorage и ReportStorage предназначены для подключения к хранилищам информации о подконтрольных устройствах, обобщенных командах и шаблонах отчетов, которая используется для реализации потребности пользователя;
- ключевыми элементами архитектуры являются UserDialogInterface и AIAdapter, показанный на рисунке 2 в виде совокупности экземпляров AI подключений, отличающиеся настройками контекста <AI role, prompt>. Оба эти элемента описаны более подробно ниже.

3. Типовой сценарий работы пользователя

Одной из наиболее сложных задач, которую потребовалось решить при создании модуля управления, оказалась задача создания типового сценария работы пользователя и алгоритмизации его с использованием LLM. При разработке этого сценария основной акцент делался на удобство использования, в качестве критерия которого выступала совокупность следующих обобщенных показателей:

- наличие возможности свободного ведения диалога с системой, под которой понималось отсутствие жестких ограничений на реплики, которые произносит пользователь;
- в алгоритмах должна присутствовать защита от ошибок ввода данных, которые может допустить пользователь;
- должна быть предусмотрена мягкая система направления диалога, чтобы пользователь не мог просить систему сделать то, что в ней не предусмотрено.

Представленный критерий является противоречивым и в упрощенном виде может быть определен следующим образом: пользователь должен иметь возможность свободно общаться с системой в ограниченном контексте и получать предсказуемый, адекватный, повторяемый и проверяемый результат на свои запросы.

Удовлетворения данного обобщенного критерия получилось достигнуть за счет одновременного применения двух приемов: тщательной настройки системного промпта, который имеет место в каждой LLM, и применения концепции микродиалогов, на которые искусственно разбивалась беседа с пользователем.

Основной особенностью системного промпта в разработанной системе является постановка задачи LLM на многокритериальный анализ семантики микродиалога, позволяющая управлять направленностью всей беседы, а также вычленять признаки, указывающие на конкретную потребность пользователя.

Основная идея микродиалогов заключается в следующем:

- пользователь вводит любое сообщение;
- система заглядывает на ограниченное количество шагов назад в беседе, выделяя несколько связанных между собой реплик, и получает фрагмент беседы, содержащий целостный смысловой контекст;
- используя специальный алгоритм, система определяет ряд показателей смысла текущей стадии беседы и принимает решение о том, является ли запрос пользователя достаточно понятным, чтобы его выполнить, либо необходимо направить диалог с

помощью наводящих вопросов или просьб, чтобы пользователь более конкретно озвучил свою потребность.

Фактически любое взаимодействие с пользователем сводится к двум шагам: определение того, что именно хочет пользователь, и выполнение его команды. Этот подход оказался удачным за исключения одной проблемы: даже при очень точной настройке системного промпта и минимальных показателях творчества (температуры) получаемый от LLM результат обладает большим разнообразием формы и содержания ответов и с достаточно высокой вероятностью может меняться вплоть до противоположного, а в некоторых ситуациях LLM может «додумывать» то, что не подразумевалось (см. ниже).

По этой причине потребовалось дополнить алгоритмы механизмами внесения поправок в ответ LLM. Этот прием позволил добиться 100% повторяемости выполнения набора тестов, которые использовались в качестве критерия оценки работоспособности программы. Окончательный алгоритм, который заложен в элемент UserDialogInterface показан на рисунке 4.

Рис. 4. Алгоритм определения потребности пользователя

Побочным положительным эффектом, который был получен в результате применения описанных выше приемов, — создание слоя обработки входных данных, который в том числе позволил предотвратить ряд возможных атак злоумышленников, стремящихся сломать АСУ за счет внедрения своих промпт-инъекций.

Таким образом ядро системы разделено на два элемента:

– элемент UserDialogInterface, в котором работают алгоритмы управления направлением ведения беседы и выявления потребности пользователя;

– элемент AIAdapter к особенностям API конкретной LLM, и реализующий интерфейс для унификации взаимодействия с языковыми моделями.

Элемент AIAdapter предназначен для компиляции набора данных запроса и его передачи указанной модели вне зависимости от её провайдера и интерфейса. Структура данного элемента представлена на рисунке 5.

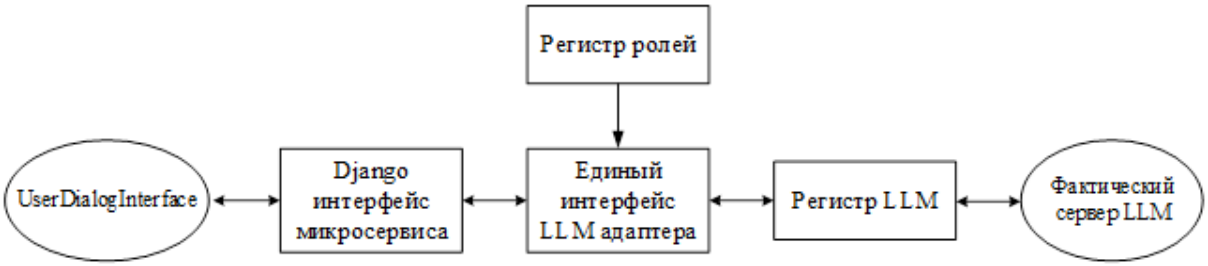

Рис. 5. Структура элемента AIAdapter

Набор данных запроса формируется на основе выбранной роли, определяющей системный промпт, и пользовательского ввода. Конфигурация роли, включённая в «Регистр ролей», также определяет возможность дополнения системного и пользовательского промптов данными из соответствующих загружаемых файлов. Данная функция позволяет для генерации ответа использовать внешнюю информацию, например, справочные данные об устройствах, лог-файлы, результаты выполнения утилит и т.д., что повышает удобство использования всего модуля управления и улучшает пользовательский опыт.

Компонент «Единый интерфейс LLM адаптера» приводит унифицированный набор данных к требуемым API провайдеров форматам и содержит методы обращения к соответствующим LLM. К технической информации, управляемой данным компонентом, относятся API ключи облачных сервисов, адреса локальных сервисов и конфигурации параметров моделей, например, температура и максимальное число токенов ответа.

Ещё одной функцией модуля является приведение результата обращения к общему виду, что также включает в себя получение системной информации о затратах токенов на запрос и ответ на него. Данная отчётность особенно важна при большом количестве пользователей, автоматизации запросов системе управления и манипуляциях относительно большими данными, так как эти ситуации требуют внимательного учёта тарификации работы LLM внешнего провайдера.

Разработанная система управления актуальна в контексте современных и перспективных стандартов подвижной связи ввиду возрастающей виртуализации архитектуры сетей. Виртуализируется ядро сети: в сетях 5G произошёл уход от специализированного оборудования к программам-функциям на стандартном серверном, — так называемой виртуализации сетевых функций NFV. Благодаря этому сама сеть перестаёт быть монолитом, нарезка сети (Network Slicing) позволяет выделить для подсети, реализующей отдельный сценарий, полный набор ресурсов: радиосеть, транспортную сеть и функции ядра. В таких условиях алгоритмы самоорганизации сети безусловно усложняются, возникает необходимость в применении новых методов анализа, контроля, оптимизации и конфигурации.

Ниша LLM заключается в интеллектуальном анализе и возможности прямого взаимодействия с оператором, преобразовании высокоуровневых команд, сценариев, в низкоуровневые процессы, планировании и объяснении действий, рекомендации решений, их самостоятельном выполнении и повышении автономности сети на всех уровнях.

5. Результаты тестирования системы

Эксперименты, связанные с применением полученного опытного образца, показали достаточно большое время отклика, превышающее общепринятые стандарты по удобству использования и требования по показателям качества систем управления связи. Основными источниками, которые могли служить причиной задержек, следующие: архитектура системы, система связи, LLM, стек технологий. Каждая из них рассмотрена отдельно.

Тестирование проводилось на достаточно простом примере, приведенном в таблице 1. Данные формировались, исходя из реального накопленного опыта эксплуатации АСУ разного назначения.

Таблица 1 – Тестовые данные для проверки системы

Идентификатор	**Описание в системе**	**Примечание**
Устройство 3	WiFi-роутер в помещении 2	OpenWRT, стандартный интерфейс управления (CLI)
Устройство 2	WiFi-роутер в помещении 3	OpenWRT, стандартный интерфейс управления (CLI)
Устройство 1	WiFi-роутер в помещении 1	OpenWRT, стандартный интерфейс управления (CLI)

Для оценки показателей производительности системы были произведены замеры времени, затрачиваемого на обработку запроса пользователя как самим модулем управления, так и сервисом LLM. В ходе измерений в качестве пользовательского ввода на вход системы подавались текстовые промпты разной длины для имитации различных сценариев применения. Результаты представлены ниже в абсолютной и относительной форме на рисунках 6 и 7.

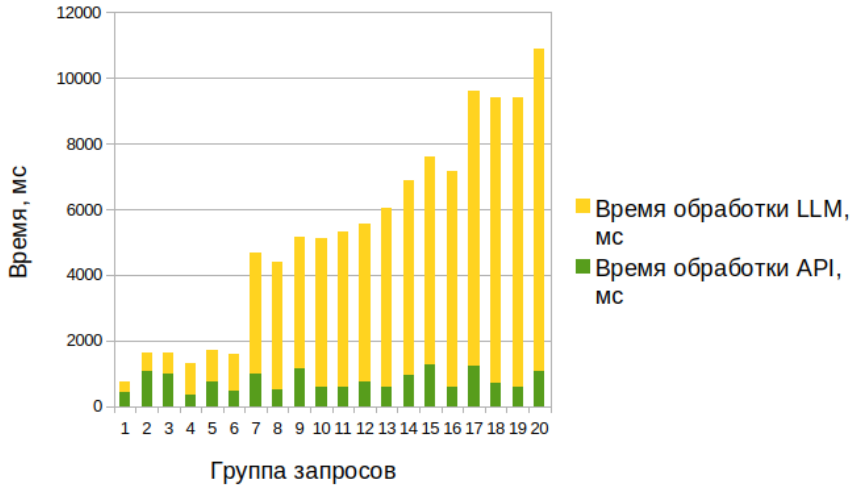

Рис. 6. Абсолютные накладные расходы времени при обработке запроса.

Измерение времени отклика на разных участках цепи передачи данных в рамках модуля управления показало, что в среднем 77% накладных расходов выпадает на время обработки запроса LLM. Расходы на обработку LLM были ниже 72% для нижних 25% наблюдений (первый квартиль) и выше 89,5% — для верхних 25% (четвертый квартиль). Медианное значение доли расходов LLM (второй квартиль) составило 86,3%.

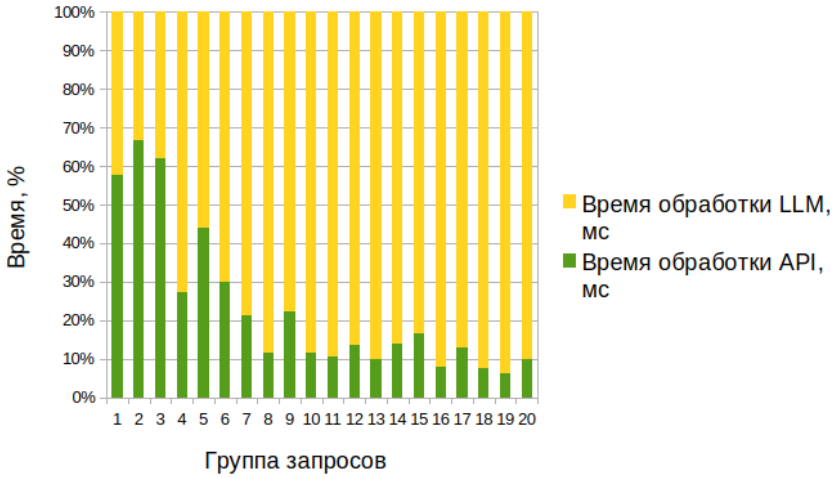

Рис. 7. Относительные накладные расходы времени при обработке запроса.

Результаты измерения производительности на разных участках показывают, что по сравнению с накладными расходами на использование LLM, существенно превышают особенности использованного стека технологий и архитектуры. Высокие показатели сопровождаемости перевесили очень маленький выигрыш по времени выполнения, который может быть получен при смене технологий.

Для оценки адекватности и повторяемости алгоритмов, использующих функции LLM, были использованы различные формулировки заданий. Ниже приведены те, которые выявили основные проблемы.

Задание 1. Есть перечень устройств, приведенный в таблице 1. Выполни команду вычисления загруженности третьего устройства.

Результат: LLM не может определить, для какого устройства нужно выполнить команду и примерно равной вероятностью выдает либо устройство 1, либо устройство 3. В редких случаях (менее 1%) может быть выбрано устройство 2.

При выяснении причин такого поведения оказалось, что в некоторых случаях LLM смотрит на содержание текста, в некоторых на номер строки в перечислении, а иногда она может добавлять к смыслу первой ячейки таблицы смысл второй. Некоторые LLM также плохо работают с числами, представленными не в текстовом виде.

Задание 2. Чтобы выполнить команду, нужно знать устройство, где эта команда должна быть выполнена. Определи, нужно ли узнать, на каком устройстве следует выполнить команду, если пользователь сказал: «Мне нужны сведения об операционной системе».

Результат: примерно в 30% случаях LLM считает, что у пользователя не нужно спрашивать, на каком устройстве нужно выполнить команду, и пытается сама решить, где это нужно сделать.

При выяснении причин такого поведения, LLM дала следующий ответ: «скорее всего, пользователь хочет выполнить команду на локальном устройстве».

Задание 3. В тебе заложены следующие функции: функция 1, функция 2, функция 3. Также ты можешь выполнить следующие команды: команда 1, команда 2, команда 3. Скажи, что ты можешь делать?

Результат: примерно равномерно распределяется вывод одного из трех ответов: перечисление только функций, перечисление только команд, перечисление и того, и другого.

Для анализа этого результата пришлось провести достаточно большое исследование, которое показало, что даже одна и та же LLM может выдавать разный смысл для одного и того же предложения при равных условиях, но в разные моменты времени.

Дополнительно следует отметить, что даже просьбе выдавать результат в определенном формате (например, в виде конкретной фразы, с конкретными знаками препинания, в виде таблице или в формате JSON), LLM решает эту задачу не строго и может выдавать дополнительную информацию или символы, которые усложняют преобразование вывода в точные параметры алгоритмов обработки или команд. Приведенные примеры упрощены по сравнению с реальными исключительно с целью демонстрации некоторых из возникающих проблем и особенностей.

Заключение

При разработке универсального модуля управления, базирующегося на применении технологии LLM выявлены следующие проблемы:

1) Высокая сложность написания системных промптов для LLM, позволяющих добиться высокой точности и повторяемости результата обработки запроса. Необходимость применения алгоритмов корректировки. После многократного переписывания и доведения до совершенства формулировок задания, которое передается LLM, остается достаточно высокая вероятность того, что результат повторной обработки одних и тех же входных данных будет существенно отличаться от предыдущего.
2) Наблюдается достаточно медленный отклик LLM при обработке даже небольшого количества данных.
3) При использовании одного и того же сеанса с LLM для обработки разных запросов возникают искажения в выводах, которые делает LLM. Эта проблема возникает при необходимости оптимизации взаимодействия с LLM с точки зрения экономии денежных средств при подключении платных LLM, а также с необходимости отслеживания истории взаимодействия пользователя с системой и управления структурой, упорядочивания и последующей аналитики входных данных. Эта проблема в том числе усложняет диагностику исключительных ситуаций и ошибок, которые могут возникать в системе.
4) Ведение свободного диалога с пользователем налагает определенные ограничения на обработку пользовательских запросов, которые встраиваются в заложенный в систему контекст. Разговорный естественный язык позволяет формулировать запросы, которые содержат внутренние противоречия, а также менять смысл слов, которые произносит пользователь, что приводит к неправильной интерпретации запроса. Эта проблема создает для пользователя необходимость вводить правила, как разговаривать с системой, что создает для него определенные неудобства и снижает уровень комфорта при работе, заставляет его думать, что противоречит требованию качественного интерфейса.

Важно подчеркнуть, что наблюдаемое поведение отражает актуальное на сегодняшний день состояние развития LLM хочется надеяться, что в ближайшем будущем ситуация поменяется. Однако, на текущий момент эти проблемы являются существенными и создают необходимость сделать следующие выводы относительно системы управления, основанной на LLM:

- эти системы можно применять только для не критически важных объектов, то есть там, где вероятные ошибки пользователя и LLM не могут привести к серьезным последствиям;
- эти системы не эффективны при необходимости управления в режиме реального времени, где любая задержка может привести к серьезным последствиям;
- эти системы могут использовать только высококвалифицированные пользователи, которые способны своевременно и самостоятельно обнаружить ошибку ИИ, а также умеющие грамотно формулировать задачу для LLM и обладающие достаточным терпением в случаях, когда система неправильно выполняет запрошенную команду или неожиданно реагирует.

Отдельную благодарность хочется передать студентам ГУАП Эмилю Гарайшину и Никите Харькову, которые проходили практику в ООО ЛИС и помогали проводить исследование особенностей работы современных LLM.

СПИСОК ЛИТЕРАТУРЫ

1. *Boateng G. O.* A Survey on Large Language Models for Communication, Network, and Service Management: Application Insights, Challenges, and Future Directions / G. O. Boateng [et al.] // IEEE Communications Surveys & Tutorials. – 2026. – Vol. 28. – P. 527-566. – DOI: 10.1109/COMST.2025.3564333.
2. *Letaief K.B.* The Roadmap to 6G: AI Empowered Wireless Networks / K.B. Letaief, W. Chen, Y. Shi, J. Zhang, Y.-J.A. Zhang // IEEE Communications Magazine. – 2019. – Vol. 57, N 8. – P. 84-90. – DOI: 10.1109/MCOM.2019.1900271.
3. *Zhou H.* Large Language Model (LLM) for Telecommunications: A Comprehensive Survey on Principles, Key Techniques, and Opportunities / H. Zhou [et al.] // IEEE Communications Surveys & Tutorials. – 2025. – Vol. 27, N 3. – P. 1955-2005. – DOI: 10.1109/COMST.2024.3465447.
4. *Wang C.* Making Network Configuration Human Friendly / C. Wang, M. Scazzariello, A. Farshin, D. Kostic, M. Chiesa // Networking and Internet Architecture. – 2023. – 8 p.
5. *Dzeparoska K.* LLM-Based Policy Generation for Intent-Based Management of Applications / K. Dzeparoska, J. Lin, A. Tizghadam, A. Leon-Garcia // 2023 19th International Conference on Network and Service Management (CNSM). – Niagara Falls, ON, Canada, 2023. – P. 1-7. – DOI: 10.23919/CNSM59352.2023.10327837.
6. *Sheng Y.* Beam Prediction Based on Large Language Models / Y. Sheng [et al.] // IEEE Wireless Communications Letters. – 2025. – Vol. 14, N 5. – P. 1406-1410. – DOI: 10.1109/LWC.2025.3543567.
7. *Эванс Э.* Предметно-ориентированное проектирование (DDD): структуризация сложных программных систем / Э. Эванс; пер. с англ. – М.: И.Д. Вильямс, 2011. – 448 с.
8. *Бек К.* Экстремальное программирование: разработка через тестирование / К. Бек. – СПб.: Питер, 2024. – 224 с.
9. *Davis A.* Bootstrapping Microservices / A. Davis. – Shelter Island: Manning, 2024. – 464 p.
10. *Аймен Э.* Облачные микросервисы / Э. Аймен; пер. с англ. В.С. Яценкова. – М.: ДМК Пресс, 2024. – 276 с.
11. *Майтак Р. В.* Python, Django, Data Science: учебное пособие / Р. В. Майтак, П.А. Пылов, А.В. Протодьяконов. – Москва; Вологда: Инфра-Инженерия, 2025. – 516 с.
12. *Персиваль Г.* Паттерны разработки на Python: TDD, DDD и событийно-ориентированная архитектура / Г. Персиваль, Б. Грегори. – СПб.: Питер, 2022. – 336 с.
13. *Woldman T.* Hands-On Microservices with Django / T. Woldman. – Bristol: Packt Publishing, 2024. – 278 p.
14. Manning publication. RabbitMQ для профессионалов [Электронный ресурс] – URL: http://onreader.mdl.ru/RabbitMQInDepth/content/index.html (дата обращения: 21.01.2026).
15. *Ричардс М.* Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы / М. Ричардс, Н. Форд. – СПб.: Питер, 2023. – 448 с.
16. *Форд Н.* Современный подход к программной архитектуре: сложные компромиссы / Н. Форд, М. Ричардс, П. Садаладж, Ж. Дехгани. – СПб.: Питер, 2023. – 480 с.
17. *Фаулер М.* Шаблоны корпоративных приложений / М. Фаулер, Д. Райс, М. Фоммел [и др.]; пер. с англ. – М.; СПб.: Диалектика, 2020. – 543 с.

18. *Дэвис К.* Шаблоны проектирования для облачной среды / К. Дэвис; пер. с англ. Д. А. Беликова. – М.: ДМК Пресс, 2020. – 388 с.
19. *Гивакс Д. Д.* Паттерны проектирования API / Д. Д. Гивакс. – СПб.: Питер, 2023. – 512 с.
20. *Феникс Д.* Промт-инжиниринг для GenAI. Паттерны надежных запросов для качественных результатов / Д. Феникс, М. Тейлор. – Астана: Спринт Бук, 2025. – 432 с.
21. *Келен О.* Разработка приложений на базе GPT-4 и ChatGPT / О. Келен, М.-А. Блете. – Астана: Спринт Бук, 2024. – 192 с.

ИНФОРМАЦИЯ ОБ АВТОРАХ

Олимпиев Алексей Александрович
Доцент, канд. техн. наук
Санкт-Петербургский государственный университет аэрокосмического приборостроения
Россия, 190000, Санкт-Петербург, ул. Большая Морская, д.67, лит. А
E-mail: olimpiev@guap.ru

Шеремет Никита Викторович
Аспирант, инженер-конструктор
Санкт-Петербургский государственный университет телекоммуникаций им. М.А. Бонч-Бруевича, ООО «Лаборатория инфокоммуникационных сетей»
Россия, 191186, Санкт-Петербург, набережная реки Мойки, дом 61, литера А
Россия, 197342, Санкт-Петербург, ул. Сердобольская, д.65, лит. А
E-mail: sidwavedata@gmail.com

INFORMATION ABOUT THE AUTHORS

Olimpiev Aleksey Aleksandrovich
PhD. tech. Sciences
Saint-Petersburg State University of Aerospace Instrumentation
67, Bolshaya Morskaia str., Saint-Petersburg, 190000, Russia
E-mail: olimpiev@guap.ru

Sheremet Nikita Viktorovich
Graduate student, design engineer
Bonch-Bruevich Saint Petersburg State University of Telecommunications
Infocommunication Networks Laboratory LLC
61, Moika River Embankment, Saint-Petersburg, 191186, Russia
65, Serdobolskaya str., Saint-Petersburg, 197342, Russia
E-mail: sidwavedata@gmail.com

Дата поступления: 21.05.2026
Дата принятия: 22.05.2026