Научная статья
УДК 004.89
doi:10.37614/2949-1215.2025.16.3.001

ТЕХНОЛОГИЯ ОПЕРАТИВНОЙ РАЗРАБОТКИ ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ СИСТЕМ НА ОСНОВЕ КОММУНИКАЦИОННЫХ ВОЗМОЖНОСТЕЙ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ

Андрей Михайлович Федоров[1], Игорь Олегович Датьев[2], Михаил Олегович Илясов[3], Иван Геннадьевич Вишняков[4], Марк Олегович Базегский[5], Даниил Сергеевич Фигуркин[6], Кристина Дмитриевна Любимова[7]
[1–6]Институт информатики и математического моделирования имени В. А. Путилова Кольского научного центра Российской академии наук, Апатиты, Россия
[7]Филиал федерального государственного автономного образовательного учреждения высшего образования «Мурманский арктический университет» в г. Апатиты, Апатиты, Россия
[1]a.fedorov@ksc.ru, https://orcid.org/0000-0002-2862-7994
[2]i.datyev@ksc.ru, https://orcid.org/0000-0002-8372-8704
[3]m.ilyasov@ksc.ru, https://orcid.org/0009-0006-9065-2396
[4]i.vishnyakov@ksc.ru, https://orcid.org/0009-0003-4938-5693
[5]markbazegskiy@yandex.ru, https://orcid.org/0000-0002-1543-3963
[6]DaniilSF@yandex.ru, https://orcid.org/0000-0002-9743-8222
[7]KLyubimova2002@mail.ru, https://orcid.org/0000-0003-2710-6481

Аннотация

Автоматизация процессов разработки информационных технологий является актуальной задачей. Современные тенденции развития требуют быстрой адаптации решений к новым условиям, однако традиционная практика классической разработки и модернизации прежних решений обладает рядом известных недостатков, ограничивающих производительность и конкурентоспособность проектов. Настоящее исследование направлено на разработку технологии оперативного создания информационных систем путем компонентной интеграции уже имеющихся разработок и интеллектуальных ассистентов и агентов. Основным инструментом такой интеграции предложено использовать стандартный протокол MCP, расширяющий коммуникационные возможности технологий, моделей и средств реализации искусственного интеллекта. Предложенное решение способствует существенному улучшению временных показателей концептуального проектирования и разработки прикладных и исследовательских информационных систем, а также повышает уровень их интеллектуализации и взаимной интеграции. Полученная технология представляет интерес для исследователей и разработчиков в сфере системного анализа, управления и обработки информации.

Ключевые слова:

информационные системы, большие языковые модели, tool calling, протокол MCP, программная разработка, агентные технологии

Благодарности:

исследование выполнено в рамках государственного задания Института информатики и математического моделирования имени В. А. Путилова Кольского научного центра Российской академии наук от Министерства науки и высшего образования РФ, тема научно-исследовательской работы: «Методы и технологии создания интеллектуальных информационных систем для поддержки развития сложных динамических систем с региональной спецификой в условиях неопределённости и риска» (шифр темы FMEZ-2025-0053).

Для цитирования:

Технология оперативной разработки интеллектуальных информационных систем на основе коммуникационных возможностей больших языковых моделей / А. М. Федоров [и др.] // Труды Кольского научного центра РАН. Серия: Технические науки. 2025. Т. 16, № 3. С. 5–21. doi:10.37614/2949-1215.2025.16.3.001.

Original article

TECHNOLOGY FOR DEVELOPING INTELLIGENT INFORMATION SYSTEMS BASED ON THE COMMUNICATIVE CAPABILITIES OF LARGE LANGUAGE MODELS

Andrey M. Fedorov[1], Igor O. Datyev[2], Mikhail O. Ilyasov[3], Ivan G. Vishnyakov[4], Mark O. Bazegskiy[5], Daniil S. Figurkin[6], Christina D. Lyubimova[7]
[1–6]Putilov Institute for Informatics and Mathematical Modeling of the Kola Science Centre of the Russian Academy of Sciences, Apatity, Russia
[7]Apatity branch of Murmansk Arctic University, Apatity, Russia

[1]*a.fedorov@ksc.ru, https://orcid.org/0000-0002-2862-7994*
[2]*i.datyev@ksc.ru, https://orcid.org/0000-0002-8372-8704*
[3]*m.ilyasov@ksc.ru, https://orcid.org/0009-0006-9065-2396*
[4]*i.vishnyakov@ksc.ru, https://orcid.org/0009-0003-4938-5693*
[5]*markbazegskiy@yandex.ru, https://orcid.org/0000-0002-1543-3963*
[6]*DaniilSF@yandex.ru, https://orcid.org/0000-0002-9743-8222*
[7]*KLyubimova2002@mail.ru, https://orcid.org/0000-0003-2710-6481*

Abstract

Automating information technology development processes is a pressing issue. Current development trends require solutions to quickly adapt to new conditions, but traditional practices of classical development and modernization of existing solutions have a number of known shortcomings that limit the productivity and competitiveness of projects. This study aims to develop a technology for the rapid creation of information systems through the component integration of existing developments and intelligent assistants and agents. The proposed primary tool for such integration is the standard MCP protocol, which expands the communication capabilities of artificial intelligence technologies, models, and implementation tools. The proposed solution significantly improves the timeframe for conceptual design and development of applied and research information systems, while also increasing their intellectualization and mutual integration. The resulting technology is of interest to researchers and developers in the fields of systems analysis, management, and information processing.

Keywords:

information systems, large language models, tool calling, MCP protocol, software development, agent technologies

Acknowledgments:

The study was carried out within the framework of the state assignment of the Putilov Institute for Informatics and Mathematical Modeling of the Kola Science Center of the Russian Academy of Sciences from the Ministry of Science and Higher Education of the Russian Federation, research topic: "Methods and technologies for creating intelligent information systems to support the development of complex dynamic systems with regional specificity in conditions of uncertainty and risk" (topic code FMEZ-2025-0053).

For citation:

Fedorov A. M., Datyev I.O., Ilyasov M. O., Vishnyakov I. G., Bazegskiy M. O., Figurkin D. S., Lyubimova C. D. Technology for developing intelligent information systems based on the communicative capabilities of large language models. *Trudy Kol'skogo nauchnogo centra RAN. Seriya: Tekhnicheskie nauki* [Transactions of the Kola Science Centre of RAS. Series: Engineering Sciences], 2025, Vol. 16, No. 3, pp. 5–21. doi:10.37614/2949-1215.2025.16.3.001.

Введение

Современные информационные технологии активно развиваются благодаря появлению новых возможностей в области технологий искусственного интеллекта (ИИ). Большие языковые модели внесли существенные коррективы в процессы разработки и взаимодействия как отдельных элементов пользовательского интерфейса информационных систем, так и их составных компонентов и протоколов внутренней и межсистемной коммуникации. Известно и вполне закономерно то, что немалая доля актуальных разработок представляет собой интеграцию уже существующих и новых решений. В общем случае эти составные компоненты гетерогенны, в некоторых уже используются элементы технологий ИИ. Эта разнородность делает процесс необходимой интеграции трудоемким и требующим неординарной настройки взаимосвязей между отдельными частями такого типа разрабатываемых информационных систем. Таким образом, исследователи и разработчики сталкиваются со сложностями в процессе оперативного создания новых систем и технологий. Этот факт повышает актуальность развития научно-практической базы, направленной на оптимизацию механизмов системной интеграции, применение эффективных интерфейсов взаимодействия компонентов и широкое использование ресурсов ИИ на определенной стандартизированной основе.

В настоящей работе представлена разработанная и опробованная технология оперативного создания прототипов информационных систем путем компонентной интеграции существующих разработок и интеллектуальных агентов с использованием современных инструментов ИИ, таких как языковые модели, и их стандартных средств обеспечения межсистемных коммуникаций на примере протокола MCP.

Значительная часть новых информационных систем и технологий создается путем объединения или расширения существующих решений и добавления в них новых функций, в том числе реализованных на базе технологий ИИ. В большом количестве случаев этот процесс сопряжен с рядом недостатков, среди которых: отсутствие единого унифицированного подхода к построению архитектурных

схем, усложняющее интеграцию компонентов; высокая продолжительность разработки от момента зарождения идеи до готовности прототипа информационной системы; использование для межкомпонентных коммуникаций разнородных технологий и средств, в том числе собственной разработки.

Устранение этих и других недостатков позволит более эффективно развивать современные информационные технологии как на концептуальном, так и на прикладном уровне.

Данное направление исследований приобрело еще большую актуальность в процессе научно-исследовательской работы коллектива авторов по теме «Методы и технологии создания интеллектуальных информационных систем для поддержки развития сложных динамических систем с региональной спецификой в условиях неопределенности и риска» [1].

Развиваемое в этом исследовании тематическое направление фокусированного сбора данных [2] потребовало отдельной комплексной проработки на концептуальном и прикладном уровнях вопросов, заявленных в данной статье.

Более того, вопросы интеграции технологий ИИ в научные и практические разработки в настоящее время стоят очень остро и требуют от их реальных и потенциальных пользователей широкого спектра теоретических знаний, практического умения и опыта использования. Динамика развития в этой сфере очень высокая. Например, за время подготовки данной статьи из существовавших независимо друг от друга двух коммуникационных протоколов взаимодействия ИИ-агентов остался только один, что внесло свои коррективы в список готовых к публикации результатов.

Благодаря предложенному в данной работе компонентному интеграционному подходу и использованию коммуникационных свойств больших языковых моделей время на разработку ИС уменьшается, эффект от повторного использования компонентов повышается, затраты на замену технологического стека снижаются.

Научная новизна предложенного решения заключается в комплексном подходе к созданию информационных систем, предусматривающем сочетание традиционного опыта разработки и передовых методов интеграции технологий ИИ, а именно больших языковых моделей, протокола MCP, механизма tool calling и техник формирования контекстных инструкций (промпт-инжиниринга). Разработанная технология является развитием практики применения системного анализа и обработки информации для решения проблемы оперативного создания прототипов информационных систем с использованием интеллектуальных агентов и языковых моделей. Методы и средства, применяемые в исследовании, соответствуют современным требованиям системного анализа, теории принятия решений и искусственного интеллекта.

В данной работе решается задача разработки технологии оперативного построения прототипов информационных систем с использованием элементов и технологий ИИ. Формально задача поставлена следующим образом: дана исходная информационная система (ИС), в которую необходимо интегрировать элементы технологии ИИ. В качестве ИИ используются большие языковые модели, технические характеристики которых позволяют им взаимодействовать с внешними программно-алгоритмическими модулям через механизм tool calling.

Необходимо разработать последовательность действий (план, схему, архитектуру) оперативной интеграции в заданную ИС элементов технологии ИИ и определить перечень подходящих и доступных элементов. Важным требованием к планированию разработки является использование стандартных или уже разработанных функциональных компонентов (ФК), готовых к интеграции.

Для достижения указанных характеристик предлагается использовать способ интеграции в языковые модели с помощью универсального протокола MCP внешних функций, называемых функциональными инструментами (ФИ). Всё перечисленное выше является составной частью разработанной технологии.

Протокол MCP

Протокол контекста модели (MCP) — это развивающийся открытый стандарт, разработанный для обеспечения бесшовной, безопасной и динамической интеграции между моделями ИИ, особенно большими языковыми моделями (LLM) [3; 4], и внешними инструментами, источниками данных и различными средами. MCP был создан компанией Anthropic для стандартизации того, как модели ИИ взаимодействуют с внешними разнородными ресурсами, обеспечивая пополнение возможностей

модели в реальном времени с учетом контекста [5; 6]. MCP фокусируется исключительно на протоколе обмена контекстом — он не определяет, как приложения ИИ используют LLM или управляют предоставленным контекстом. MCP решает давние проблемы взаимодействия систем ИИ, управления контекстом и безопасного расширения возможностей моделей. Предназначен для построения мультиагентных систем для решения задач в различных предметных областях как в научных исследованиях, так и в реальном секторе экономики [7; 8].

Архитектура MCP

MCP использует архитектуру [9] клиент-сервер, где хост MCP — ИИ-приложение, устанавливает соединения с одним или несколькими серверами MCP. Хост MCP создает один клиент MCP для каждого сервера MCP. Каждый клиент MCP поддерживает выделенное индивидуальное соединение с соответствующим сервером MCP.

Ключевыми компонентами архитектуры MCP являются: хост MCP — приложение ИИ, которое координирует и управляет одним или несколькими клиентами MCP; клиент MCP — компонент, который поддерживает соединение с сервером MCP и получает контекст от сервера MCP для использования хостом MCP; сервер MCP — программа, предоставляющая контекст клиентам MCP. MCP-сервер относится к программе, которая обслуживает контекстные данные независимо от места ее работы. MCP-серверы могут работать локально или удаленно.

Два слоя протокола MCP: 1) уровень данных: определяет протокол на основе JSON-RPC для клиент-серверного взаимодействия, включая управление жизненным циклом и основные примитивы, такие как инструменты, ресурсы, запросы и уведомления; 2) транспортный уровень: определяет механизмы связи и каналы, обеспечивающие обмен данными между клиентами и серверами, включая установление транспортного соединения, формирование сообщений и авторизацию.

Уровень данных реализует протокол обмена на основе JSON-RPC 2.0 [10], который определяет структуру и семантику сообщений. Этот уровень включает в себя: 1) управление жизненным циклом: осуществляет инициализацию соединения, согласование возможностей и завершение соединения между клиентами и серверами; 2) серверные функции: позволяет серверам предоставлять основные функции, включая инструменты для действий ИИ, ресурсы для контекстных данных и запросы шаблонов взаимодействия с клиентом и от него; 3) клиентские функции: позволяет серверам запрашивать у клиента выборку из LLM-хоста, получать ввод данных от пользователя и регистрировать сообщения для клиента; 4) служебные функции: поддерживает дополнительные возможности, такие как уведомления об обновлениях в режиме реального времени и отслеживание хода выполнения длительных операций.

Транспортный уровень управляет каналами связи и аутентификацией между клиентами и серверами. Он отвечает за установление соединения, формирование сообщений и защищенную связь между участниками MCP.

MCP поддерживает два транспортных механизма: *Stdio transport*: использует стандартные потоки ввода/вывода для прямого взаимодействия между локальными процессами на одной машине, обеспечивая оптимальную производительность без сетевых издержек; *Streamable HTTP transport*: использует HTTP POST для сообщений клиент-сервер с опциональными событиями, отправленными сервером, для потоковой передачи. Этот транспорт обеспечивает взаимодействие с удаленным сервером и поддерживает стандартные методы HTTP-аутентификации, включая токены-носители, ключи API и настраиваемые заголовки. MCP рекомендует использовать OAuth для получения токенов аутентификации.

Транспортный уровень абстрагирует информацию о взаимодействии от уровня протокола, обеспечивая единый формат сообщений JSON-RPC 2.0 для всех транспортных механизмов.

Протокол уровня данных. Основной частью MCP является определение схемы и семантики между клиентами и серверами MCP. Именно уровень данных, в частности набор примитивов, определяет способы обмена контекстом между серверами MCP и клиентами MCP.

MCP использует JSON-RPC 2.0 в качестве базового протокола RPC. Клиент и серверы отправляют запросы, ответы и уведомления друг другу.

Управление жизненным циклом. Цель управления жизненным циклом — согласовать функции и операции, поддерживаемые клиентом или сервером, такие как инструменты, ресурсы или подсказки, поддерживаемые как клиентом, так и сервером.

Примитивы

Примитивы MCP — важнейший концепт MCP. Примитивы определяют типы контекстной информации, которой можно поделиться с приложениями ИИ, и диапазон доступных действий. MCP определяет три основных примитива, которые могут предоставлять серверы: 1) *инструменты (Tools)* — исполняемые функции, которые приложения ИИ могут вызывать для выполнения действий (например, файловых операций, вызовов API, запросов к базе данных); 2) *ресурсы (Resources)* — источники данных, предоставляющие контекстную информацию приложениям ИИ (например, содержимое файлов, записи базы данных, ответы API); 3) *запросы к языковой модели (Promts, промпты)* — многоразовые шаблоны, помогающие структурировать взаимодействие с языковыми моделями (например, системные промпты, «few-shot» примеры).

Каждому типу примитива соответствуют методы обнаружения (`*/list`), извлечения (`*/get`) и (в некоторых случаях) выполнения (`tools/call`). Клиенты MCP будут использовать методы `*/list` для поиска доступных примитивов. Например, клиент может сначала составить список всех доступных инструментов (`tools/list`), а затем выполнить их. Такая структура позволяет создавать динамические списки.

MCP также определяет примитивы, которые могут предоставлять клиенты. Эти примитивы позволяют разработчикам серверов MCP создавать более сложные взаимодействия.

Выборка (Sampling) позволяет серверам запрашивать дополнения языковой модели из клиентского приложения ИИ. Это полезно, когда авторы серверов хотят получить доступ к языковой модели, но при этом сохранить независимость от модели и не включать SDK языковой модели в свой сервер MCP. Они могут использовать метод `sampling/complete` для запроса дополнения языковой модели из клиентского приложения ИИ.

Извлечение (Elicitation) позволяет серверам запрашивать дополнительную информацию у пользователей. Это полезно, когда авторы серверов хотят получить дополнительную информацию от пользователя или запросить подтверждение действия. Они могут использовать метод `elicitation/request` для запроса дополнительной информации у пользователя.

Ведение журнала (Logging) позволяет серверам отправлять сообщения журнала клиентам для отладки и мониторинга.

Уведомления (Notifications)

Протокол поддерживает уведомления в режиме реального времени для обеспечения динамических обновлений между серверами и клиентами. Так, при изменении доступных инструментов сервера, например при появлении новых функций или изменении существующих инструментов, сервер может отправлять уведомления об обновлениях инструментов, чтобы информировать подключенных клиентов об этих изменениях. Уведомления отправляются в виде уведомлений JSON-RPC 2.0 (без ожидания ответа) и позволяют серверам MCP предоставлять обновления подключенным клиентам в режиме реального времени.

Инициализация

Процесс инициализации является ключевой частью управления жизненным циклом MCP и служит нескольким важным целям:

1. Согласование версии протокола. Поле `protocolVersion` (например, "2025-06-18") гарантирует, что клиент и сервер используют совместимые версии протокола. Это предотвращает ошибки связи, которые могут возникнуть при попытке взаимодействия разных версий. Если взаимосовместимая версия не согласована, соединение следует разорвать.

2. Определение возможностей. Объект `capabilities` позволяет каждой стороне указать поддерживаемые функции, включая [примитивы](#primitives), которые она может обрабатывать (инструменты, ресурсы, подсказки), а также указать, поддерживает ли она такие функции, как [уведомления](#notifications). Это обеспечивает эффективное взаимодействие и позволяет избегать неподдерживаемых операций.

3. Обмен идентификацией. Объекты `clientInfo` и `serverInfo` предоставляют информацию об идентификации и версиях для отладки и обеспечения совместимости.

Во время инициализации менеджер клиентов MCP приложения ИИ устанавливает соединения с настроенными серверами и сохраняет их возможности для последующего использования. Приложение использует эту информацию, чтобы определить, какие серверы могут предоставлять определенные типы функций (инструменты, ресурсы, подсказки) и поддерживают ли они обновления в реальном времени.

Обнаружение инструментов

Запрос «tools/list» прост и не содержит параметров.

Ответ содержит массив «инструменты», который предоставляет полные метаданные о каждом доступном инструменте. Эта структура на основе массива позволяет серверам одновременно предоставлять несколько инструментов, сохраняя при этом четкие границы между различными функциями.

Приложение AI извлекает доступные инструменты со всех подключенных серверов MCP и объединяет их в единый реестр инструментов, к которому имеет доступ языковая модель. Это позволяет LLM получать список доступных действий и автоматически генерировать соответствующие вызовы инструментов во время диалогов.

Ответ на выполнение инструмента

Ответ демонстрирует гибкую систему контента MCP:

1. Массив `content`: ответы инструмента возвращают массив объектов контента, что позволяет создавать ответы в различных форматах (текст, изображения, ресурсы и т. д.).

2. Типы контента: каждый объект контента имеет поле `type`. В этом примере `"type": "text"` обозначает простой текстовый контент, но MCP поддерживает различные типы контента для различных вариантов использования.

3. Структурированный вывод: ответ предоставляет полезную информацию, которую приложение ИИ может использовать в качестве контекста для взаимодействия с языковой моделью.

Этот шаблон выполнения позволяет приложениям ИИ динамически вызывать функции сервера и получать структурированные ответы, которые можно интегрировать в диалоги с языковыми моделями.

Когда языковая модель решает использовать инструмент во время диалога, приложение ИИ перехватывает вызов инструмента, направляет его на соответствующий сервер MCP, выполняет его и возвращает результаты обратно в LLM в рамках потока диалога. Это позволяет LLM получать доступ к данным в режиме реального времени и выполнять действия во внешнем мире.

MCP поддерживает уведомления в режиме реального времени, позволяющие серверам информировать клиентов об изменениях без явного запроса. Это демонстрирует систему уведомлений — ключевую функцию, обеспечивающую синхронизацию и быстрое реагирование подключений MCP.

Уведомления об изменении списка инструментов

При изменении доступных инструментов сервера, например при появлении новых функций, изменении существующих инструментов или их временной недоступности, сервер может заблаговременно уведомить подключенных клиентов.

Ключевые особенности уведомлений MCP:

1. Ответ не требуется: в уведомлении отсутствует поле `id`. Это соответствует семантике уведомлений JSON-RPC 2.0, где ответ не ожидается и не отправляется.

2. На основе возможностей: Это уведомление отправляется только серверами, которые объявили "listChanged": true в своих возможностях инструментов во время инициализации (как показано на шаге 1).

3. Управляемое событиями: сервер решает, когда отправлять уведомления, основываясь на изменениях внутреннего состояния, что делает соединения MCP динамичными и отзывчивыми.

Получив уведомление, клиент обычно реагирует, запрашивая обновленный список инструментов. Это создает цикл обновления, который поддерживает актуальность информации клиента о доступных инструментах:

Эта система уведомлений критически важна по нескольким причинам: 1) динамические среды: инструменты могут появляться и исчезать в зависимости от состояния сервера, внешних зависимостей или прав пользователя; 2) эффективность: клиентам не нужно спрашивать об изменениях, они

уведомляются об обновлениях; 3) консистентность: обеспечивает клиентам постоянную точную информацию о доступных возможностях сервера; 4) совместная работа в реальном времени: обеспечивает адаптивность ИИ-приложений к изменяющимся контекстам.

Этот шаблон уведомлений распространяется не только на инструменты, но и на другие примитивы MCP, обеспечивая комплексную синхронизацию в реальном времени между клиентами и серверами. Когда ИИ-приложение получает уведомление об изменении инструментов, оно обновляет свой реестр инструментов и доступные возможности LLM. Это гарантирует доступ к актуальному набору инструментов, а LLM может динамически адаптироваться к новым функциям по мере их появления.

Вызов инструментов (tool calling)

В последние годы большие языковые модели (LLM) стали широко использоваться в различных приложениях, включая генерацию дополненного поиска (RAG), системы рекомендаций и корпоративные среды. Одной из ключевых возможностей LLM является вызов инструментов (tool calling), который позволяет моделям взаимодействовать с внешними системами и расширять свои возможности.

Tool calling — это механизм, благодаря которому большие языковые модели (LLM) взаимодействуют с внешними системами и инструментами, а также выполняют задачи, в которых требуется доступ к внешним ресурсам, таким как базы данных, API и другие сервисы. Tool calling обычно реализуется через интерфейс, который позволяет моделям отправлять запросы внешним системам и получать ответы в формате, который может быть обработан моделью.

К преимуществам tool calling следует отнести: 1) расширение возможностей LLM: tool calling позволяет моделям выполнять задачи, в которых требуется доступ к внешним ресурсам, таким как базы данных и API; 2) улучшение производительности: tool calling может значительно улучшить производительность моделей, так как они могут использовать внешние системы для выполнения задач, которые требуют больших вычислительных ресурсов; 3) повышение точности: tool calling позволяет моделям получать более точные ответы на запросы, так как они могут использовать внешние системы для проверки и уточнения информации.

Однако при использовании tool calling необходимо учитывать доступность, надежность и безопасность внешних систем, к которым будут происходить обращения и которые будут возвращать результаты вызова функций.

Современные исследования и разработки в области tool calling направлены в том числе на решение вопросов, связанных с интеграцией, производительностью и безопасностью. Например, в [11] авторы предлагают онлайн-оптимизированный RAG, который непрерывно адаптирует векторы (эмбеддинги) извлекаемых данных из реальных взаимодействий, используя минимальную обратную связь. В работе [12] представлен CoreThink Agentic Reasoner — фреймворк, дополняющий LLM-модели легковесным слоем символьных рассуждений для структурной декомпозиции и адаптивного управления инструментами. Авторы [13] предлагают инструменты естественного языка (NLT), которые заменяют программный вызов инструментов JSON на вывод на естественном языке. В работе [14] исследуется вызов инструментов для арабского языка и рассмотрены три ключевых вопроса: необходимость наличия данных на арабском языке, влияние настройки инструкций общего назначения на производительность вызова инструментов и ценность тонкой настройки для конкретных высокоприоритетных инструментов. Авторы [15] изучают вызов инструментов в регулируемых корпоративных средах, таких как финтех. В [16] предлагают унифицированный подход к интеграции инструментов, который абстрагирует различия протоколов и оптимизирует производительность выполнения.

Таким образом, tool calling является критически важной возможностью для больших языковых моделей, позволяющей им взаимодействовать с внешними системами и расширять свои возможности. Механизм tool calling стал активно развиваться в последние годы усилиями исследователей и компаний OpenAI, Google и других. Например, OpenAI активно использует tool calling в своих моделях, таких как GPT-4, для взаимодействия с внешними API и сервисами.

Tool calling существенно повышает объяснимость и детерминированность полученных с помощью технологий ИИ результатов, поскольку основан на вызове «классических» функций. Основная сложность — это вероятностный характер процедуры выбора языковой моделью функций, соответствующих поставленной задаче. Обычно эта сложность преодолевается за счет предварительного

создания полных стандартизированных описаний вызываемых функций и правильно подобранных контекстных инструкций (промптов). Более подробно особенности использования tool calling представлены в работах [11; 17].

Технология оперативного создания прототипов информационных систем на основе ИИ

В соответствии с поставленной во введении задачей разработана технология, используя которую можно оперативно создавать прототипы интеллектуальных ИС. В рамках данной разработки представлен концепт архитектуры интеллектуальной ИС на основе большой языковой модели, определен набор средств функционального расширения и наполнения ее возможностей, а также алгоритм их совместной интеграции в качестве основного связующего звена. Алгоритмическая блок-схема решения поставленной задачи показана на рис. 1 в виде стандартной UML-диаграммы активности.

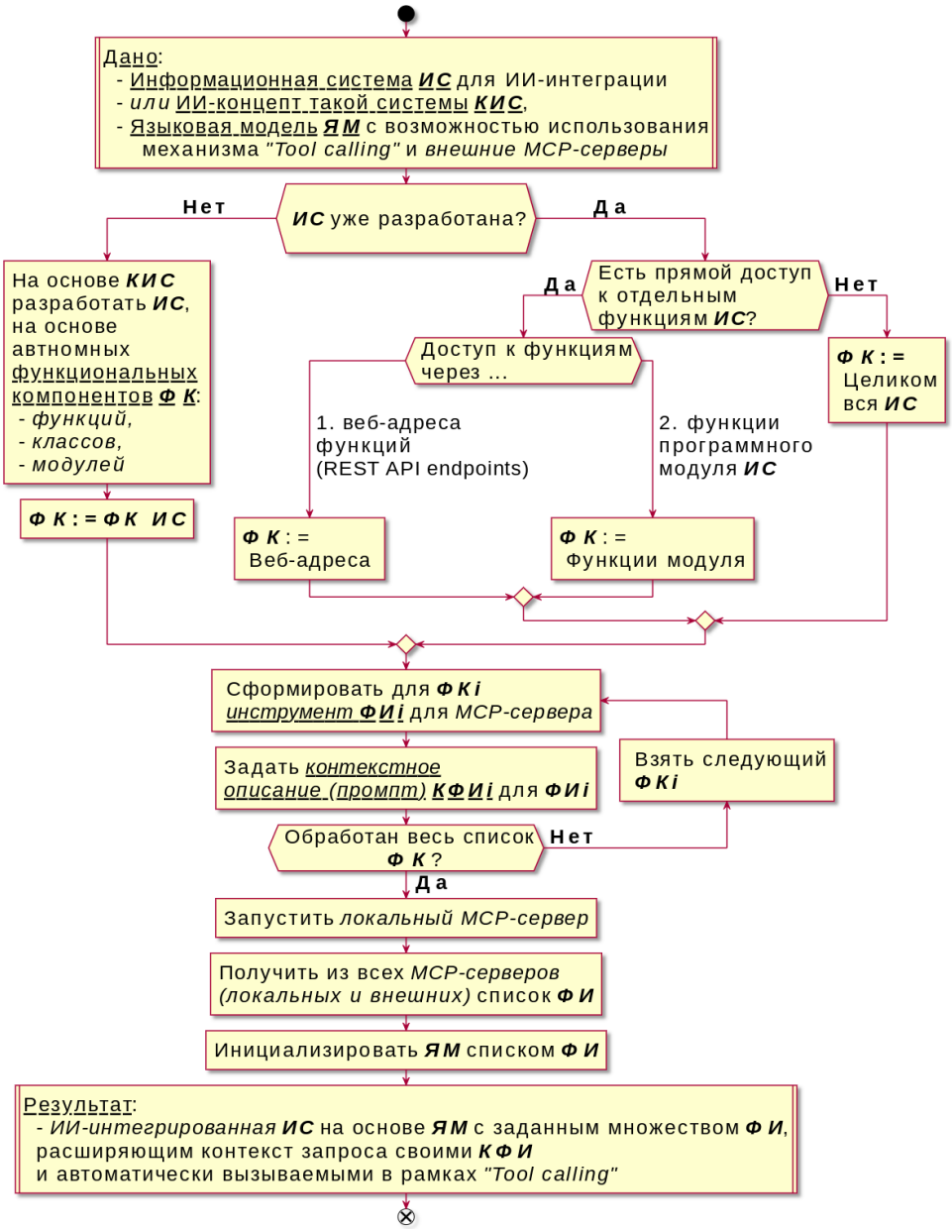

Рис. 1. Технология оперативного создания прототипов ИС на основе ИИ

Таким образом, оперативно создаваемые прототипы интеллектуальных систем архитектурно являются модульными агентами на основе больших языковых моделей, к которым по стандартному протоколу MCP подключаются внешние функциональные ресурсы — инструменты. Само интеллектуальное ядро такого агента выполняет связующую коммуникационную роль, обеспечивающую универсальный человеко-машинный интерфейс на основе естественного языка. Преимущество такого подхода в том, что большое количество типовых MCP-инструментов можно выбрать из специализированных репозиториев (например, fastmcp.me). Недостающие в работе специфические модули необходимо разработать самостоятельно. Такую разработку можно целенаправленно вести с учетом спецификации MCP. Однако, если есть готовые (унаследованные) модули, построенные без учета требований MCP, их можно интегрировать в предложенную архитектуру на уровне конфигурирования локального MCP-сервера. В данной работе учтены подобные варианты, и на схеме представлены отдельные решения как для разной степени открытости исходного кода консольных (скриптовых) модулей, так и для веб-разработок, реализованных на базе архитектуры REST API, где доступ к функциям производится по веб-адресам (маршрутам).

Практическая реализация технологии оперативного создания прототипов ИС на основе ИИ

В данном разделе представлен пример реализации технологии оперативного создания прототипов ИС на основе ИИ применительно к разработанной ранее ИС автоматизированного мониторинга чатов «ВКонтакте». Функциями данной ИС являются: 1) авторизация в ВК Мессенджер через сохраненный пользовательский профиль барузера Firefox; 2) открытие чатов по их названию через поисковую систему «ВК»; 3) отправка текстовых сообщений в активный чат; 4) создание тестовых сообщений для анализа структуры DOM; 5) прокрутка (динамическая подгрузка) истории сообщений; 6) извлечение контента сообщений (парсинг, сбор данных).

Эффективность и результативность работы основной функции системы (п. 6) обеспечивается остальными вспомогательными функциями. Такая функциональная специфика на практике отражает суть исследуемого и развиваемого коллективом авторов подхода фокусированного сбора данных.

На рисунке 2 в виде UML-диаграммы вариантов использования представлены способы реализации имеющейся ИС до и после преобразования в интеллектуальную систему. Набор функций исходной системы остался прежним, а пользовательский интерфейс реализован на основе запросов в свободной форме на естественном языке посредством создания интеллектуального агента.

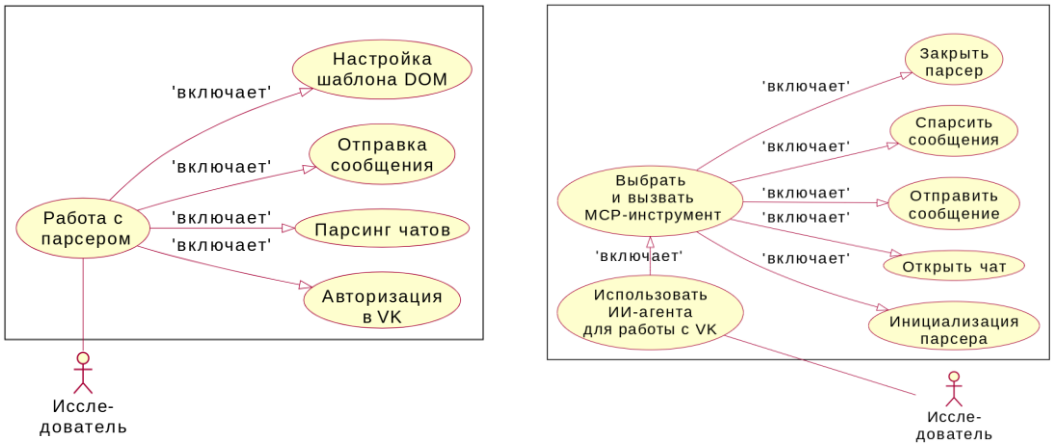

Рис. 2. Варианты действий пользователя ИС, построенной без использования ИИ (слева) и варианты действий пользователя ИС, построенной на базе ИИ (справа)

На рисунке 3 в виде UML-диаграммы компонентов представлена структура интеллектуальной ИС: ИИ-агента и MCP-сервера с реализованными в нем функциональными инструментами. Применяемая технология интеграции позволяет оставить полностью неизменными структурные элементы существующей (до преобразования) ИС. С помощью сервера создается оболочка вокруг работающих функций ИС, что делает их доступными через стандартизированный программный интерфейс.

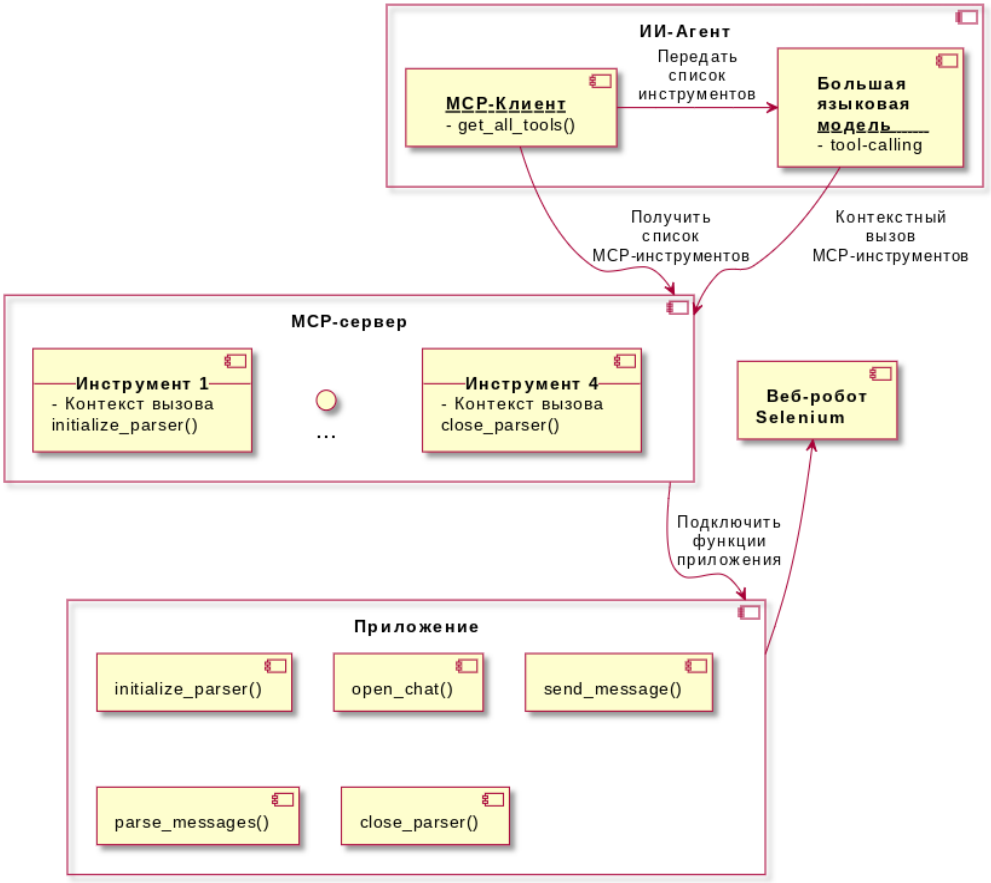

Рис. 3. Компоненты интеллектуальной ИС, построенной на основе технологий ИИ

В программной реализации MCP-сервер использует фреймворк FastMCP для создания набора инструментов. Сервер хранит экземпляр ИС (VKMessageParser) в глобальной переменной и инициализирует его при первом обращении через специальную функцию-обертку (wrapper). Создано пять инструментов-оберток, каждый из которых соответствует конкретному методу исходной ИС, в том числе инициализация сессии, открытие чатов, отправка сообщений, парсинг истории сообщений и завершение сессии. Каждый инструмент вызывает соответствующий метод ИС и возвращает полученный результат. Сервер использует порт 8000 и HTTP-протокол для обмена данными.

Таким образом, MCP-сервер предоставляет языковой модели функции для работы с мессенджером «ВКонтакте», реализованные в ИС ранее (до преобразования). При запуске интеллектуальная ИС устанавливает соединение с сервером и получает перечень доступных инструментов. Каждый инструмент соответствует определенной функции исходной ИС. Например, инструмент для открытия чатов представлен на листинге 1. Декоратор @mcp.tool указывает, что эта функция определяется как MCP-инструмент. Аналогично оформлены и другие инструменты-функции: initialize_parser(), send_message(), parse_messages(), close_parser().

```
@mcp.tool
def open_chat(chat_name: str) -> str:
    """Открыть чат в VK по его названию"""
    parser = VKMessageParser()
    success = parser.open_chat(chat_name)
return f"Чат '{chat_name}' успешно открыт" if success else f"Не удалось открыть чат
'{chat_name}'"
```

Листинг 1. Функция-инструмент для открытия чатов

Часть ИС, выступающая в роли интеллектуального агента, реализована как LangGraph-приложение [18] с архитектурой ReAct (Reasoning + Acting). В нем создается MCP-клиент с использованием интерфейса MultiServerMCPClient. Таким образом, агент запускает MCP-сервер как дочерний процесс и общается с ним через стандартные потоки ввода-вывода (stdio). Пример конфигурации представлен на листинге 2.

```
client = MultiServerMCPClient(
  {
    "vk_server": {
      "transport": "stdio",
      "command": sys.executable,
      "args": [str(Path(__file__).parent / "mcp_vk.py")],
    }
  }
)
```

Листинг 2. Инициализация клиента описанием целевого MCP-сервера

Соединение агента с большой языковой моделью происходит с помощью конфигурации, представленной на листинге 3. Языковая модель развернута на локальном сервере ИИММ. При необходимости можно подключать языковые модели, размещенные как на локальных, так и на облачных ресурсах. Для более детерминированных ответов модели в настройках подключения к модели параметр температуры (temperature) установлен в нулевое значение. Интеграция языковой модели и полученных из MCP-сервера инструментов происходит в процессе формирования специализированного реактивного объекта.

```
model = ChatOpenAI(
  model="reedmayhew/claude-3.7-sonnet-reasoning-gemma3-12B",
  base_url="http://адрес-сервера-иимм:11434/v1/",
  api_key="none",
  temperature=0,
  max_tokens=4096)
tools = await client.get_tools()
agent = create_react_agent(model, tools)
```

Листинг 3. Инициализация подключения агента к большой языковой модели

На рисунке 4 в виде UML-диаграммы последовательности представлено взаимодействие компонентов ИС в процессе отправки запроса пользователя агенту на открытие чата и парсинг сообщений. Агент, получив запрос, обращается к большой языковой модели для выбора подходящих инструментов. Языковая модель анализирует запрос и возвращает агенту информацию о необходимых инструментах. В данном случае это «open_chat» и «parse_messages». Этот вызов передается через «MultiServerMCPClient» к MCP-серверу, который выполняет соответствующую функцию, ранее представленную на листинге 1. В свою очередь, MCP-сервер вызывает ранее реализованные методы «open_chat» и «parse_messages», которые с помощью модуля веб-роботизации Selenium выполняют операции в веб-версии мессенджера «ВКонтакте».

На рисунке 5 в виде UML-диаграммы развертывания представлена ИС автоматизированного мониторинга чатов «ВКонтакте» с использованием интеллектуального агента.

ИС разворачивается в двух изолированных докер-контейнерах. Контейнер «mcp-vk-server» содержит MCP-сервер, предоставляющий инструменты парсинга, а также исходную ИС, в которой реализованы функции для работы веб-робота в браузере с помощью библиотеки Selenium.

В контейнере «mcp-client» реализован интеллектуальный агент на базе фреймворка «LangChain», который отвечает за планирование действий, обработку диалога, и в нем же располагается MCP-клиент для подключения к MCP-серверу, чтобы получить список доступных агенту инструментов.

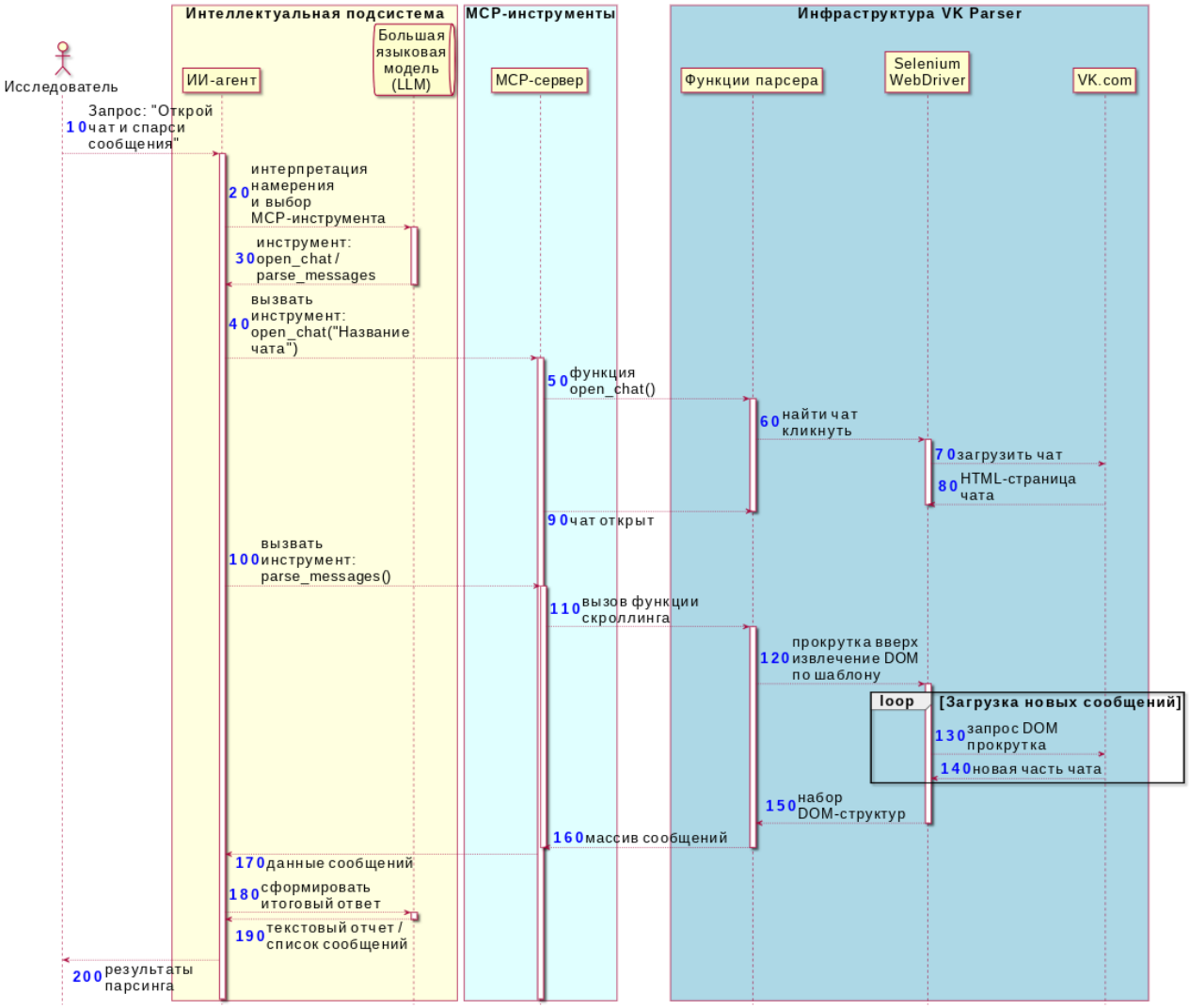

Рис. 4. Последовательность взаимодействий компонентов интеллектуальной ИС

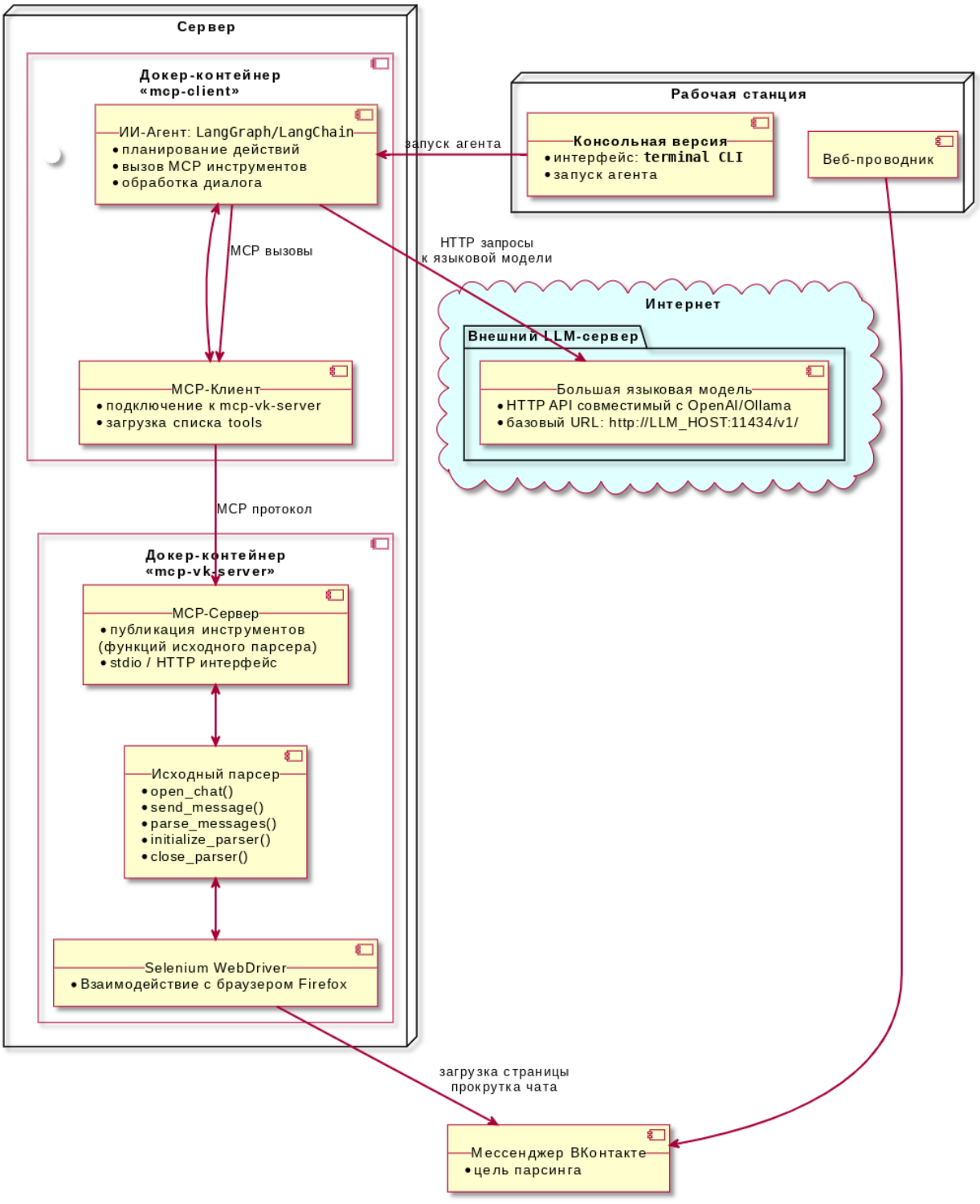

Рис. 5. Архитектура ИС автоматизированного мониторинга чатов «ВКонтакте» на основе ИИ

ИС взаимодействует с языковой моделью, развернутой на локальном сервере ИИММ, для обработки естественного языка, а также с веб-версией мессенджера «ВКонтакте». Пользователь взаимодействует с системой через консольный интерфейс на рабочей станции, формулируя запросы на естественном языке.

Обсуждение

Для оценки работоспособности и эффективности прототипа ИИ-интегрированной системы проведена серия экспериментов. Экспертно оценена способность агента понимать и правильно интерпретировать пользовательские запросы на естественном языке. В данном исследовании, в отличие от классического общераспространенного варианта использования, результаты генеративной работы языковой модели реализуют функции управления. Именно проявление этих коммуникационных возможностей ИИ стало предметом данной исследовательской работы. Максимально корректное восприятие языковой моделью пользовательских запросов обеспечивает правильный выбор и активацию подключенных к ней MCP-инструментов. С учетом отмеченной специфики, первоочередными стали вопросы обеспечения устойчивости используемых моделей к галлюцинациям и минимизация ошибочных действий.

Выбранная и развернутая на локальном сервере ИИММ языковая модель (claude-3.7-sonnet-reasoning-gemma3-12B) в целом справилась с поставленными на данном этапе тестовыми задачами и продемонстрировала исчерпывающие для этого возможности. Тестовая замена указанной версии модели на более простые варианты значительно снижала эффективность работы ИС. Однако, в силу нетривиальности выбора адекватных метрик, соответствующие формальные сравнительные оценки проведенных экспериментов авторы планируют представить в следующих своих работах.

Таким образом, по результатам проведенных работ можно отметить, что предложенная технология обладает рядом достоинств: 1) принципиальная возможность интеллектуализации практически любых проектируемых и разрабатываемых ИС; 2) доступные методы интеграции и унификации подходов к разработке прототипов информационных систем; 3) возможность оперативного внесения изменений и расширения функциональности системы — масштабируемость; 4) снижение времени на проектирование и тестирование решений; 5) использование уже имеющихся программных компонентов — повторное использование; 6) возможность предварительной оценки функционального наполнения проекта интеллектуальной ИС и его принципиальной работы без необходимости создания больших объемов программного кода.

Разработана технология, позволяющая создавать прототипы интеллектуальных информационных систем, в которых пользовательский интерфейс реализован на естественном языке и позволяет гибко и динамично формировать функциональное наполнение системы. Управление всей системой и отдельными ее компонентами производится с помощью контекстных инструкций (промптов).

В общем случае разработанные на основе представленной технологии интеллектуальные системы и их компоненты могут быть преобразованы в автономных агентов. Известны методы построения из таких элементов целых мультиагентных систем, в которых взаимодействие также строится на основе коммуникационных возможностей больших языковых моделей. В частности, к таким методам относятся протоколы A2A и ANP. Результаты по этому направлению исследований будут представлены авторами в будущих публикациях.

К недостаткам предложенного решения относятся конфабуляции (галлюцинации) и недетерминированность результатов работы больших языковых моделей. Это негативное свойство моделей переносится и на основной рабочий механизм функциональной интеграции (tool calling), на базе которого реализованы те самые коммуникационные возможности больших языковых моделей.

В данном случае в качестве возможных решений рассматриваются: 1) формирование точных описаний MCP-инструментов; 2) подбор хороших контекстных инструкций (промтов); 3) организация промежуточных контрольных проверок и метрик правильности ответов; 4) использование актуальных и максимально производительных вариантов больших языковых моделей.

В целом потенциал и возможности разработанной технологии во многом перекрывают указанные недостатки, которые на текущем этапе можно считать временными и вполне разрешимыми.

Заключение

Представленные в данной работе теоретические и прикладные технологические аспекты позволяют контролируемо интегрировать элементы технологии ИИ в разрабатываемые классическим способом информационные системы. Таким образом, продемонстрирована принципиальная возможность использования практически в любых информационных системах в качестве интерфейсно-коммуникационного компонента интеллектуальных агентов и ассистентов на основе больших языковых моделей. При желании и необходимости в рамках описанной технологии функциональная и целевая нагрузка на эти интеллектуальные компоненты может быть скорректирована. В целом способ ИИ-интеграции функций ИС на основе протокола MCP и возможность выбора нужной глубины этой интеграции позволяют решать задачу контролируемого и объяснимого использования современных интеллектуальных технологий.

Полученные результаты коллектив авторов планирует непосредственно использовать для развития своих исследований в области фокусированного сбора и интеллектуализированного извлечения информации из открытых онлайн-источников. В уже начатых и активно продолжающихся по этому направлению работах для ранее сформулированных теоретических и концептуальных аспектов разрабатываются прикладные прототипы, предназначенные для исследовательского тестирования и опробования компонентов обобщающей информационной технологии фокусированного сбора.

Список источников

1. Информационная карта НИОКТР. URL: https://gisnauka.ru/nioktr/detail/3A4F1NQ3Q3O68L885FV6A9XO (дата обращения: 20.10.2025).
2. Датьев И. О., Федоров А. М., Ревякин А. А. Фокусированный сбор и обработка открытых данных социальных медиа // Онтология проектирования. 2024. Т. 14, № 4 (54). С. 569–581. doi:10.18287/2223-9537-2024-14-4-569-581.
3. Han, Junxiao; Yu, Zheng,; Bao, Lingfeng; Liu, Jiakun; Wan, Yao; Yin, Jianwei; Deng, Shuiguang; Han, Song (2025). From LLMs to Agents: A Comparative Evaluation of LLMs and LLM-based Agents in Security Patch Detection. 10.48550/arXiv.2511.08060.
4. Li, Rui; Gu, Jia-Chen; Kung, Po-Nien; Heming, Xia; liu, Junfeng; Kong, Xiangwen; Sui, Zhifang; Peng, Nanyun (2025). LLM-REVal: Can We Trust LLM Reviewers Yet? 10.48550/arXiv.2510.12367.
5. Krishnan, Naveen Kumar (2025). Advancing Multi-Agent Systems Through Model Context Protocol: Architecture, Implementation, and Applications. Preprint. https://arxiv.org/abs/2504.21030.
6. Ayyagari, Vallikranth. Model Context Protocol for Agentic AI: Enabling Contextual Interoperability Across Systems // International Journal of Computational and Experimental Science and Engineering. 2025. Vol. 11, no. 3, Aug. doi:10.22399/ijcesen.3678.
7. Pankaj Agrawal. Model context protocol: Architectural framework for reducing AI dependency conflicts in financial services. *World Journal of Advanced Engineering Technology and Sciences*. 2025. 15 (03). P. 1916–1923. doi:10.30574/wjaets.2025.15.3.1128.
8. Zhang, Qian; Xie, Le. PowerAgent: A Road Map Toward Agentic Intelligence in Power Systems: Foundation Model, Model Context Protocol, and Workflow // IEEE Power and Energy Magazine. 2025. 23. P. 93–101. 10.1109/MPE.2025.3579718.1-8.
9. Model Context Protocol Architecture. URL: https://modelcontextprotocol.io/docs/learn/architecture (дата обращения: 20.10.2025).
10. JSON-RPC 2.0 Specification. URL: https://www.jsonrpc.org/ (дата обращения: 20.10.2025).
11. Pan, Yu; Li, Xiaocheng; Wang, Hanzhao (2025). Online-Optimized RAG for Tool Use and Function Calling. 10.48550/arXiv.2509.20415.
12. Bhat, Vishvesh; Ghugarkar, Omkar; McAuley, Julian (2025). On Generalization in Agentic Tool Calling: CoreThink Agentic Reasoner and MAVEN Dataset. 10.48550/arXiv.2510.22898.

13. Johnson, Reid; Pain, Michelle; West, Jordan (2025). Natural Language Tools: A Natural Language Approach to Tool Calling In Large Language Agents. 10.48550/arXiv.2510.14453.
14. Ersoy, Asim; Altinisik, Enes; Sencar, Husrev; Darwish, Kareem (2025). Tool Calling for Arabic LLMs: Data Strategies and Instruction Tuning. 10.48550/arXiv.2509.20957.
15. Osuagwu, Richard; Cook, Thomas; Masoud, Maraim; Ghosal, Koustav; Mattivi, Riccardo (2025). ScaleCall—Agentic Tool Calling at Scale for Fintech: Challenges, Methods, and Deployment Insights. 10.48550/arXiv.2511.00074.
16. Doh, Seungheon; Choi, Keunwoo; Nam, Juhan (2025). TalkPlay-Tools: Conversational Music Recommendation with LLM Tool Calling. 10.48550/arXiv.2510.01698.
17. Ross, Hayley; Mahabaleshwarkar, Ameya; Suhara, Yoshi (2025). When2Call: When (not) to Call Tools. 10.48550/arXiv.2504.18851.
18. Taulli, T., Deshmukh, G. Introduction to LangGraph // Building Generative AI Agents. Apress, Berkeley, CA, 2025. https://doi.org/10.1007/979-8-8688-1134-0_9.

References

1. R&D Information Card. Available at: https://gisnauka.ru/nioktr/detail/3A4F1NQ3Q3O68L885FV6A9XO (accessed 20.10.2025).
2. Datyev I. O., Fedorov A. M., Reviakin A. A. Focused collection and processing of open social media data. *Ontology of designing*, 2024, 14 (4), pp. 569–581. (In Russ.). doi:10.18287/2223-9537-2024-14-4-569-581.
3. Han, Junxiao & Yu, Zheng & Bao, Lingfeng & Liu, Jiakun & Wan, Yao & Yin, Jianwei & Deng, Shuiguang & Han, Song. (2025). From LLMs to Agents: A Comparative Evaluation of LLMs and LLM-based Agents in Security Patch Detection. 10.48550/arXiv.2511.08060.
4. Li, Rui & Gu, Jia-Chen & Kung, Po-Nien & Heming, Xia & liu, Junfeng & Kong, Xiangwen & Sui, Zhifang & Peng, Nanyun. (2025). LLM-REVal: Can We Trust LLM Reviewers Yet?. 10.48550/arXiv.2510.12367.
5. Krishnan, Naveen Kumar. (2025). Advancing Multi-Agent Systems Through Model Context Protocol: Architecture, Implementation, and Applications. Preprint. https://arxiv.org/abs/2504.21030.
6. Ayyagari, Vallikranth. Model Context Protocol for Agentic AI: Enabling Contextual Interoperability Across Systems. *International Journal of Computational and Experimental Science and Engineering*, 2025, vol. 11, no. 3, Aug. doi:10.22399/ijcesen.3678.
7. Pankaj Agrawal. Model context protocol: Architectural framework for reducing AI dependency conflicts in financial services. *World Journal of Advanced Engineering Technology and Sciences*, 2025, 15 (03), pp. 1916–1923. doi: 10.30574/wjaets.2025.15.3.1128.
8. Zhang, Qian & Xie, Le. PowerAgent: A Road Map Toward Agentic Intelligence in Power Systems: Foundation Model, Model Context Protocol, and Workflow. *IEEE Power and Energy Magazine*, 2025, 23, pp. 93–101. 10.1109/MPE.2025.3579718.1-8.
9. Model Context Protocol Architecture. Available at: https://modelcontextprotocol.io/docs/learn/architecture (accessed 20.10.2025).
10. JSON-RPC 2.0 Specification. Available at: https://www.jsonrpc.org/ (accessed 20.10.2025).
11. Pan, Yu & Li, Xiaocheng & Wang, Hanzhao. (2025). Online-Optimized RAG for Tool Use and Function Calling. 10.48550/arXiv.2509.20415.
12. Bhat, Vishvesh & Ghugarkar, Omkar & McAuley, Julian. (2025). On Generalization in Agentic Tool Calling: CoreThink Agentic Reasoner and MAVEN Dataset. 10.48550/arXiv.2510.22898.
13. Johnson, Reid & Pain, Michelle & West, Jordan. (2025). Natural Language Tools: A Natural Language Approach to Tool Calling In Large Language Agents. 10.48550/arXiv.2510.14453.
14. Ersoy, Asim & Altinisik, Enes & Sencar, Husrev & Darwish, Kareem. (2025). Tool Calling for Arabic LLMs: Data Strategies and Instruction Tuning. 10.48550/arXiv.2509.20957.
15. Osuagwu, Richard & Cook, Thomas & Masoud, Maraim & Ghosal, Koustav & Mattivi, Riccardo. (2025). ScaleCall - Agentic Tool Calling at Scale for Fintech: Challenges, Methods, and Deployment Insights. 10.48550/arXiv.2511.00074.
16. Doh, Seungheon & Choi, Keunwoo & Nam, Juhan. (2025). TalkPlay-Tools: Conversational Music Recommendation with LLM Tool Calling. 10.48550/arXiv.2510.01698.
17. Ross, Hayley & Mahabaleshwarkar, Ameya & Suhara, Yoshi. (2025). When2Call: When (not) to Call Tools. 10.48550/arXiv.2504.18851.
18. Taulli, T., Deshmukh, G. Introduction to LangGraph. In: *Building Generative AI Agents*. Apress, Berkeley, CA, 2025. https://doi.org/10.1007/979-8-8688-1134-0_9.

Информация об авторах

А. М. Федоров — кандидат технических наук, ведущий научный сотрудник;
И. О. Датьев — кандидат технических наук, старший научный сотрудник;
М. О. Илясов — программист;
И. Г. Вишняков — аспирант ФИЦ КНЦ РАН, системный администратор ИИММ КНЦ РАН;
М. О. Базегский — стажер-исследователь;
Д. С. Фигуркин — стажер-исследователь;
К. Д. Любимова — студент.

Information about the authors

A. M. Fedorov — Candidate of Science (Tech.), Leading Research Officer;
I. O. Datyev — Candidate of Science (Tech.), Senior Research Officer;
M. O. Ilyasov — Programmer;
I. G. Vishnyakov — Postgraduate Student of the Federal Research Center of the Kola Science Center of the Russian Academy of Sciences, System Administrator of the IIMM KSC RAS;
M. O. Bazegskiy — Research Intern;
D. S. Figurkin — Research Intern;
C. D. Lyubimova — Student.

Статья поступила в редакцию 22.10.2025; одобрена после рецензирования 27.11.2025; принята к публикации 02.12.2025.
The article was submitted 22.10.2025; approved after reviewing 27.11.2025; accepted for publication 02.12.2025.