СИСТЕМНЫЙ АНАЛИЗ, УПРАВЛЕНИЕ И ОБРАБОТКА ИНФОРМАЦИИ, СТАТИСТИКА

УДК 004.89 Научная статья
https://doi.org/10.35330/1991-6639-2026-28-2-11-24
ВІММВС

Применение конечных автоматов как управляющей модели в агентных RAG-системах для корпоративных баз знаний: системный подход и формализация

Р. С. Розум, А. С. Кузнецов✉

Российский государственный социальный университет
129226, Россия, Москва, ул. Вильгельма Пика, 4, стр. 1

Аннотация. В статье с позиций системного анализа подробно рассмотрено применение теории конечных автоматов в качестве управляющей модели в агентных RAG-системах управления корпоративными базами знаний. Приведены основные требования к данным системам по показателям точности, предсказуемости и управляемости. Рассмотрено применение метода конечных автоматов, что позволяет ввести устойчивые состояния в систему и организовать управление. Приведены практические примеры применения конечных автоматов в RAG-системах. Создано формализованное описание применения FSM-агента в корпоративных системах. Агент в системе рассматривается как управляемый объект, формализуемый в виде конечного автомата, обладающего дискретным набором состояний, с учетом событий среды и сформулированного набора условий переходов. Приведена математическая модель FSM-агента, выполненная на основе математической функции описания множеств. Построены диаграмма компонентов FSM-RAG системы, отражающая архитектурное решение, которое может быть использовано в качестве основы для внедрения в корпоративные информационные системы, и визуальная диаграмма смены состояний FSM-RAG в виде графа переходов. Представленные в статье схемы и модель создают основу для внедрения более надежных агентов. Они будут поддаваться проверке и формализации в бизнес-среде.

Цель исследования – формализация процесса управления агентной RAG-системой корпоративной базы знаний на основе детерминированного конечного автомата и разработка математической модели FSM-агента как управляющей подсистемы.

Методология исследования включает методы системного анализа и синтеза, сравнительный анализ формальных управляющих моделей, построение математической модели автомата Мура, архитектурное моделирование и граф переходов состояний.

Результаты. Предложена формальная модель FSM-агента для RAG-системы; разработана архитектура FSM-RAG с выделением состояний, событий и функций переходов; построены диаграмма компонентов и граф состояний; показана применимость модели для корпоративных баз знаний с неструктурированными данными.

Выводы. Использование детерминированного конечного автомата в роли управляющей модели повышает предсказуемость, наблюдаемость и формальную проверяемость поведения RAG-системы; обеспечивает контролируемую многоэтапную обработку запросов; позволяет анализировать завершимость сценариев и сложность алгоритма; создает основу для внедрения управляемых RAG-агентов в корпоративных информационных системах.

Ключевые слова: метод конечных автоматов, управляющие модели, большие языковые модели, агентные системы, базы знаний, системный подход, формализация

Поступила 14.11.2025, одобрена после рецензирования 16.12.2025, принята к публикации 25.03.2026

Для цитирования. Розум Р. С., Кузнецов А. С. Применение конечных автоматов как управляющей модели в агентных RAG-системах для корпоративных баз знаний: системный подход и формализация // Известия Кабардино-Балкарского научного центра РАН. 2026. Т. 28. № 2. С. 11–24. DOI: 10.35330/1991-6639-2026-28-2-11-24

MSC: 00A72 Original article

Application of finite state machines as a control model in agent-based RAG systems for corporate knowledge bases: a systems approach and formalization

R.S. Rozum, A.S. Kuznetsov✉

Russian State Social University
4, Wilhelm Pieck street, building 1, Moscow, 129226, Russia

Abstract. This research article, from a systems analysis perspective, examines in detail the application of finite automata theory as a control model in agent-based RAG systems for managing corporate knowledge bases. Key requirements for these systems in terms of accuracy, predictability, and controllability are outlined. The application of the finite automata method, which allows for the introduction of stable states into the system and the organization of control, is discussed. Practical examples of the application of finite automata in RAG systems are provided. A formalized description of the application of an FSM agent in corporate systems is created. An agent in the system is considered a controlled object, formalized as a finite automaton with a discrete set of states, taking into account environmental events and a formulated set of transition conditions. A mathematical model of the FSM agent, based on the mathematical function of describing sets, is presented. A diagram of the FSM-RAG system components is constructed, reflecting the architectural solution that can be used as a basis for implementation in corporate information systems. A visual diagram of the FSM-RAG state transitions is presented in the form of a transition graph. The framework and model presented in the article provide a foundation for the implementation of more robust agents. They will be verifiable and formalizable in a business environment.

Aim. The study is to formalize the process of managing an agent-based RAG system of a corporate knowledge based on a deterministic finite state machine and to develop a mathematical model of an FSM agent as a control subsystem.

The **research methodology** includes methods of system analysis and synthesis, comparative analysis of formal control models, construction of a mathematical model of a Moore machine, architectural modeling and a state transition graph.

Results. A formal model of an FSM agent for a RAG system was proposed; an FSM-RAG architecture was developed with the identification of states, events, and transition functions; a component diagram and state graph were constructed; and the applicability of the model to corporate knowledge bases with unstructured data was demonstrated.

Conclusions. Using a deterministic finite state machine as a control model improves the predictability, observability, and formal verifiability of RAG system behavior; provides controlled multi-stage request processing; enables scenario termination and algorithm complexity analysis; creates the basis for implementing controlled RAG agents in corporate information systems.

Keywords: finite state machine method, control models, large language models, agent-based systems, knowledge bases, systems approach, formalization

Submitted 14.11.2025, approved after reviewing 16.12.2025, accepted for publication 25.03.2026

For citation. Rozum R.S., Kuznetsov A.S. Application of finite state machines as a control model in agent-based RAG systems for corporate knowledge bases: a systems approach and formalization. *News of the Kabardino-Balkarian Scientific Center of RAS.* 2026. Vol. 28. No. 2. Pp. 11–24. DOI: 10.35330/1991-6639-2026-28-2-11-24

Введение

В связи с нарастающей популярностью использования Large Language Model (LLM) систем возникают более строгие требования к их точности, безопасности, предсказуемости и управляемости [1]. С точки зрения пользователя LLM – это черный ящик [1], который не всегда выдает предсказуемый и требуемый результат. Для того чтобы ответ от LLM был более предсказуемым и управляемым, а также чтобы не приходилось дорого дообучать модель, используются Retrieval-Augmented Generation (RAG) системы, которые позволяют создавать более понятные и контролируемые схемы управления бизнес-процессами с использованием LLM и внешних данных [2]. Но часто такие схемы становятся очень сложными и мало управляемыми, что возвращает проблему черного ящика в системах с применением LLM. Для решения этой проблемы можно использовать Finite State Machine (FSM), которые позволяют ввести в систему состояния и управлять этими состояниями, что увеличивает предсказуемость и управляемость системой.

В существующих RAG-системах FSM используют как вспомогательный механизм, например, чтобы ввести ограничения на формат вывода или перепроверить вывод из LLM. То есть FSM используется чаще всего на этапе генерации текста LLM, но не для управления всей логикой поведения агента.

В данной статье предлагается способ формализации RAG-систем с LLM с использованием FSM в роли главного управляющего механизма принятия решения агента, включая математическое описание и описание применимости, на примере корпоративной системы знаний.

Предложенная архитектура позволит связать множество шагов RAG-системы, таких как «понимание запроса», «декомпозиция», «извлечение данных», «генерация ответа», «верификация результата», в единый модуль FSM с понятными и управляемыми состояниями.

Научная новизна этого подхода в том, что модель конечного автомата применяется не просто как дополнительный элемент. Например, в таких фреймворках, как Autogen, Langchain, Microsoft AutoGPT и прочих, она используется именно так [3]. Здесь же она становится основным управляющим механизмом. Это обеспечивает предсказуемое и понятное поведение RAG-агента. Существующие решения обычно делают упор на гибкие скрипты, неявные рабочие процессы или деревья поведения. Предложенная модель FSM отличается. Она позволяет описать поведение агента в виде конечного автомата. При этом возможны проверка, оценка полноты, отслеживание переходов и анализ состояний системы.

Особенность в том, что модель используется для корпоративных баз знаний. В этом контексте нужны более строгие требования к объяснимости. Также важны надежность и контроль над взаимодействием с пользователем и источниками данных.

Обзор существующих подходов к управлению поведением RAG-систем

RAG-системы, которые используют LLM, обычно реализуются в виде многоэтапных конвейеров, включающих в себя анализ запроса, извлечение релевантных данных из базы данных, генерацию ответа [6]. Эти этапы также делятся на различные способы анализа запроса, для структурированных и неструктурированных данных, различные источники данных, способность генерировать ответ несколькими моделями и алгоритмами. Поэтому вопрос управления последовательностью этапов и условиями переходов между ними является ключевым для обеспечения предсказуемой и объяснимой работы RAG-системы.

Одним из самых распространенных подходов к управлению RAG является использование строго заданных сценариев или линейных конвейеров. Другими словами, сценарный подход. Такой подход прост в реализации, но при этом не позволяет явно фиксировать текущее состояние системы, что затрудняет анализ поведения системы, из-за чего могут снижаться предсказуемость, стабильность и релевантность ответов. Также в такой системе все компоненты имеют жесткую связанность, и изменить или добавить новые правила и сценарии без влияния на все компоненты системы становится очень сложным.

Другим направлением в управлении RAG-системами является использование workflow-движков и систем оркестрации [6]. В них этапы обработки запроса рассматриваются как независимые задачи. Такой подход позволяет более гибко конфигурировать последовательность вызовов, но в первую очередь он ориентирован преимущественно на управление вычислительными процессами и ресурсами, а также не предоставляет строгую формальную модель поведения такой системы. Логика принятия решений часто остается распределенной и под управлением различных внешних систем, что слабо поддается формальному анализу.

В задачах управления поведением интеллектуальных систем часто применяются поведенческие деревья (Behavior Trees). Они позволяют описывать сложные сценарии с иерархией условий и действий. Такой подход широко используется в робототехнике и игровых AI-системах, однако для RAG-систем он может иметь ограничения. Поведенческие деревья обычно ориентированы на реактивное поведение и не обеспечивают удобный или формализованный механизм фиксации глобальных состояний системы, что усложняет анализ достижимости конечных состояний и завершения сценария в целом.

В современных фреймворках для построения LLM-агентов и RAG-систем, таких как AutoGen, LangChain и аналогичные, конечные автоматы могут использоваться в качестве вспомогательного механизма [3]. FSM применяются для ограничения формата вывода, контроля отдельных этапов генерации текстов или реализации простых переходов между агентами [9]. При этом FSM не рассматривается как основной механизм управления всей RAG-системой, а используется на отдельных участках конвейера обработки запроса.

Для управления поведением интеллектуальных систем могут применяться более сложные формальные модели, такие как автоматы с магазинной памятью или иерархические конечные автоматы [4]. Эти модели позволяют описывать вложенные и рекурсивные сценарии обработки, а также динамически изменяющиеся структуры. Однако использование таких моделей приводит к ряду практических ограничений при внедрении в реальные RAG-системы. Так как такие модели характеризуются частичной определенностью поведения, возрастает структурная сложность модели, повышается аналитическая сложность формальной верификации. В отличие от этих автоматов классические конечные автоматы обеспечивают полностью определенность переходов, явную фиксацию текущего состояния и конечное множество сценариев. Это делает FSM более удобной, простой и практически применимой управляющей моделью для RAG-систем.

Сравнительный анализ формальных управляющих моделей для управления RAG-системой

Для формального сравнения управляющих моделей используются следующие критерии [4]:

– определенность переходов – существует ли однозначное соответствие между текущим состоянием, входным событием и следующим состоянием;

– наблюдаемость состояния – возможно ли восстановить текущее состояние системы на основе внешних данных без скрытого контекста;

– структура памяти – наличие или отсутствие дополнительной памяти, влияющей на поведение;

– формальная проверяемость – возможность анализа свойств системы, таких как достижимость конечных состояний, завершимость сценариев и полнота переходов;

– декомпозиция поведения – возможность разбиения логики на конечное множество независимых или слабо связанных состояний без нарушения целостности модели;

– детерминированность выполнения – воспроизводимость поведения при одинаковых вводных параметрах и условиях среды.

Таблица 1. Сравнительный анализ управляющих моделей

Table 1. Comparative analysis of control models

Критерий	Детерминированный конечный автомат (FSM)	Иерархический конечный автомат (HFSM)	Автомат с магазинной памятью (PDA)
Определенность переходов	Переход определяется однозначной функцией $\delta: S \times \Sigma \to S$	Переход определяется функцией δ с учетом активного уровня иерархии	Переход определяется функцией δ с учетом состояния магазина
Наблюдаемость состояния	Текущее состояние полностью задается элементом множества S	Текущее состояние определяется комбинацией активного состояния и уровня иерархии	Текущее состояние определяется парой: состояние, содержимое магазина
Структура памяти	Дополнительная память отсутствует	Используется иерархический контекст состояний	Используется неограниченная магазинная память
Формальная проверяемость	Анализ достижимости, завершимости и полноты переходов осуществляется на конечном графе состояний	Анализ требует учета вложенных автоматов и межуровневых переходов	Анализ требует учета динамически изменяемого содержимого магазина
Декомпозиция поведения	Поведение декомпозируется на конечное множество независимых состояний	Поведение декомпозируется с учетом иерархической вложенности	Поведение декомпозируется с учетом зависимостей от состояния магазина
Детерминированность выполнения	Поведение детерминировано при детерминированной функции переходов	Детерминированность зависит от структуры иерархии	Детерминированность зависит от операции над магазином

В таблице 1 приведен сравнительный анализ основных управляющих моделей, основанных на конечных автоматах по приведенным критериям.

Детерминированные конечные автоматы обеспечивают однозначность в поведении системы [4]. Для каждого состояния и события есть заданный переход, что позволяет в любой момент однозначно определить текущее состояние RAG-системы и восстановить цепочку принятых решений.

В иерархических автоматах состояние определяется не только текущим узлом, но и активным уровнем вложенности, что приводит к частичной определенности поведения и усложняет анализ достижимости финальных состояний системы [4]. Иерархический автомат может гарантировать завершение отдельных ветвей, локальных сценариев, но не всей системы глобально.

В автоматах с магазинной памятью поведение дополнительно зависит от содержимого стека. Содержимое стека не ограничивается конечным набором состояний, что образует потенциально бесконечное множество конфигураций [4]. Это усложняет формальную проверку свойств системы, таких как достижимость конечных состояний.

Как FSM дополняет архитектуру RAG

RAG уже активно используется во многих корпоративных системах с применением LLM. Применение FSM – это способ усилить архитектуру RAG за счет управления поведением системы [5]. При работе с RAG можно столкнуться со следующими проблемами:

– нет понятия текущего этапа или состояния обработки запроса, в процессе работы система находится в неизвестном состоянии;

– нельзя задать гибкие стратегии поведения переходов между этапами;

– отсутствуют переходы между состояниями в зависимости от условий среды;

– система сама принимает решения, поэтому невозможно оценить сложность работы системы.

Использование FSM представляет логику поведения RAG-системы как управляемую последовательность состояний, где каждый этап – это отражение текущего состояния системы. FSM описывает функции и правила их вызова: «что и когда», не затрагивая саму логику системы. Можно описать состояния в виде отдельных функций, например: PARSE, RETRIEVE, GENERATE, REPHRASE, ERROR и т.д. А далее просто управлять вызовами этих функций в зависимости от данных и текущего состояния системы (табл. 2).

Таблица 2. Примеры практического применения FSM в RAG

Table 2. Examples of practical application of FSM in RAG

Сценарий	Использование FSM
Retriever дал нерелевантные документы	FSM позволяет перейти в *REPHRASE -> RETRY* вместо генерации
Нужно выбрать источник по типу запроса	В состоянии *RETRIEVE* FSM учитывает тип запроса и выбирает источник
Нужно уточнить данные у пользователя	FSM переходит в *REQUEST_FEEDBACK*, вызывает внешний модуль и переходит в режим ожидания
Контроль глубины итераций	FSM хранит счетчик, ограничивает циклы *REPHRASE -> RETRY*

Таким образом, FSM здесь выполняет роль управляющего механизма, который определяет стратегию работы системы на основе текущего состояния и внешних условий.

Это позволяет создать дискретную управляющую систему, обладающую наблюдаемостью и устойчивостью, с возможностью формального анализа поведения.

Математическая модель FSM-агента как управляющей подсистемы

Для применения в корпоративных системах поведение FSM-агента должно быть формализовано. Агент в RAG-системе рассматривается как управляемый объект, поведение которого описывается конечным автоматом с дискретными состояниями, событиями среды и возможностью описать условия переходов. С точки зрения системного анализа такой подход является наиболее формализованным в такой системе.

Математическая модель FSM-агента для RAG выглядит так:

$$A = (S, \Sigma, \delta, s_0, F, \omega),$$

где:

S – дискретное множество фаз управления жизненным циклом запроса, введенное для обеспечения наблюдаемости, ограниченности и формального анализа поведения RAG-системы;

Σ – конечное множество наблюдаемых управляющих событий, отражающих результаты выполнения этапов RAG-системы, а не обрабатываемые данные (например, событие входящего запроса, результат обработки запроса, ошибки и так далее);

$\delta: S \times \Sigma \to S$ – функция переходов между состояниями, которая задает детерминированную политику управления фазами RAG-системы и определяет допустимые сценарии обработки на основе наблюдаемых событий;

$s_0 \in$ S – начальное состояние, вводится для формального задания точки инициализации управляющего процесса обработки запроса и обеспечивает единое начало всех сценариев поведения RAG-системы;

$F \subseteq S$ – множество завершающих состояний (например, *FINISH*, *ERROR*, *REQUEST_FEEDBACK* и другие). Этот параметр вводится для формального задания логического завершения управляющего сценария обработки запроса и позволяет анализировать корректность и завершимость поведения RAG-системы;

$\omega: S \to A$ – функция действий, выполняемых агентом при нахождении в состоянии (вызов модуля поиска, выход в цикл уточнения, запрос в языковую модель и прочее). Формализует управляющие воздействия FSM на компоненты RAG-системы путем сопоставления каждому состоянию соответствующих операций и обеспечивает разделение логики управления и исполнения.

Таким образом, предложенная модель представляет собой расширенный автомат Мура (FSM), дополненный функцией действий ω, что позволяет определить поведение в каждом состоянии, а не только описать логику переходов.

Выбор автомата Мура как управляющей модели обусловлен тем, что RAG представляет собой фазовый процесс обработки запроса. Использование автомата с фиксированными состояниями позволяет формально зафиксировать этап обработки, обеспечить наблюдаемость и отделить управление от выполнения вычислений.

Особенности обработки неструктурированных данных в FSM-управляемых RAG-системах

В большинстве своем корпоративные базы знаний – это неструктурированные данные [6]. В них могут содержаться документы в различных форматах PDF, DOCX, XLS, PNG и другие. Это усложняет конвейер RAG из-за добавления множества этапов и стратегий извлечения и обработки данных из разных источников в разных форматах. Из-за отсутствия явной модели управления RAG-системой эти этапы обычно реализуются в виде неформальных эвристик, что снижает предсказуемость и понятность поведения RAG-системы, увеличивая риски возникновения ошибок и неопределенностей.

Использование FSM в качестве модели управления RAG-системой позволяет явно выделить этапы обработки неструктурированных данных и формализовать стратегии переходов между ними. FSM позволяет улучшить качество извлечения информации из неструк-

турированных источников путем введения наблюдаемых управляющих событий, на основании которых система может принимать решения, например, о дополнительной постобработке данных или об уточнении запроса у пользователя.

Специфика работы с неструктурированными данными отражается в модели через множество событий Σ и функцию переходов δ.

Предложенный подход позволяет сохранить детерминированность и наблюдаемость RAG-системы при работе в том числе и с неструктурированными данными.

Анализ влияния размена базы знаний на поведение FSM-управляемой RAG-системы

В корпоративных системах размер базы знаний может достигать сотен и тысяч документов, так же, как и постоянный рост этой базы знаний, это оказывает прямое влияние на характеристики RAG-системы, в первую очередь на этапе извлечения информации. Увеличение объема базы знаний приводит к увеличению времени поиска, снижению релевантности извлекаемых данных и увеличению вероятности повторных циклов обработки запроса.

Отсутствие формализованного механизма управления приводит к неконтролируемому увеличению времени и числу обращений к LLM. Из-за этого увеличивается неопределенность поведения RAG-системы и усложняется анализ возможных ошибок и возможной деградации производительности системы.

Использование FSM для управления RAG-системой позволяет локализовать влияние масштаба базы знаний в рамках отдельных состояний и правил. За счет детерминированной функции переходов δ FSM обеспечивает управляемость поведения системы при любом объеме базы знаний и при ее увеличении со временем.

Такой подход не устраняет вычислительную сложность извлечения информации, но обеспечивает предсказуемость и формальный анализ поведения RAG-системы при любом масштабе базы знаний.

Многоэтапные сценарии обработки запросов в RAG-системе под управлением FSM-агента

В корпоративных системах часто используются сложные многоэтапные сценарии обработки запросов. В RAG-системах без формализованного механизма управления невозможно знать заранее число этапов обработки, сложно контролировать возвраты и количество повторных циклов, а также гарантировать логичное завершение процесса [7, 8].

FSM позволяет формализовать многоэтапную обработку запроса в виде последовательности и циклов состояний, которые управляются функцией переходов δ. FSM явно фиксирует текущую фазу обработки, дает контроль над глубиной многоэтапного анализа за счет конечности множества состояний, явного задания допустимых циклов переходов и обязательного достижения завершающих состояний и позволяет детерминировать переходы между этапами на основе наблюдаемых событий.

В рамках предложенной модели повторные этапы обработки реализуются через циклы в функции переходов δ, а завершение многоэтапного сценария определяется достижением одного из состояний множества F. Такой подход обеспечивает управляемость, наблюдаемость и формальную анализируемость поведения RAG-системы при обработке сложных и многоэтапных запросов, не нарушая конечность модели.

Архитектура RAG-системы с управляющим FSM-агентом

Архитектура RAG-системы с управляющим FSM-агентом представляет собой набор ключевых компонентов системы и их связи. Предложенная на рисунке 1 архитектура может быть применена как шаблон для внедрения в корпоративные информационные системы [9].

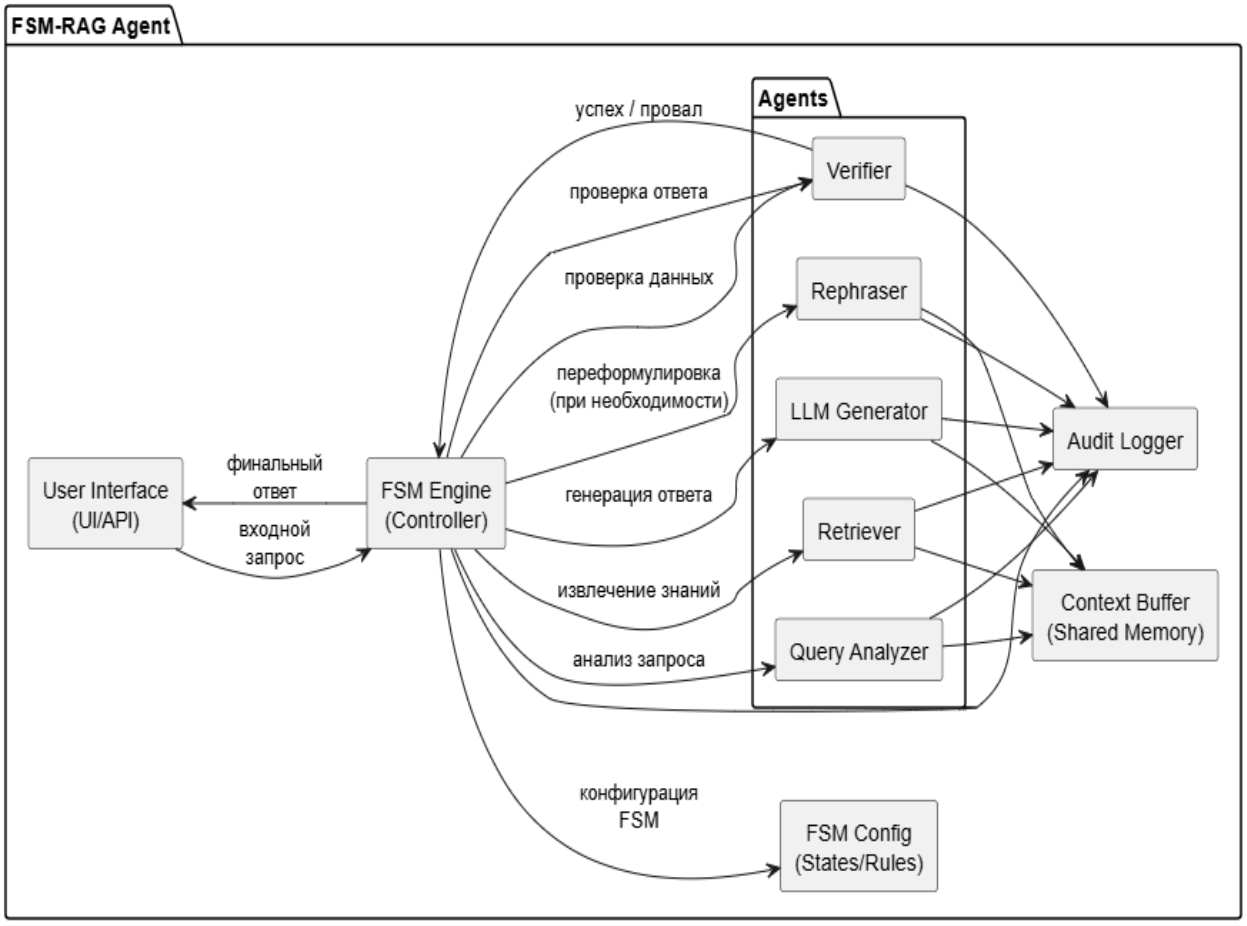

***Рис. 1**. Диаграмма компонентов FSM-RAG системы*

***Fig. 1**. FSM-RAG system component diagram*

Взаимодействие компонентов осуществляется по следующей схеме: FSM-Engine получает входной запрос от пользователя, инициирует последовательные вызовы агентов (модуль анализа, извлечение информации, генератор ответа). Полученный результат проходит проверку на полноту и соответствие заданным параметрам, в случае некорректности FSM инициирует переформулирование запроса, приводящее к повторному циклу поиска и генерации. Все промежуточные результаты, статусы и сигналы сохраняются в контекстном буфере и используются FSM-Engine для принятия решений о переходах. Дополнительно журнал всех событий, переходов, вызовов и ответов фиксируется в подсистеме Audit Logger для целей анализа ошибок и отладки.

На рисунке 2 представлена диаграмма переходов состояний FSM-агента, который реализует управление RAG-системой.

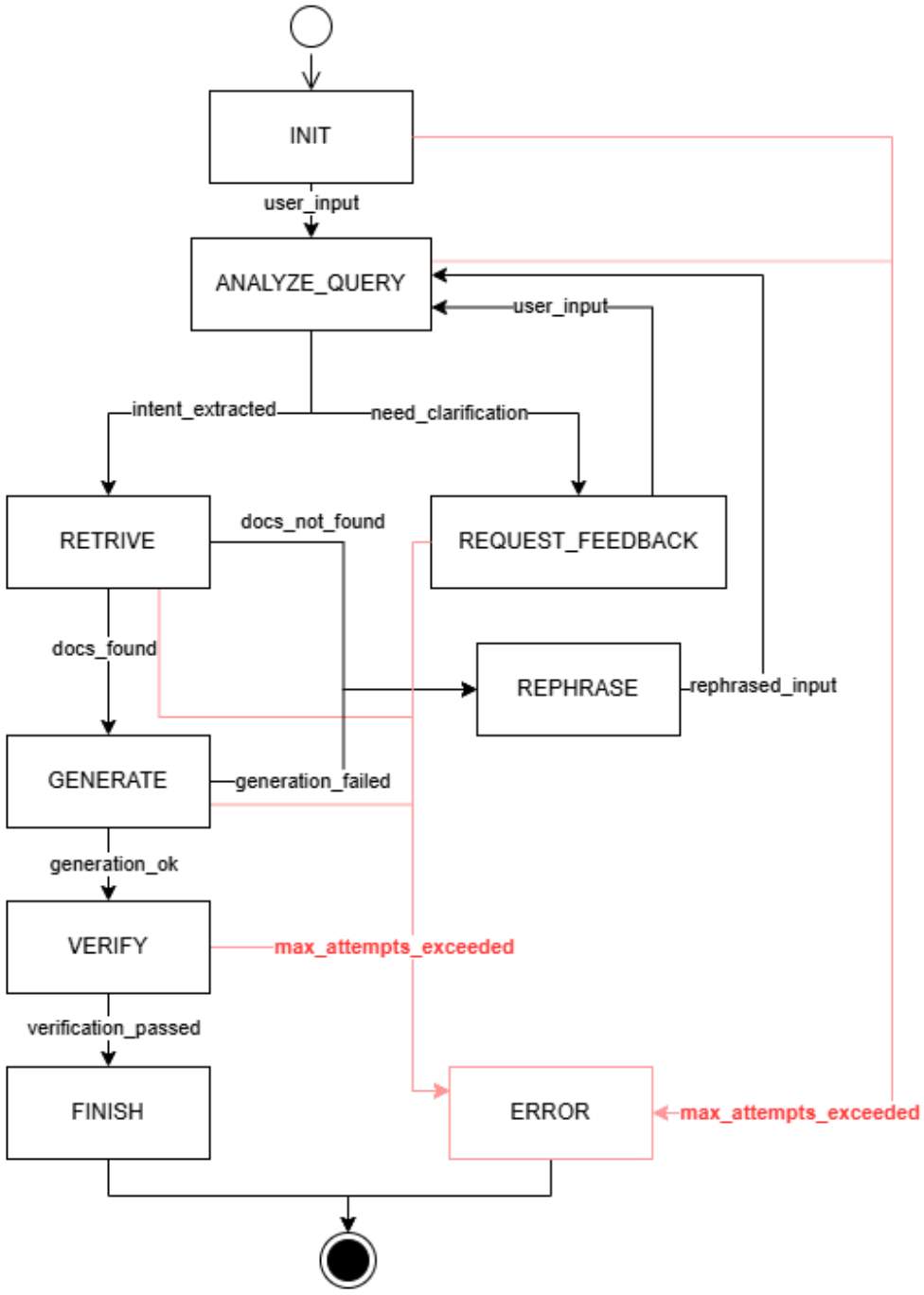

***Рис. 2**. Граф переходов состояний FSM-RAG системы*

***Fig. 2**. State transition graph of the FSM-RAG system*

Приведенные выше формальные модели FSM-агента для RAG-систем отображают основные компоненты архитектуры и переходы между состояниями, что позволяет строго формализовать RAG-систему [10].

Компоненты архитектуры и граф переходов состояний соответствуют математической модели FSM, описанной выше:

– S – соответствует фазам: анализ, поиск, генерация (*INIT*, *ANALYZE_QUERY*, *REQUEST_FEEDBACK*, *RETRIVE*, *GENERATE*, *VERIFY*, *REPHRASE*, *FINISH*, *ERROR*);

– Σ – внешние и внутренние события (*user_input*, *docs_found* и прочее);

– $\delta: S \times \Sigma \to S$ – логика вызовов агентов (проверка ответа, генерация, поиск);

– $s_0 \in S$ – начальное состояние (*INIT*);

– $F \subseteq S$ – множество завершающих состояний (например, *FINISH*, *ERROR*, *REQUEST_FEEDBACK* и другие);

– $\omega: S \to A$ – вызов соответствующего агента в зависимости от состояния (например, *RETRIVE* → *Модуль поиска*).

Таким образом, получается формальная модель FSM для RAG, которая позволяет строго определить все переходы, действия, состояния. Поведение RAG и LLM перестает быть «черным ящиком». Можно делать выводы о сложности модели, полноте и корректности автомата, а также можно протестировать модель и доказать завершимость алгоритма (доходимость до FINISH или ERROR). Например, можно провести расчет верхней границы числа действий агента.

Допустим:

– максимальное число переформулировок (*REPHRASE*) – k;

– общее число переходов в цепочке без ошибок – 5 ($INIT \to ANALYZE\_QUERY \to$ RETRIEVE $\to$ GENERATE $\to$ VERIFY);

– количество переходов при одном цикле переформулировки (*REPHRASE*) – 3, например, если не удалось найти документы или ответ не валиден ($REPHRASE: ANALYZE\_QUERY \to$ RETRIVE $\to$ GENERATE).

Тогда

$$T_{\max} = 5 + 3k.$$

Вывод:

– получаем $T_{\max}$ – верхнюю границу числа действий агента;

– можно использовать это значение для оценки производительности;

– можно использовать как лимит для работы системы.

Реализация FSM как управляющего механизма дает возможность использовать предложенную модель в корпоративных системах поддержки знаний [10, 11]. Эти системы нуждаются в высокой степени надежности, объяснимости и предсказуемости. Такие сценарии охватывают техническую поддержку, доступ к нормативным документам, консультации на основе внутренних баз знаний.

Заключение

В этой работе предложено управление поведением агента RAG на основе конечного автомата в качестве базовой модели управления. Проанализировав некоторые из существующих архитектур и потребности корпоративных интеллектуальных систем, можно обосновать, почему FSM лучше всего подходит для задач, ориентированных на предсказуемость, объяснимость и управляемость взаимодействия с языковыми моделями [12]. Построенная формальная модель была полезна для описания поведения агента: для оценки сложности сценария, анализа того, завершится ли когда-нибудь какой-либо процесс, и пригодна для структурного тестирования всех возможных переходов.

В рамках работы показано, что использование FSM в роли управляющей подсистемы позволяет учитывать особенности корпоративных систем, включая работу с неструктурированными источниками данных, масштабирование базы знаний и обработку сложных многоэтапных запросов.

Разработанная архитектура показывает, как модель работает на практике в корпоративных базах знаний. FSM позволяет четко определить логику взаимодействия частей, составляющих систему RAG: анализ запросов, стратегия извлечения контекста, генерация ответов и проверка. Представленные схемы и модель создают основу для внедрения более надежных агентов. Они будут поддаваться проверке и формализации в бизнес-среде. В будущем подход может быть расширен за счет включения иерархических автоматов или вероятностных вариантов. Кроме того, его стоит адаптировать к мультиагентным системам [8], где роли и контексты динамично меняются.

СПИСОК ЛИТЕРАТУРЫ

1. *Намиот Д. Е., Ильюшин Е. А.* О киберрисках генеративного Искусственного Интеллекта // International Journal of Open Information Technologies. 2024. Vol. 12. No. 10. Pp. 109–119. EDN: JZCUQS

2. *Намиот Д. Е., Ильюшин Е. А.* Архитектура LLM агентов // International Journal of Open Information Technologies. 2025. Т. 13. № 1. С. 67–74. EDN: VIMKYB

3. FSM Group Chat – User-specified agent transitions – https://microsoft.github.io/autogen/0.2/blog/2024/02/11/FSM-GroupChat (дата обращения: 13.11.2025 г.)

4. *Нестеров Л. А.* Методы синтеза и анализа дискретных конечных автоматов // Resour. Technol. 2001. № 3. URL: https://cyberleninka.ru/article/n/metody-sinteza-i-analiza-diskretnyh-konechnyh-avtomatov (дата обращения: 13.11.2025).

5. Can large language models help developers with robotic finite state machine modification. https://arxiv.org/html/2412.05625v1 (дата обращения: 13.11.2025 г.)

6. *Куприяновский В. П., Аленьков В. В., Соколов И. А. и др.* Умная инфраструктура, физические и информационные активы, Smart Cities, BIM, GIS и IoT // International Journal of Open Information Technologies. 2017. Vol. 5. No. 10. Pp. 55–86. EDN: ZISODV

7. *Youssef Maklad, Fares Wael, Wael Elsersy, Ali Hamdi.* Retrieval augmented generation based LLM evaluation for protocol state machine inference with chain-of-thought reasoning. https://arxiv.org/abs/2502.15727 (дата обращения: 13.11.2025 г.)

8. MetaAgent: Automatically Constructing multi-agent systems based on Finite State Machines. https://arxiv.org/abs/2507.22606 (дата обращения: 13.11.2025 г.)

9. What OpenAI ChatGPT Pro Means for AI Agents and Agentic AI. https://www.teneo.ai/blog/what-openai-chatgpt-pro-means-for-ai-agents-and-agentic-ai Retrieved: Dec, 2024

10. *Розум Р. С., Кузнецов А. С.* Системный подход к оценке эффективности методов кластеризации и их программных реализаций // Рефлексия. 2025. № 2. С. 87–91. EDN: XASNUX

11. *Розум Р. С.*, *Кузнецов А. С.* Архитектура автоматизированной информационной системы обработки и семантического анализа запросов к системам обслуживания реального времени // Тенденции развития науки и образования. 2024. № 115–15. С. 101–107. DOI: 10.18411/trnio-11-2024-709. EDN: DOSWGC

12. *Кузнецов А. С.* Информационное моделирование объектов, процессов и систем: принципы формализации, классификации и верификации: монография. Санкт-Петербург: Сциентиа, 2026. 106 с. ISBN-13 (15) 978-5-907902-76-3

REFERENCES

1. Namiot D.E., Eugene E.A. On cyber risks of generative artificial intelligence. *International Journal of Open Information Technologies*. 2024. Vol. 12. No. 10. 109–119. EDN: JZCUQS. (In Russian)

2. Namiot D.E., Ilyushin E.A. Architecture of LLM agents. *International Journal of Open Information Technologies.* 2025. Vol. 13. No. 1. Pp. 67–74. EDN: VIMKYB. (In Russian)

3. FSM Group Chat – User-specified agent transitions. https://microsoft.github.io/autogen/0.2/blog/2024/02/11/FSM-GroupChat (accessed: 13/11/2025)

4. Nesterov L.A. Methods for synthesis and analysis of discrete finite automata. *Resour. Technol.* 2001. No. 3. URL: https://cyberleninka.ru/article/n/metody-sinteza-i-analiza-diskretnyh-konechnyh-avtomatov (accessed: 11/13/2025). (In Russian)

5. Can large language models help developers with robotic finite state machine modification. https://arxiv.org/html/2412.05625v1 (accessed: 13/11/2025)

6. Kupriyanovsky V.P., Alenkov V.V., Sokolov I.A. et al. Smart infrastructure, physical and information assets, Smart Cities, BIM, GIS and IoT. *International Journal of Open Information Technologies*. 2017. Vol. 5. No. 10. Pp. 55–86. EDN: ZISODV. (In Russian)

7. *Youssef Maklad, Fares Wael, Wael Elsersy, Ali Hamdi.* Retrieval augmented generation based LLM evaluation for protocol state machine inference with chain-of-thought reasoning. https://arxiv.org/abs/2502.15727 (accessed: 13/11/2025)

8. MetaAgent: Automatically constructing Multi-Agent Systems based on finite state machines. https://arxiv.org/abs/2507.22606 (accessed: 13/11/2025)

9. What OpenAI ChatGPT Pro Means for AI Agents and Agentic AI. https://www.teneo.ai/blog/what-openai-chatgpt-pro-means-for-ai-agents-and-agentic-ai Retrieved: Dec, 2024

10. Rozum R.S., Kuznetsov A.S. A systems approach to assessing the effectiveness of clustering methods and their software implementations. *Reflection*. 2025. No. 2. Pp. 87–91. EDN: XASNUX. (In Russian)

11. Rozum R.S., Kuznetsov A.S. Architecture of an automated information system for processing and semantic analysis of requests to real-time service systems. *Trends in the Development of Science and Education*. 2024. No. 115–15. Pp. 101–107. DOI: 10.18411/trnio-11-2024-709. EDN: DOSWGC. (In Russian)

12. Kuznetsov A.S. *Informatsionnoye modelirovaniye ob"yektov, protsessov i sistem: printsipy formalizatsii, klassifikatsii i verifikatsii* [Information modeling of objects, processes and systems: principles of formalization, classification and verification: monograph]. St. Petersburg: Scientia, 2026. 106 p. ISBN-13 (15) 978-5-907902-76-3. (In Russian)

Вклад авторов: все авторы сделали эквивалентный вклад в подготовку публикации. Авторы заявляют об отсутствии конфликта интересов.

Contribution of the authors: the authors contributed equally to this article. The authors declare no conflict of interest.

Финансирование. Исследование проведено без спонсорской поддержки.

Funding. The study was performed without external funding.

Информация об авторах

Розум Роман Сергеевич, аспирант кафедры информационных технологий, искусственного интеллекта и общественно-социальных технологий цифрового общества, Российский государственный социальный университет;

129226, Россия, Москва, ул. Вильгельма Пика, 4, стр. 1;

romanrozum@yandex.ru, ORCID: https://orcid.org/0000-0002-2276-842X, SPIN-код: 2190-5760

Кузнецов Андрей Сергеевич, канд. тех. наук, доцент, доцент кафедры информационных технологий, искусственного интеллекта и общественно-социальных технологий цифрового общества, заместитель руководителя по научной деятельности факультета политических и социальных технологий, Российский государственный социальный университет;

129226, Россия, Москва, ул. Вильгельма Пика, 4, стр. 1;

askgoogle@internet.ru, ORCID: https://orcid.org/0000-0003-1569-4765, SPIN-код: 8442-7210

Information about the authors

Roman S. Rozum, Postgraduate Student, Department of Information Technology, Artificial Intelligence and Public and Social Technologies of the Digital Society, Russian State Social University;

4, Wilhelm Pieck street, building 1, Moscow, 129226, Russia;

romanrozum@yandex.ru, ORCID: https://orcid.org/0000-0002-2276-842X, SPIN-code: 2190-5760

Andrey S. Kuznetsov, Candidate of Engineering Sciences, Associate Professor, Associate Professor of the Department of Information Technologies, Artificial Intelligence and Social Technologies of Digital Society, Russian State Social University;

4, Wilhelm Pieck street, building 1, Moscow, 129226, Russia;

askgoogle@internet.ru, ORCID: https://orcid.org/0000-0003-1569-4765, SPIN-code: 8442-7210