

Управление контекстом в интегрированных средах разработки на базе искусственного интеллекта на примере Cursor

Л.П. Москаленко, А.В. Соколова, М.Ю. Тимохин

Санкт-Петербургский государственный архитектурно-строительный университет

Аннотация: В статье рассматриваются интегрированные среды разработки на базе искусственного интеллекта как инновационный инструмент программирования, обеспечивающий автоматизацию рутинных задач разработки программного обеспечения. В качестве основного объекта исследования выступает среда разработки Cursor, разработанная компанией Anysphere. Проводится анализ архитектурных особенностей системы, включая агентный подход к взаимодействию с кодом, механизмы управления контекстом через генерацию с дополнением извлеченными данными, индексацию кодовой базы с использованием векторных представлений и деревьев Меркла для оптимизации обновлений. Выявлены ключевые ограничения современных интегрированных сред разработки на базе искусственного интеллекта: проблемы с размером контекстного окна, производительностью индексации больших репозиторий, точностью извлечения контекста, а также вопросы приватности и безопасности. Особое внимание уделяется человеческому фактору – недостаточной компетентности разработчиков в области эффективного управления контекстом и создания качественных промптов. Статья обосновывает необходимость создания предварительного агента для управления контекстом, способного технически оптимизировать процессы и направлять пользователей к эффективным практикам работы с интегрированными средами разработки на базе искусственного интеллекта.

Ключевые слова: интегрированная среда разработки, искусственный интеллект, среда разработки Cursor, большие языковые модели, управление контекстом, генерация с дополнением извлеченными данными, индексация кодовой базы, деревья Меркла, агентный подход, разработка программного обеспечения.

Введение

Интегрированные среды разработки (далее – ИСР) на базе искусственного интеллекта (далее – ИИ) представляют собой инновационное развитие традиционных инструментов программирования, где большие языковые модели (Large Language Models – LLM) интегрируются для автоматизации рутинных задач и оптимизации всего цикла создания программного обеспечения (далее – ПО). В эпоху стремительного научно-технологического прогресса информационные технологии выступают ключевым вектором инноваций, а системы на основе ИИ и машинного обучения позволяют эффективно обрабатывать огромные объемы данных в

реальном времени, выявлять скрытые закономерности и генерировать объективные варианты решений [1]. Особенно актуально это для современных корпоративных ИТ-проектов, которые характеризуются высокой сложностью, мультидисциплинарным подходом и необходимостью работы с большими данными – как текстовыми, так и визуальными [2]. В результате ИСР на базе ИИ позволяют разработчикам сосредоточиться на стратегических и творческих аспектах работы, делегируя рутинные операции – от генерации кода до выявления уязвимостей – интеллектуальным системам, что в целом повышает продуктивность и качество конечного продукта [3].

Одной из ключевых особенностей ИСР на базе ИИ служит интеллектуальное дополнение кода. Оно выходит за пределы простого автодополнения строк и предлагает целые блоки кода, опираясь на контекст проекта. Функция усиливается контекстно-зависимыми рекомендациями: они анализируют общие цели и структуру кодовой базы, предлагая персонализированные варианты, адаптированные к стилю кода разработчика. Кроме того, такие среды оснащены механизмами обнаружения и исправления ошибок в режиме реального времени – они сканируют код на потенциальные уязвимости, предлагают решения и тем самым снижают число синтаксических ошибок, а также сокращают время на ручную отладку [4].

Разнообразие ИСР с интеграцией ИИ отражается в различных уровнях их совместимости с существующими платформами разработки – от бесшовного внедрения до отдельных плагинов, специально адаптированных для инструментов вроде кроссплатформенного редактора исходного кода Visual Studio Code (далее – VS Code) от компании Microsoft. Кроме того, системы различаются по степени поддержки языков программирования и фреймворков: от веб-ориентированных до системных [5]. Для пользователей

различия также заметны между решениями с открытым исходным кодом и коммерческими разработками. Первые обеспечивают повышенную кастомизацию, хотя и требуют более сложной настройки, в то время как вторые обеспечивают премиум-функции без необходимости самостоятельного выбора и оплаты конкретных LLM. В корпоративных средах особый акцент делается на обеспечение безопасности, гарантируя защиту кодовой базы и используемой информации.

Среда разработки Cursor

Одним из ведущих примеров ИСР на базе ИИ является среда разработки Cursor – редактор кода, эффективно использующий ИИ для достижения целей, разработанный компанией Anysphere как ответвление от Visual Studio Code, выпущенный в марте 2023 года [6, 7]. Интерфейс Cursor сохранил знакомый дизайн VS Code, но добавил специализированные панели для взаимодействия с ИИ, включая чат-окно и инструменты для контекстного управления кодом.

По данным на июнь 2025 года компания привлекла 900 млн долларов в раунде финансирования Series C при оценке в 9,9 млрд долларов, что сделало ее одной из самых быстрорастущих стартапов в сфере ИИ [8]. Cursor имеет более 1 млн ежедневных активных пользователей (Daily Active Users – DAU) и показал рост годового повторяющегося дохода (Annual Recurring Revenue – ARR) на 9900% по сравнению с предыдущим годом [9]. Популярность Cursor подтверждается его усиливающимся ростом: по оценкам, в 2024 году он набрал 1 млн пользователей всего за 16 месяцев после запуска, из которых 360 тыс. – платящие клиенты, без значительных маркетинговых кампаний, в основном за счет рекомендаций разработчиков [10].

Основная логика работы Cursor, как прослойки между пользователем и LLM, строится на комбинации системных и пользовательских запросов для ИИ (промптов), а также инструментов. Системные промпты задают роль

модели, к примеру: "Ты – мощный агентский помощник по кодированию на основе искусственного интеллекта, работающий на базе Claude 3.5 Sonnet. Ты работаешь исключительно в Cursor – лучшей в мире среде разработки", и определяют структуру ответа, состоящего из этапов планирования, анализа и редактирования кода. Пользовательские промпты представляют собой запросы в чате или команды автодополнения кода от разработчика. В момент отправки запроса Cursor интегрирует его с системными инструкциями, а также добавляет @-теги для ссылок на файлы, указанные пользователем, создавая единый контекст для LLM. Агентный подход Cursor также объясняется использованием инструментов, таких как чтение файла, редактирование файла, поиск в Интернете, поиск по кодовой базе и других – именно это позволяет прослойке между пользователем и LLM итеративно взаимодействовать с кодом разработчика, вызывая инструменты для получения данных, корректируя действия на основе результатов и ожидая подтверждения или дополнительных уточнений от пользователя [11].

Cursor позволяет расширить базовые инструменты посредством протокола контекста модели (Model Context Protocol – MCP), который предоставляет большим языковым моделям дополнительный контекст через интеграцию с внешними системами и данными. Таким образом, разработчики могут использовать собственноручно настроенные инструменты для обращения к внешним источникам, при этом используя человекочитаемые текстовые запросы, которые преобразуются в вызов внешних функций [11].

Работа с контекстом

Переходя к более глубокому пониманию принципов работы Cursor, хочется отметить, что эффективность ИСР напрямую зависит от концепции контекстного окна – объема информации, который LLM может учитывать одновременно при обработке запроса и генерации ответа, измеряющегося в

токенах – базовых единицах, на которые модель разбивает текст. Чем больше контекстное окно, тем большим объемом информации о проекте располагает модель, что позволяет ей генерировать более релевантные ответы. Однако расход токенов ограничен вычислительными и финансовыми затратами, что побуждает разработчиков к более эффективному управлению контекстом и его оптимизации.

В Cursor управление контекстом реализовано через несколько механизмов, адаптированных для различных сценариев взаимодействия с пользователем. Один из них – автодополнение кода по нажатию клавиши Tab, через которое модель генерирует предложения, учитывая локальный контекст текущего файла. Собранный код шифруется на клиентской стороне и отправляется на сервер Cursor, который использует либо собственную модель, либо популярные LLM для кодогенерации, и генерирует предложения, возвращаемые пользователю в зашифрованном виде.

Для выполнения более сложных задач Cursor предлагает взаимодействие с чатом – многоступенчатый процесс, интегрирующий генерацию с дополнением извлеченными данными (Retrieval-Augmented Generation – RAG) – подход, где модель обогащается извлеченными данными, так как сервер Cursor не хранит код на сервере, а периодически запрашивает необходимые фрагменты. Процесс происходит следующим образом:

- 1) Код пользователя разбивается на «чанки» (chunks) – логически разделенные фрагменты, – которые шифруются и отправляются на сервер Cursor [11].

- 2) На сервере на основе чанков с помощью OpenAI генерируются эмбединги (векторные представления) [12], хранимые в векторной базе данных Turbopuffer. Процесс, называемый индексацией кодовой базы, представляет собой представление текста (кода) в числовые векторы

фиксированной размерности, где расстояние между векторами отражает их семантическую близость [11].

3) После индексации, в случае если пользователь внес изменения в файл, потребовалось бы пересобрать всю структуру проекта, однако это обходят через деревья Меркла – древовидную структуру, в которой каждый листовый узел помечен криптографическим хешем блока данных, а каждый нелистовой узел – хешем меток своих дочерних узлов. Такая древовидная структура позволяет обнаруживать изменения на любом уровне путем сравнения хеш-значений. Первым делом ИСР локально фрагментирует файлы кодовой базы, а затем сканирует проект и вычисляет дерево Меркла из хешей всех допустимых файлов, кроме файлов «.gitignore» и «.cursorignore». Затем, хеш-дерево на сервере сравнивают с деревом, построенным локально у пользователя, и вычисляют несовпадения хешей – обновляют соответствующие файлы по цепочке, обновляя индексацию [11].

4) Далее, при поступлении пользовательского запроса в чат, система применяет RAG: генерирует эмбединг запроса, выполняет поиск по семантической близости в векторной базе данных для нахождения релевантных фрагментов, клиент локально извлекает актуальный код из указанных файлов, собирает контекст и отправляет его вместе с запросом на сервер для обработки LLM, которая генерирует ответ с учетом обогащенного контекста.

Недостатки и альтернативы

Несмотря на все преимущества Cursor, ИСР имеет ряд недостатков. На основе отзывов пользователей (forum.cursor.com, X.com, Reddit.com) можно выделить следующие проблемы:

- ограничения размера контекстного окна: Cursor не может эффективно обрабатывать очень большие кодовые базы из-за лимитов контекстного окна LLM;

- замедление производительности при индексации больших репозиторий: индексация может заикливаться или занимать много времени/ресурсов;
- низкая точность и полнота в RAG: возвращается много лишней информации и пропускаются важные фрагменты кода, что приводит к необходимости вручную указывать файлы для добавления в контекст и прописывать правила в файле «.cursorrules»;
- задержки и повышенные ресурсозатраты: обработка контекста вызывает ошибки, повышенное использование памяти и медленную генерацию;
- проблемы с приватностью и безопасностью контекста: хотя код не хранится на серверах, шифрование и отправка фрагментов вызывают опасения утечек, особенно в корпоративных средах;
- ограниченность в рамках подписки: подписка на Cursor стоит фиксированную сумму, но пользователи не видят детальной статистики по расходу токенов.

Становится очевидным, что эффективное разрешение этих проблем требует определенных решений для мониторинга, динамического управления контекстом и его оптимизации. Это позволило бы разработчикам отслеживать перегрузку контекстного окна, предотвращать потери информации и корректировать поведение системы в режиме реального времени. Однако, поскольку Cursor представляет собой программное обеспечение с закрытым исходным кодом, прямые модификации его прослойки недоступны для разработчиков.

В качестве альтернативы стоит обратить внимание на открытые ИИ-ориентированные ИСР и инструменты, которые позволяют работать с исходным кодом и вносить изменения. Важно отметить, что в этих проектах управление контекстом может отличаться от Cursor: например, в Cline [13]

произошел частичный переход от RAG к агентному поиску: через просмотр дерева папок, анализ импортов, построение и разбор абстрактных синтаксических деревьев (Abstract Syntax Trees – AST) и рассуждения о дальнейших действиях. Такой подход, обозначаемый разработчиками как «контекстная инженерия», позволяет модели не извлекать релевантные фрагменты из векторного индекса, а самостоятельно формировать внутреннюю карту проекта, подобно тому, как это делает человек. В Cline реализованы механизмы динамического управления контекстом: агент адаптирует окно восприятия в зависимости от сложности задачи, а также использует «память проекта» и специальные контекстные файлы, сохраняющие понимание состояния между сессиями [14].

Заключение

Проведенный анализ ИСР на базе ИИ, в частности Cursor, демонстрирует значительный потенциал интеграции больших языковых моделей в процессы разработки программного обеспечения. Были выявлены ключевые архитектурные особенности системы, включая агентный подход к взаимодействию с кодом, многоуровневую систему управления контекстом и механизм индексации кодовой базы с использованием деревьев Меркла, что особенно важно для производительности в реальных проектах разработки. На текущий момент ИСР на базе ИИ имеют ряд ограничений, связанных с размером и использованием контекстного окна. Однако, помимо этого, на эффективность использования системы существенно влияет человеческий фактор – существует фундаментальная проблема неграмотности разработчиков в области эффективного управления контекстом и создания качественных промптов, а также чрезмерного перекладывания ответственности на LLM при работе с большими объемами данных.

Пользователи зачастую не обладают достаточными навыками для оптимального использования контекста, что приводит к неэффективному

расходуванию токенов и снижению качества ответов ИИ-ассистентов. Особенно критична ситуация с новыми членами команды разработки, которые пытаются использовать ИИ для понимания архитектуры и логики существующих решений, вместо того чтобы полагаться на документацию и историю проекта, которые, возможно, не были оставлены участниками команды. Дополнительным фактором неэффективности является недоиспользование существующих инструментов и возможностей ИСР: многие разработчики не осознают или не используют весь потенциал таких функций, как автоматическое создание документации в процессе разработки, интеграция с протоколом контекста модели (MCP), документирование архитектурных решений и процессов разработки – команды попросту не извлекают максимальную пользу из доступных технологических возможностей.

Перечисленные проблемы указывают на необходимость создания предварительного агента для управления контекстом, способного не только технически оптимизировать процессы, но и направлять пользователей к использованию эффективных практик работы с ИИ-ориентированными ИСР. Особое внимание следует уделить вопросам адаптации системы под различные уровни экспертизы разработчиков и созданию интуитивных интерфейсов для управления контекстом.

Литература

1. Тимохин М.Ю., Шаранин В.Ю. Искусственный интеллект и теория принятия решений: современные тенденции. Инженерный вестник Дона. 2023. № 10. URL: ivdon.ru/ru/magazine/archive/n10y2023/8746.
2. Касымов А.А., Лысенко А. Синтез нейронных сетей и системного анализа с применением сократических методов для управления корпоративными ИТ-проектами. Инженерный вестник Дона. 2025. № 2. URL: ivdon.ru/ru/magazine/archive/n2y2025/9819.

3. Курочкин И.А., Хартман А. Cursor AI как инструмент специалиста по защите информации. Фундаментальные и прикладные исследования в информатике и цифровизации: Материалы симпозиума XX (LII) Международной научной конференции студентов, аспирантов и молодых ученых, посвященной 80-й годовщине Победы в Великой Отечественной войне. Кемерово. Кемеровский государственный университет. 2025. С. 188-191.
 4. Sergazy M., Tokseit D.K. Enhancing developer productivity with integrated artificial intelligence and cybersecurity considerations. Искусственный интеллект и обратные задачи в науке, технике и индустрии. Астана. Евразийский национальный университет им. Л.Н. Гумилева. 2025. Рр. 377-378.
 5. Горбенко Ю.А. AI в программировании: помощник или конкурент? Научный диалог: теория и практика: Сборник научных статей по итогам работы Международного научного форума. Москва. ООО "Инфинити". 2025. С. 41-48. DOI 10.34660/INF.2025.29.62.005.
 6. Сайт продукта Cursor. URL: cursor.com/home.
 7. Kumar Y., Akinwunmi I., Kruger D. Evaluating the Advantage of an AI-Native IDE Cursor on Programmer Performance. 2025 IEEE Integrated STEM Education Conference (ISEC). Princeton. NJ, USA. 2025. Рр. 1-8. DOI 10.1109/ISEC64801.2025.11147402.
 8. Azevedo Mary Ann. AI-Powered Coding Tool Anysphere Raises \$900M at \$9.9B Valuation – Its Third Round In Less Than One Year. URL: news.crunchbase.com/ai/anysphere-cursor-venture-funding-thrive.
 9. Сайт компании Anysphere. URL: research.contrary.com/company/anysphere.
 10. How Many People Use Cursor AI in 2025? URL: word-spinner.com/blog/how-many-people-use-cursor-ai.
-

11. Официальная документация Cursor. URL: cursor.com/ru/docs.
12. Vector embeddings. URL: platform.openai.com/docs/guides/embeddings.
13. Сайт продукта Cline. URL: cline.bot.
14. Baumann Nick. AI Engineering Isn't Magic, It's Method: Key Strategies for Building Better Software Faster. URL: cline.bot/blog/ai-engineering-isnt-magic-its-method-key-strategies-for-building-better-software-faster.

Авторы согласны на обработку и хранение персональных данных.

References

1. Timokhin M.Yu., Sharanin V.Yu. Inzhenernyj vestnik Dona. 2023. № 10. URL: ivdon.ru/ru/magazine/archive/n10y2023/8746.
2. Kasymov A.A., Lysenko A. Inzhenernyj vestnik Dona. 2025. № 2. URL: ivdon.ru/ru/magazine/archive/n2y2025/9819.
3. Kurochkin I.A., Khartman A. Materialy simpoziuma XX (LII) Mezhdunarodnoy nauchnoy konferentsii studentov, aspirantov i molodykh uchenykh, posvyashchennoy 80-y godovshchine Pobedy v Velikoy Otechestvennoy voyne. Kemerovo. Kemerovskiy gosudarstvennyy universitet. 2025. P. 188-191.
4. Sergazy M., Tokseit D.K. Iskusstvennyy intellekt i obratnye zadachi v nauke, tekhnike i industrii. Astana. Evraziyskiy natsional'nyy universitet im. L.N. Gumileva. 2025. Pp. 377-378.
5. Gorbenko Yu.A. Sbornik nauchnykh statey po itogam raboty Mezhdunarodnogo nauchnogo foruma. Moskva. OOO "Infiniti". 2025. P. 41-48. DOI 10.34660/INF.2025.29.62.005.
6. Sayt produkta Cursor [Cursor Product website]. URL: cursor.com/home.

7. Kumar Y., Akinwunmi I., Kruger D. 2025 IEEE Integrated STEM Education Conference (ISEC). Princeton. NJ, USA. 2025. Pp. 1-8. DOI 10.1109/ISEC64801.2025.11147402.

8. Azevedo Mary Ann. AI-Powered Coding Tool Anysphere Raises \$900M at \$9.9B Valuation – Its Third Round In Less Than One Year. URL: news.crunchbase.com/ai/anysphere-cursor-venture-funding-thrive.

9. Sayt kompanii Anysphere [Anysphere Company Website]. URL: research.contrary.com/company/anysphere.

10. How Many People Use Cursor AI in 2025? URL: word-spinner.com/blog/how-many-people-use-cursor-ai.

11. Ofitsial'naya dokumentatsiya Cursor. URL: cursor.com/ru/docs.

12. Vector embeddings. URL: platform.openai.com/docs/guides/embeddings.

13. Sayt produkta Cline. URL: cline.bot.

14. Baumann Nick. AI Engineering Isn't Magic, It's Method: Key Strategies for Building Better Software Faster. URL: cline.bot/blog/ai-engineering-isnt-magic-its-method-key-strategies-for-building-better-software-faster.

Дата поступления: 1.10.2025

Дата публикации: 27.11.2025