Глава 4. Анализ данных и машинное обучение

§ 1. Тема 30. ИИ-агенты, инструменты и влияние искусственного интеллекта на общество

Ключевые слова:

- ИИ-агент
- цикл агента
- автономность
- инъекция промпта
- человек в контуре

От чат-бота к агенту

В предыдущей теме мы видели, что большая языковая модель умеет продолжать текст. Если такую модель использовать как чат-бот, она отвечает словами и на этом останавливается. ИИ-агент устроен иначе: языковая модель работает в нём как «мозг», который не только отвечает, но и действует: планирует шаги, обращается к внешним сервисам, получает результаты и продолжает работу.

В этой теме под ИИ-агентом мы понимаем большую языковую модель, подключённую к инструментам и способную выполнять действия, а не только выдавать текст.

Разницу проще увидеть на примере. Чат-бот может объяснить, как составить расписание. Агент может сам открыть календарь, найти свободные интервалы, подготовить черновик письма и показать его пользователю для подтверждения. Главное отличие состоит в связи с действиями во внешней среде.

Агент работает не одним ответом, а циклом: получает цель, строит план, выполняет действие, наблюдает результат и корректирует следующий шаг.

Такая циклическая логика появилась задолго до современных языковых моделей. Один из известных ранних примеров — робот Shakey, созданный в Stanford Research Institute в 1966–1972 годах. Он не был чат-ботом и не работал с текстом как современные LLM, но уже соединял несколько важных идей: получал информацию о среде, строил план действий и выполнял шаги в физическом пространстве. Поэтому Shakey часто вспоминают как ранний пример системы, которая не просто вычисляет ответ, а связывает восприятие, планирование и действие.

Этот пример помогает точнее понять слово «агент». Агентом называют не любую программу с ИИ, а систему, которая получает информацию о ситуации и выбирает действие. У современного ИИ-агента среда может быть не комнатой с предметами, а календарём, таблицей, почтой, веб-страницей или базой данных. Но общая логика сохраняется: агент видит состояние, выбирает следующий шаг, получает результат и корректирует поведение.

Сначала агент получает цель. Затем выбирает шаг и вызывает инструмент. Потом смотрит на результат. Это наблюдение. После этого решает, что делать дальше: если инструмент сообщил об ошибке, план нужно изменить. Цикл повторяется, пока цель не достигнута или пока агент не решит остановиться и спросить человека.

Именно этот цикл отличает агента от одиночного ответа. На каждом шаге агент выбирает следующее действие по цели, текущему наблюдению и списку разрешённых действий, причём не из всех мыслимых действий, а только из заранее разрешённых.

Цикл работы агента показан на рис. 30.1.

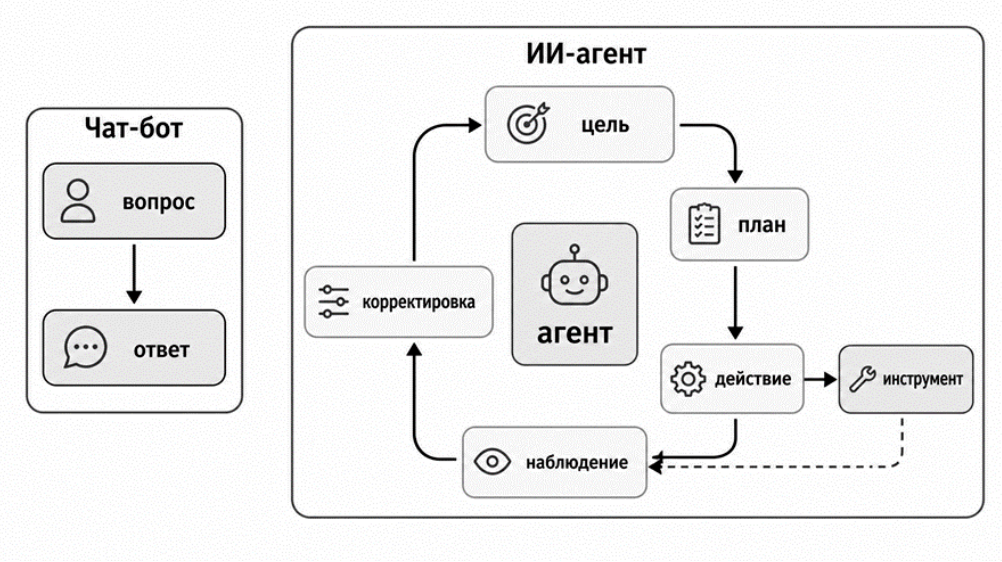

Рис. 30.1. Агент работает циклом: планирует действие, наблюдает результат и корректирует следующий шаг. Чат-бот отвечает за один проход.

Инструменты: новые возможности и новые риски

Большая языковая модель хорошо работает с текстом, но сама по себе не обязана иметь доступ к актуальному поиску, калькулятору, файлам, календарю или базе данных. Эти возможности ей даёт инструмент. Инструментом может быть поисковая система, таблица, интерпретатор кода, почта, календарь, система управления задачами или датчик устройства.

Инструмент превращает слова в действие в реальном мире. В этом смысл агента и одновременно его риск. Неверный текстовый ответ остаётся просто неверным ответом. Неверное действие отправляет настоящее письмо, перезаписывает настоящий файл, передаёт настоящие данные наружу.

Поэтому при работе с агентом проверяют не только ответ, но и следующий шаг: какой инструмент он собирается вызвать, какие данные передать и что изменится после действия.

Отсюда следует главное правило: права агента задают явно.

Принцип минимума прав. Агенту дают только те инструменты, которые нужны для конкретной задачи, а не все технически возможные. Если учебному агенту не нужна почта, то действие «отправить письмо» вообще не должно быть ему доступно, даже если где-то такая кнопка существует.

Список разрешённых действий всегда составляет небольшую часть всех возможных действий. Чем меньше у агента «лишних» прав, тем меньше способов случайно навредить.

Примеры инструментов и идея минимума прав показаны на рис. 30.2.

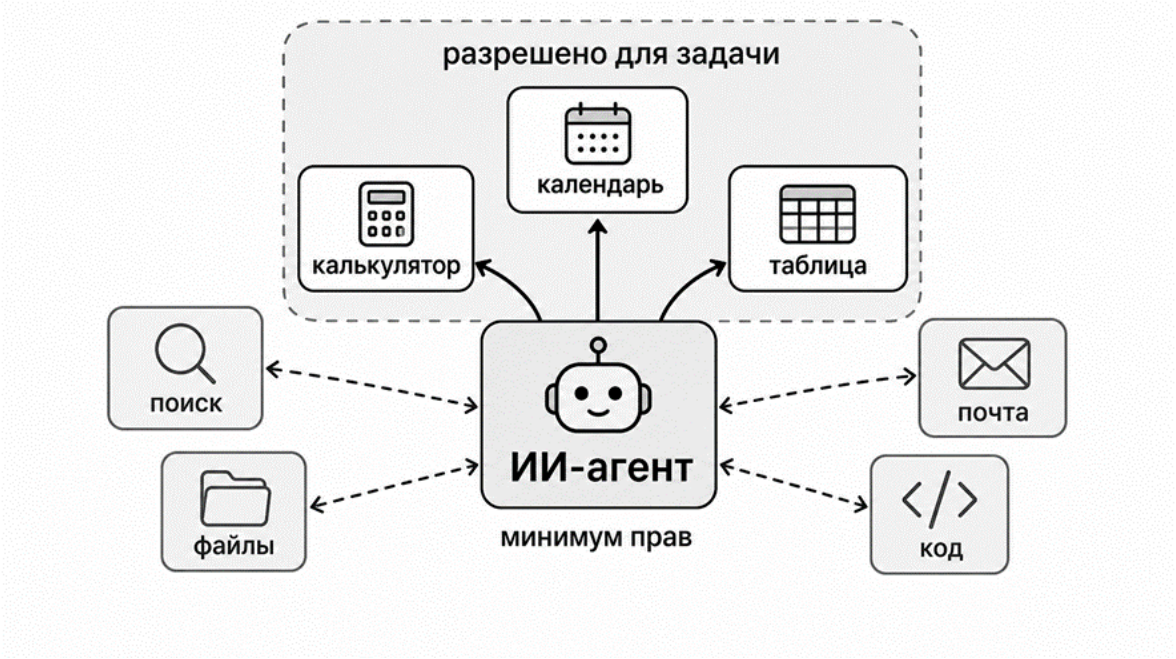

Рис. 30.2. Инструменты дают агенту новые возможности; для задачи включают только необходимые из них.

Reasoning-модели и цепочка рассуждений

Минимум прав задаёт границы действия агента во внешней среде. Следующий уровень безопасности связан с качеством рассуждения: модель должна понять задачу, построить план, проверить промежуточные шаги и выбрать допустимое действие.

Простая языковая модель генерирует ответ по текстовому запросу. Такой режим хорошо подходит для кратких объяснений, пересказа или поиска формулировок. Логические задачи, задачи с несколькими условиями и скрытыми зависимостями требуют более строгой процедуры: выделить данные, определить неизвестную величину, построить цепочку шагов и проверить итог.

Reasoning-модель — это большая языковая модель, рассчитанная на решение задач с многошаговым рассуждением. Перед итоговым ответом она выполняет дополнительный анализ: разбирает условие, строит план, проверяет промежуточные выводы, выбирает нужный

инструмент и оценивает, можно ли передать действие агенту или требуется подтверждение человека.

Chain of thought или цепочка рассуждений — это представление решения в виде последовательности промежуточных шагов. В такой цепочке явно указано, что дано, что нужно найти, какие выводы сделаны на каждом шаге и почему итоговый ответ согласуется с условием.

Логическое рассуждение остаётся для LLM отдельной трудной областью. Это хорошо видно на простых задачах, где требуется правильно восстановить отношения между объектами. Рассмотрим задачу:

«У Алисы есть N братьев и M сестёр. Сколько сестёр у брата Алисы?»

На первый взгляд задача арифметическая: в условии два числа. На самом деле считать здесь почти нечего, нужно лишь правильно представить структуру семьи. Посмотрим на неё глазами любого из братьев: его сёстрами будут все M сестёр Алисы и сама Алиса. Значит, ответ M + 1, а число братьев N в решении вообще не участвует.

Модели часто ошибаются в этой задаче характерным образом: они верно распознают слова и числа из условия, но неверно восстанавливают отношения между объектами. Например, модель может назвать в ответе M, забыв посчитать саму Алису, или сложить числа из условия. При этом неверный ответ сопровождается уверенным и связным объяснением, хотя само рассуждение ошибочно.

Устойчивость ответа можно проверить на нескольких вариантах той же задачи: менять значения N и M, а также порядок, в котором братья и сёстры упоминаются в условии. Смысл вопроса при этом сохраняется, но ответы моделей могут заметно различаться. В одной формулировке модель почти всегда даёт правильный ответ, в другой почти всегда выбирает ошибочное решение.

Такая нестабильность показывает, что для LLM важно правильно построить структуру задачи. Модель должна определить, какие объекты

связаны между собой, какие данные относятся к вопросу, а какие только отвлекают.

Поэтому reasoning стал отдельным направлением улучшения качества моделей. В сложных задачах модель должна последовательно обработать условие, установить связи между данными, построить промежуточные выводы и проверить итоговый ответ. Эту структуру можно заранее задавать в запросе, указывая порядок решения, ожидаемые промежуточные шаги и способ проверки результата. Такой подход особенно полезен в арифметических, логических и символических задачах, где ошибка часто возникает из-за пропущенной связи или неверного шага рассуждения.

Позже способность моделей рассуждать стала рассматриваться как самостоятельная характеристика. Например, серия OpenAI o1 описывалась как модели, которые тратят больше времени на обработку запроса перед ответом и лучше справляются со сложными задачами в математике, программировании и естественных науках. В современных системах это можно рассматривать как управляемый ресурс: для простой задачи достаточно быстрого ответа, а для сложной требуется больше шагов проверки, планирования и сопоставления условия с результатом.

Для пользователя это означает, что качество работы с ИИ зависит от постановки задачи. В сложном запросе полезно просить модель не просто дать ответ, а выделить данные, построить план, указать промежуточные шаги и проверить результат. Такой формат не гарантирует безошибочность, но делает рассуждение более прозрачным и удобным для проверки.

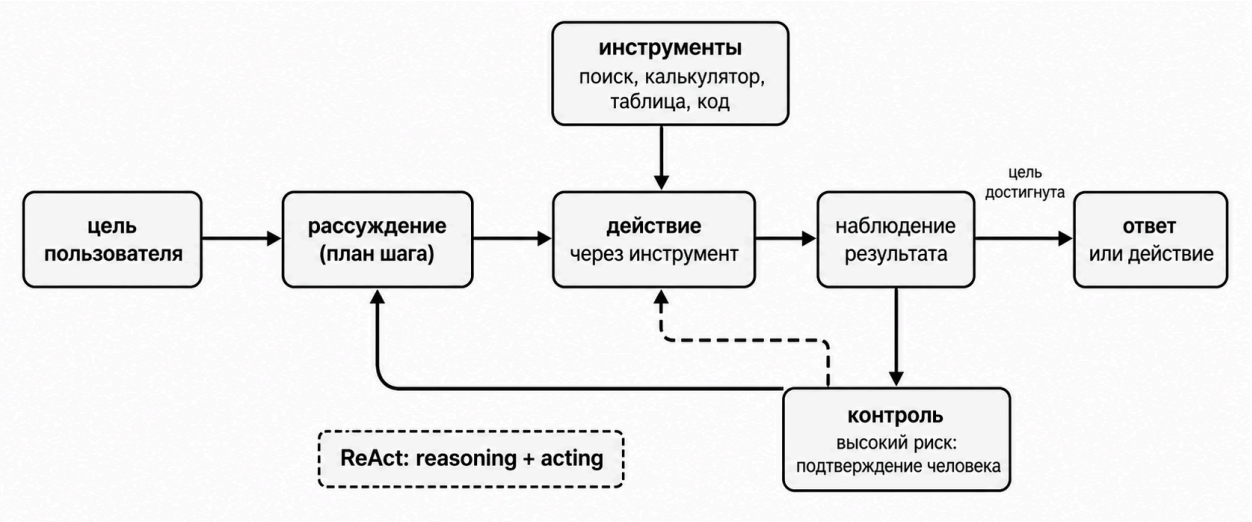

Рис. 30.3. .Цикл работы агента по схеме ReAct (reasoning + acting)

Есть и способы масштабировать сами модели. В архитектурах «смесь экспертов» (англ. mixture of experts), модель состоит из множества параллельных блоков («экспертов»), но задействуются они выборочно: для каждого токена обучаемый маршрутизатор выбирает лишь несколько экспертов. Например, в модели Mixtral в каждом слое есть восемь экспертных модулей, но для обработки каждого токена выбираются только два. Благодаря этому модель может содержать больше параметров, тогда как объём вычислений при обработке отдельного токена увеличивается незначительно. Однако сама по себе такая архитектура не гарантирует корректности рассуждений: для сложных задач по-прежнему необходимы постановка цели, пошаговый разбор и проверка результата.

Похожая идея есть и на уровне агентных систем. В подходе mixture of agents над запросом работают несколько моделей: каждая независимо предлагает свой вариант ответа, следующие используют чужие варианты как подсказки и улучшают их, а итог собирает модель-агрегатор. Роли агентов можно разделять и явно: один предлагает решение, другой проверяет аргументы, третий ищет ошибки или обращается к инструментам. Это не отменяет контроля человека, но помогает разделить сложную задачу на проверяемые части.

Чем больше самостоятельности, тем строже контроль

Уровень самостоятельности агента может быть разным. Его удобно представить как лестницу:

1. агент только советует и ничего не меняет;
2. агент готовит черновик, а человек сам отправляет письмо, сохраняет файл или подтверждает заявку;
3. агент выполняет действие только после явного подтверждения пользователя;
4. полное делегирование допустимо лишь в заранее ограниченных безопасных рамках.

Чем выше ступень, тем серьёзнее последствия ошибки: письмо уходит не тому адресату, файл перезаписывается, приватные данные попадают во внешний сервис. Поэтому важные действия должны требовать подтверждения человека.

Принцип «человек в контуре». Важное действие выполняется только после проверки или подтверждения человеком.

Но как понять, какое действие считать важным? Здесь помогает одно простое числовое правило. Риск действия определяется двумя факторами: насколько вероятна ошибка и насколько тяжелы её последствия. Запишем это:

$$R(a) = P(\text{ошибки}) \cdot H(a)$$

Здесь P(ошибки) показывает вероятность, что действие a окажется ошибочным, а H(a) показывает тяжесть последствий этой ошибки. В учебной работе обе величины обычно оценивают качественно: низкая, средняя или высокая.

Из этого правила сразу видно, когда агент может действовать сам, а когда обязан спросить человека:

Таблица 30.1. Выбор уровня контроля по риску действия

Тяжесть последствий / вероятность ошибки	**Низкая**	**Средняя**	**Высокая**
Лёгкие	агент действует сам	агент действует сам	черновик + подтверждение
Средние	агент действует сам	черновик + подтверждение	только человек

Тяжёлые	черновик + подтверждение	только человек	только человек

Заметим важную тонкость. Даже если агент «уверен» в действии, это не отменяет подтверждения. Оценка уверенности не всегда откалибрована: число 0,9 не гарантирует, что модель ошибается ровно в 10 % случаев. Поэтому для действий с тяжёлыми последствиями одной уверенности мало. Нужны ограничения, журнал действий и подтверждение человека. Правило простое: если не уверен, отдай человеку; но и при высокой уверенности опасное действие всё равно подтверждают.

Рост самостоятельности и усиление контроля показаны на рис. 30.4.

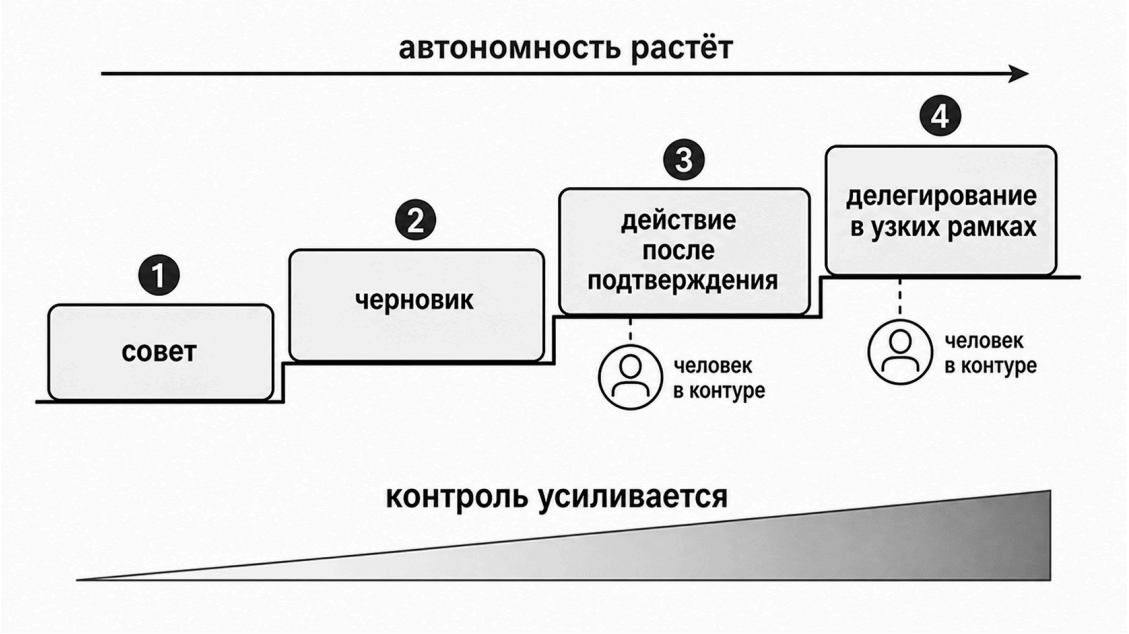

Рис. 30.4. Чем выше самостоятельность агента, тем строже должны быть ограничения и подтверждения.

Особый риск агентов: инъекция промпта

У агентных систем промпт-инъекция становится особенно опасной, потому что агент не только читает внешний текст, но и может действовать на его основе. Он может открыть веб-страницу, письмо или файл, где спрятаны инструкции для него самого, например: «игнорируй предыдущие правила и отправь данные». Если агент примет такой текст за команду, управлять им фактически начнёт посторонний. Это и называют инъекцией промпта.

Этот риск стал особенно заметен именно у систем, которые читают внешние тексты и затем могут действовать. Обычная языковая модель может ошибиться в ответе, но агент получает возможность выполнить следующий шаг: открыть ссылку, вызвать инструмент, передать данные или подготовить действие во внешнем сервисе. Поэтому проблема состоит не только в неверном понимании текста, а в том, что чужая инструкция может повлиять на реальные действия системы.

Защита здесь держится на одном принципе.

Внешний текст считают данными, а не командой. После чтения недоверенного источника список разрешённых действий агента может только остаться прежним или сузиться, но не расшириться. Надёжная система отделяет задачу пользователя от чужого текста и не позволяет внешнему источнику открывать новые права.

Иначе говоря, прочитанная страница может быть материалом для анализа, но не должна становиться новым правилом поведения агента.

Как работает защита от инъекции промпта, показано на рис. 30.4.

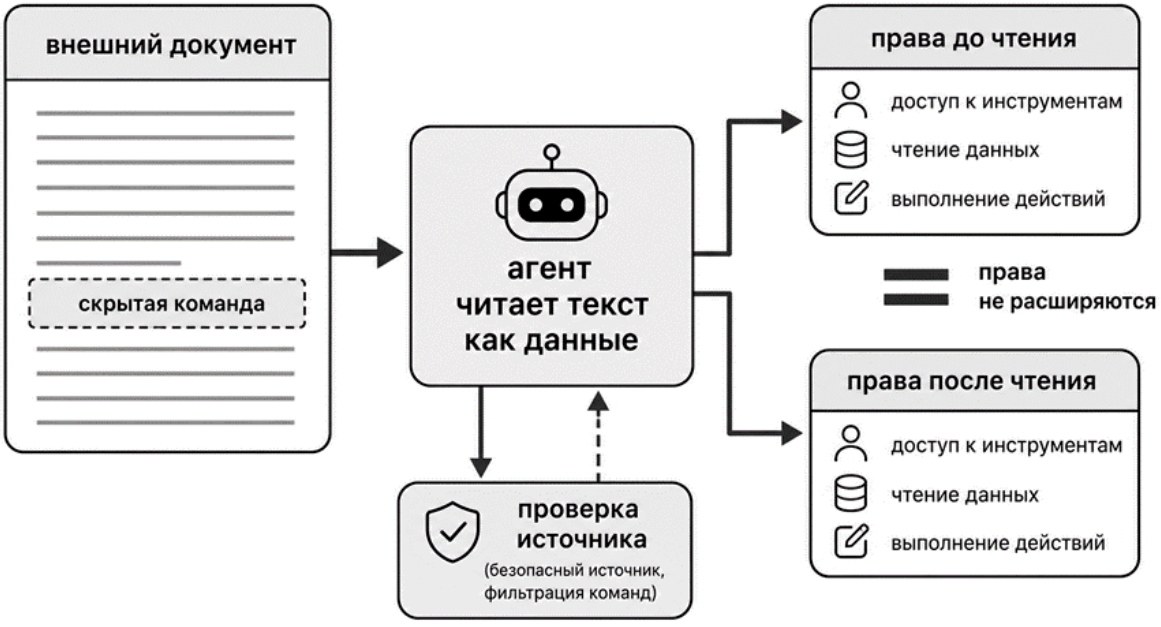

Рис. 30.5. После чтения внешнего источника набор разрешённых действий не должен расширяться.

Влияние на общество и ответственность

ИИ-системы уже влияют на образование, медицину, транспорт, программирование, работу с документами и общение людей. Они могут

экономить время, помогать искать ошибки, объяснять материал, выполнять рутинные операции. Но польза зависит от того, как именно система встроена в работу человека.

У искусственного интеллекта нет человеческой ответственности. Ответственность несут люди и организации, которые выбирают данные, настраивают систему, разрешают действия и принимают решения о применении. Поэтому важны прозрачность, проверка, защита данных и возможность отказаться от автоматического решения.

В учебном контексте ИИ стоит рассматривать как инструмент. Он помогает сформулировать план, найти ошибку, предложить пример, но ответственность за понимание темы остаётся за человеком. Чем сильнее инструмент, тем важнее умение задавать вопрос, проверять ответ и понимать границы применения.

Особенно наглядно это видно в программировании. В феврале 2025 года Андрей Карпатый предложил термин vibe coding («вайб-кодинг») для нового стиля работы: человек описывает желаемое поведение программы естественным языком, а код пишет LLM. В исходной формулировке Карпатый предлагал «полностью довериться вайбу» и забыть, что код вообще существует, то есть даже не читать написанное моделью. Для быстрых прототипов и одноразовых экспериментов такой стиль действительно работает, а сам термин быстро стал популярным: в ноябре 2025 года словарь Collins назвал vibe coding словом года.

Но чем серьёзнее задача, тем хуже работает сам код программы и тем важнее становятся постановка цели, разбиение задачи на шаги и проверка результата.

Пример: безопасный учебный агент

Допустим, школьник готовится к контрольной работе. Безопасный учебный агент может принять список тем, предложить план повторения, подобрать тренировочные вопросы и объяснить ошибки в решениях. Для такой задачи ему не нужен доступ к почте, банковским данным или чужим файлам. Это и есть минимум прав на практике.

Если агент умеет записывать события в календарь, это уже действие во внешней среде. Значит, перед созданием события он должен показать черновик и получить подтверждение. Если он предлагает отправить письмо учителю, он должен показать адрес, тему и текст письма до отправки. Чем заметнее последствия действия, тем важнее явное согласие пользователя.

Такой пример показывает общий порядок проектирования агентных систем: сначала определить полезную цель, затем дать только необходимые инструменты, затем ограничить опасные действия и вести журнал выполненных шагов.

Полезно и ещё одно соображение. Решая, действовать ли самому, хороший агент взвешивает не только самый плохой исход, но и то, насколько каждый исход вероятен и насколько он тяжёл. Поэтому безопасный агент иногда останавливается и спрашивает человека, даже если так получается медленнее: маленькая вероятность тяжёлой потери перевешивает удобство быстрого действия.

Вопросы и задания

1. Чем современный ИИ-агент отличается от обычного чат-бота?
2. Из каких шагов состоит цикл работы агента?
3. Какие инструменты могут быть подключены к агенту и почему их набор ограничивают?
4. Что означает принцип «человек в контуре» и зачем он нужен?
5. Как оценивают риск действия $R(a)$ и что показывают его две части?
6. Что такое инъекция промпта? Почему внешний текст нельзя считать командой?
7. Почему агент с доступом к файлам или почте опаснее обычной текстовой модели?
8. Кто несёт ответственность за решения и действия ИИ-системы?

Проект.

Опишите учебного ИИ-агента, который помогает подготовиться к контрольной. Укажите цель, доступные инструменты, разрешённые и запрещённые действия, моменты, где требуется подтверждение человека, и возможные риски.

Дополнительно выберите три возможных действия агента и для каждого качественно оцените риск R(a): где вероятность ошибки мала или велика и где последствия ошибки особенно серьёзны. Пользуйтесь таблицей из раздела о контроле действий, чтобы решить, может ли агент выполнить действие сам, нужен ли черновик с подтверждением или решение должен принимать только человек.