Краткое содержание гайда Anthropic по построению агентов

Мы работали с десятками команд, создающих агентов LLM в разных отраслях. Последовательно, наиболее успешные реализации используют простые, составные шаблоны, а не сложные фреймворки.

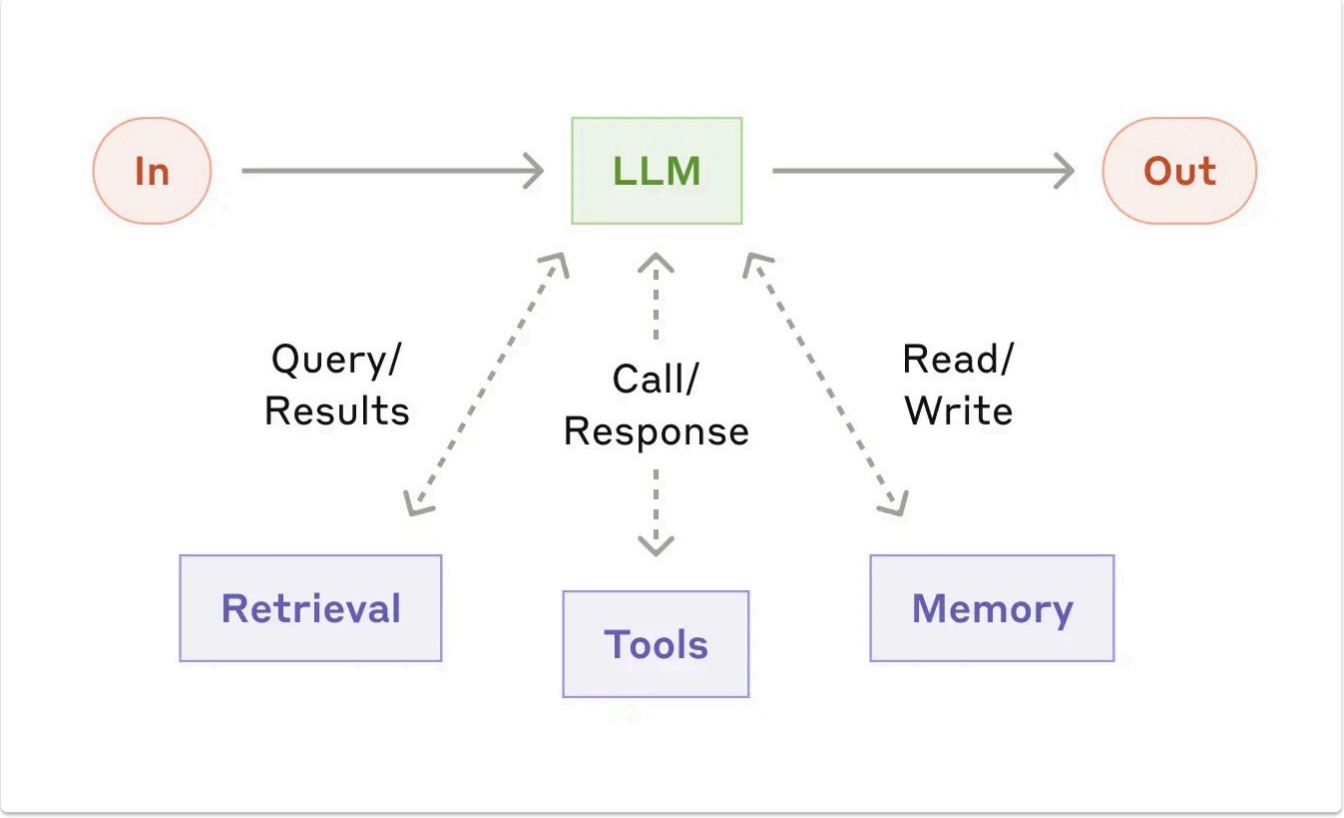
Подготовлено Дзен Директором

Основа

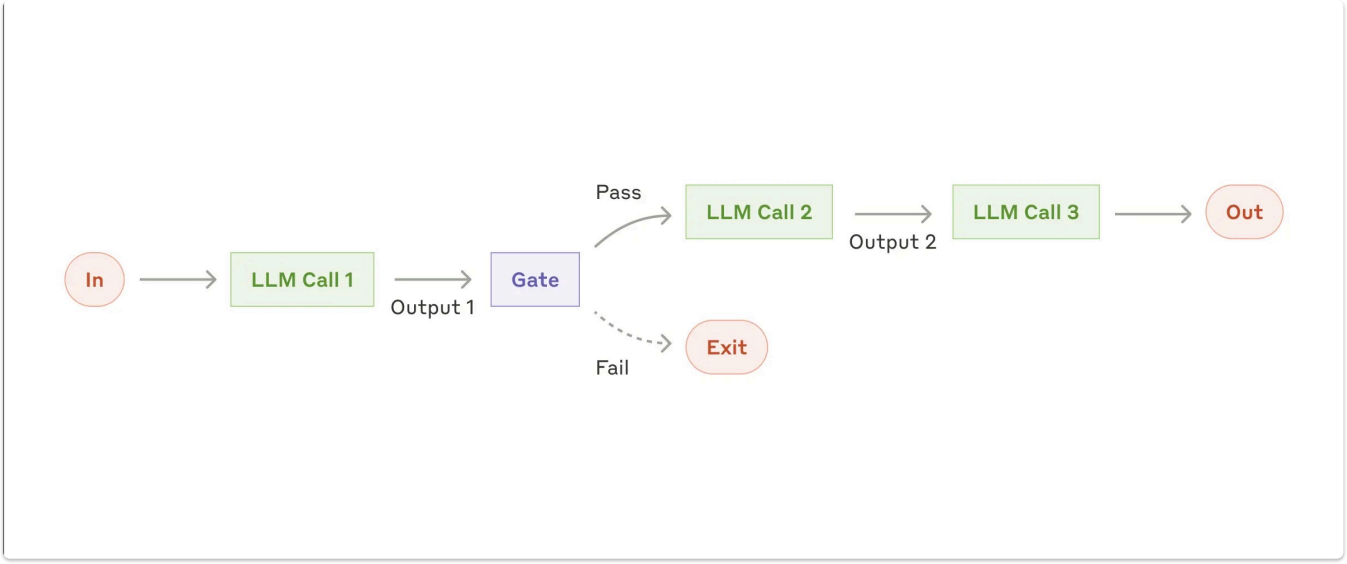
- **Что такое агенты?** Ключевая разница между автоматизированным процессом (workflow) и агентом в том, что первые отрабатывают по заранее определенному пути; а вторые — самостоятельно выбирают следующий шаг процесса.
- **Когда использовать агентов?** Когда нужна гибкость и принятие решений с использованием ЛЛМ, когда количество шагов и их последовательность заранее неизвестны. Если важнее предсказуемость и четкая последовательность — используйте простую автоматизацию.
- **Какие есть ограничения у агентов?** Они дороже в работе и более подвержены ошибкам.
- **Когда использовать фреймворки?** Фреймворки упрощают разработку, но добавляют абстрации и усложняют отладку; способствуют повышению сложности где она не нужна. Книга рецептов с примерами. Вот примеры таких фреймворков:
  - LangGraph от LangChain;
  - AI Agent framework от Amazon Bedrock;
  - Rivet, визуальный конструктор процессов (по аналогии с n8n);
  - Vellum, еще один инструмент с графическим интерфейсом для создания и тестирования сложных рабочих процессов.
- Успешные решения состоят из *простых* и *составных* шаблонов, а не сложных фреймворков или библиотек.

Компоненты

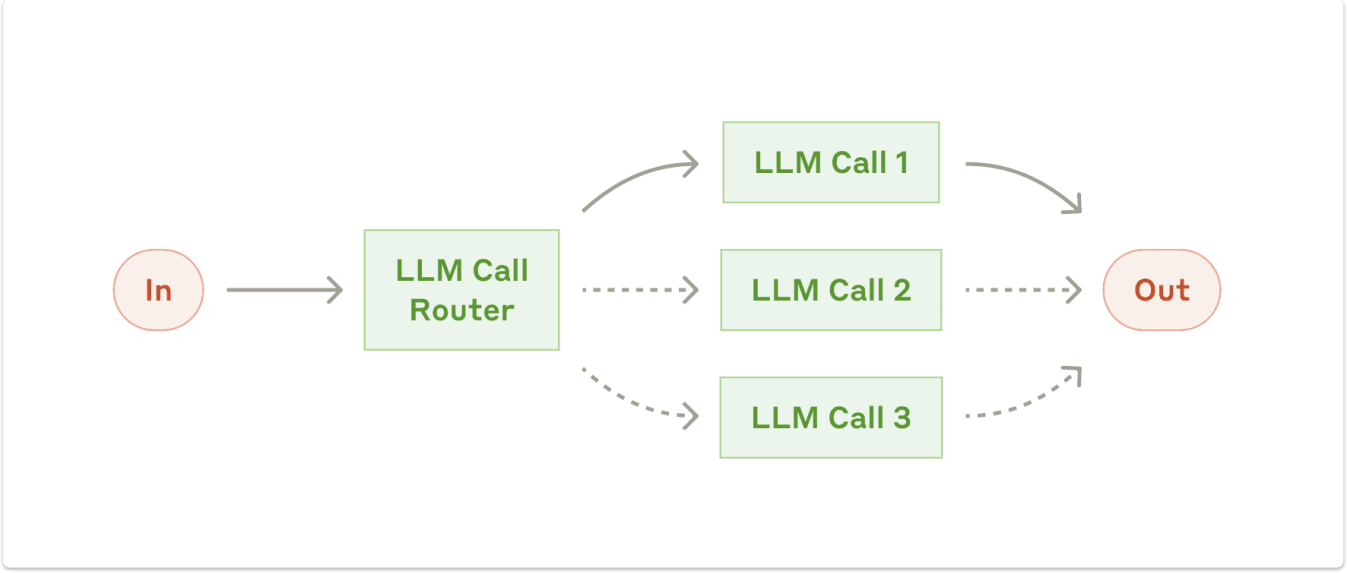
LLM. Основной компонент — ЛЛМ с «улучшениями» в виде поиска, инструментов и памяти. Для доступа к инструментам рекомендуется использовать MCP-протокол. Пример реализации.



Последовательность шагов — процесс при котором на каждом из которых выполняется какое-то действие: принимается решение или осуществляется проверка. Этот подход идеален, когда задача может быть декомпозирована на подзадачи. Основная цель: повысить точность за счет времени ожидания и упростить запросы в ЛЛМ.



Маршрутизация — подход позволяющий классифицировать вводные данные и затем перенаправить их на вызов специализированной функции. Без этого подхода оптимизация для одного вида входных данных может повредить производительности других входных данных.



Когда применять:

- Различные типы обращений в поддержке пользователей.
- Перенаправление простых вопросов на более дешевые модели для оптимизации цены.

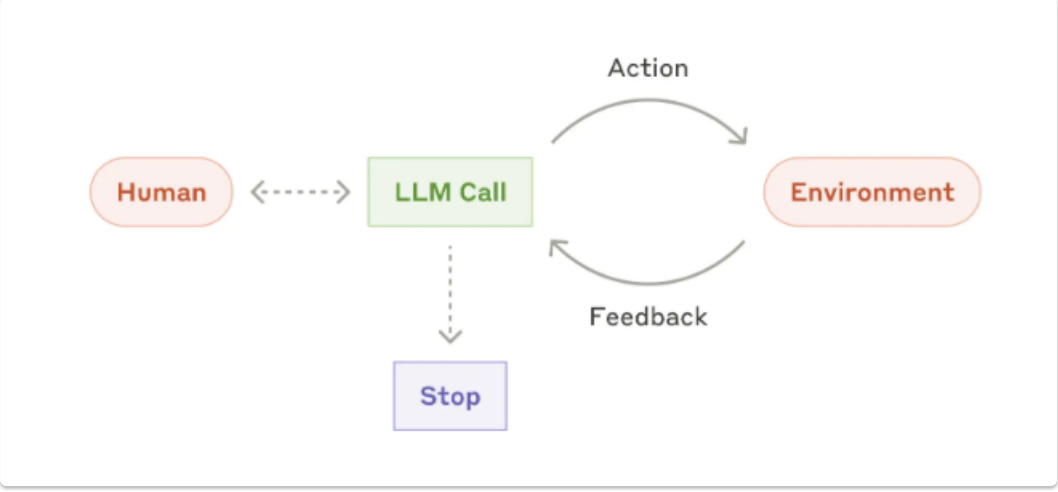
Параллелизация — подход при котором обработка задач процесса происходит параллельно. Бывает двух типов: параллельное выполнение подзадач и многократное выполнение для получения разных результатов. Когда работает:

- Для обеспечения безопасности, когда один запрос выполняет проверку ввода, а другой его обработку.
- Оценка качества выполнения задачи — каждый запрос проверяет один критерий, либо каждый запрос выполняет проверку по-разному для повышения надежности.
- Поиск баланса между ложно-позитивными и ложно-негативными выводами при проверке.

Оркестратор — разделение задач на подзадачи и последующий синтез результатов. Ключевым отличием от параллелизации является ее гибкость — подзадачи не определены заранее, а определяются оркестратором на основе конкретного ввода.

Оценщик-оптимизатор — подход при котором один запрос предоставляет результат, а другой его оценивает и предоставляет обратную связь. Пример применения: перевод текстов, поисковые запросы, где нужно постоянно оценивать, достигнут результат или нет.

Агенты



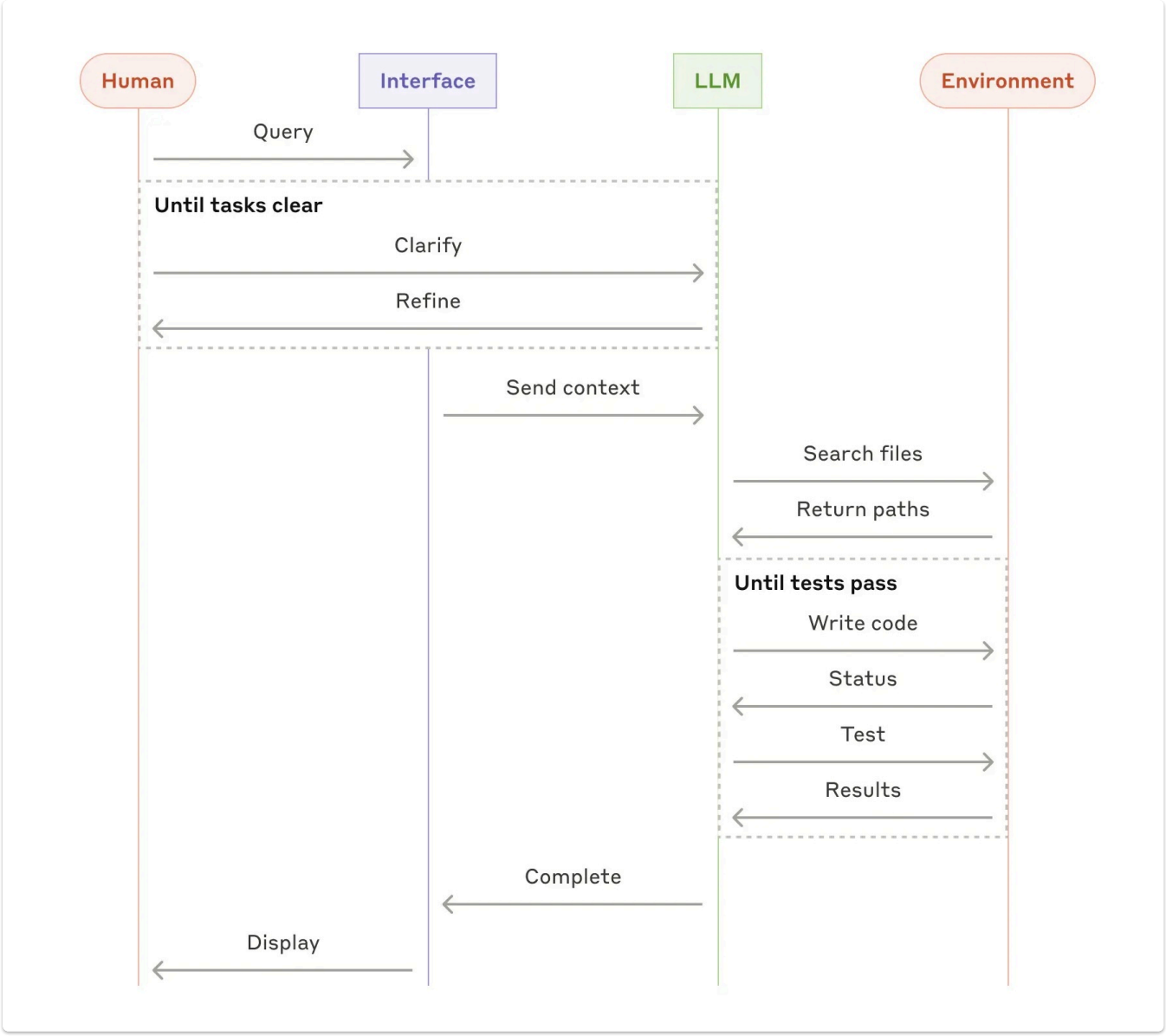
- Агенты получают задачу от человека, могут возвращаться к нему за дополнительной информацией.
- Важно, чтобы агент мог оценивать свой прогресс на основании доступа к фактам из окружения.
- Выполнение задачи может заканчиваться не только результатом, но и по условию. Например, когда сделано слишком много шагов.
- Агенты — это как правило системы с простыми правилами, работающие в цикле и опирающиеся на инструменты. Поэтому инструменты так важны.
- Ключом к успеху, как и в случае с любыми функциями LLM, является измерение производительности и итерация реализаций. Повторяю: вам следует подумать о добавлении сложности только тогда, когда это явно улучшает результаты.

Примеры агентов

Следующие примеры взяты из наших собственных реализаций:

- Агент кодирования для решения задач SWE-bench, которые включают в себя редактирование множества файлов на основе описания задачи;
- Наша эталонная реализация «использование компьютера», где Клод использует компьютер для выполнения задач.

Схема агента-разработчика



Выводы

Успех в работе с ЛЛМ заключается не в создании самой сложной системы. Речь идет о создании правильной системы для ваших нужд. Начните с простых промтов, оптимизируйте их с помощью комплексной оценки и добавляйте многошаговые агентские системы только тогда, когда более простые решения не работают.

При внедрении агентов мы стараемся следовать трем основным принципам:

- Поддерживайте простоту в дизайне вашего агента.
- Отдайте приоритет прозрачности, явно показав шаги планирования агента.
- Аккуратно работайте над интерфейсом агент-компьютер (ACI) используя тщательное документирование.

Фреймворки могут помочь вам быстро начать работу, но не стесняйтесь сокращать слои абстракции и работать с базовыми компонентами по мере перехода к эксплуатации. Следуя этим принципам, вы можете создавать агентов, которые являются не только мощными, но и надежными, обслуживаемыми и заслуживают доверие своих пользователей.