КОНСПЕКТ-ОПОРА ДЛЯ ОБСУЖДЕНИЯ

Модуль 5 - Агенты

Как устроены агенты: модель, обвязка, инструменты, контекст и границы

Главная рамка

Агент - это не магическая сущность и не автономный сотрудник. Это LLM, помещенная в цикл работы, обвязанная инструментами, памятью, правилами и ограничениями. Практическая ценность появляется не от слова "агент", а от правильно выбранной задачи, контекста, навыков и проверки результата.

Спикер: Юрий Агеев

Модуль: 5

Тема: Агенты как практическая сборка: от минимального цикла до рабочего помощника с контекстом, навыками и инструментами.

Источник: Транскрипт лекции m5-transcript.srt

Документ сделан как опора для разговора: не линейный пересказ, а рамка, тезисы, спорные места и практические проверки.

Главный вопрос лекции

Вопрос

Как перестать воспринимать агентов как хайп или магию и понять, из каких частей они состоят, где реально усиливают работу, а где требуют жестких границ, контекста и валидации?

Организующая рамка

Рабочая формула модуля: агент = модель + обвязка + инструменты + контекст/память + границы.

Слой	Зачем нужен
Модель / "мозг"	LLM не принимает решения в человеческом смысле. Она продолжает текст и выбирает следующий вероятный шаг внутри заданной ситуации.
Обвязка / harness	Цикл, который хранит сообщения, отправляет их модели, вызывает инструменты, ведет сессию и спрашивает разрешения.
Инструменты / MCP	Чтение и запись файлов, интернет, API, почта, GitHub, Figma и другие внешние действия. Именно они превращают чат в рабочую систему.
Контекст / экзокортекс	Папка с файлами, CLAUDE.md / AGENTS.md, шаблоны, навыки, заметки и правила, которые придают форму поведению агента.
Границы / безопасность	Разрешения, песочница, режимы read-only/full access, хуки, лимиты, валидация, подтверждения человека.

Связка с предыдущими модулями

M1 дает границы и контракты результата. M2 напоминает про доверие и проверку. M3 дает контекст. M4 дает workflows. M5 собирает это в делегирование: агент получает задачу, контекст, инструменты и ограничения.

Ключевые тезисы

1. Агент - это не "живой интеллект", а LLM в рабочем цикле.

- **Почему важно:** Снимает лишнюю мистику и страх: можно разбирать агент как инженерную сборку, а не как черный ящик с волей.
- **Где работает / где не работает:** Работает как базовое определение для практики. Не работает, если требовать от агента человеческой мотивации, самостоятельного понимания целей и устойчивого здравого смысла.
- **На чем держится:** Лекция начинается с различения workflow и агента: в workflow порядок шагов задает человек, в агенте модель выбирает следующий шаг в процессе работы.

2. Магия агента находится не в модели, а в обвязке.

- **Почему важно:** Одна и та же модель может быть бесполезным чатом или рабочим помощником в зависимости от цикла, инструментов, памяти, разрешений и правил.
- **Где работает / где не работает:** Работает в инструментах вроде Claude Code / Codex, где уже встроены контракты, запросы разрешений и работа с файлами. Не работает в "голом" чате без доступа к среде.
- **На чем держится:** В демонстрации минимальный агент сначала был простым чат-циклом, затем получил инструмент чтения файлов и начал использовать контекст.

3. Границы важнее энтузиазма: агент без ограничений быстро становится рискованным.

- **Почему важно:** Как только агент получает запись в файлы, интернет, почту или API, ошибка перестает быть только текстовой - она может стать действием.
- **Где работает / где не работает:** Работает там, где заданы разрешения, бэкапы, read-only режимы, подтверждения и отдельные запреты на деструктивные команды. Не работает, если все держится только на просьбе "будь аккуратен".
- **На чем держится:** При попытке добавить редактирование файлов минимальный агент начал ломаться и галлюцинировать; затем обсуждались permissions, sandbox, hooks и ограничения MCP.

4. Контекст превращает универсальную модель в помощника под конкретную работу.

- **Почему важно:** Без устойчивого контекста агент каждый раз заново угадывает, кто вы, что важно, какие правила соблюдать и как выглядит хороший результат.
- **Где работает / где не работает:** Работает через папку-экзокортекс, CLAUDE.md / AGENTS.md, шаблоны, навыки и файлы проекта. Не работает, если контекст слишком общий, устаревший, раздутый или противоречивый.
- **На чем держится:** В лекции агент настраивался через интервью о продукте, аудитории, конкурентах и задачах, а затем использовал этот контекст в тестовых прогонах.

5. Навыки и шаблоны - это способ делегировать повторяемую работу, а не переписывать один и тот же промпт.

- **Почему важно:** Они позволяют собрать reusable-единицу работы: триггер, описание, входы, разрешенные инструменты, формат результата и валидацию.
- **Где работает / где не работает:** Работает для meeting notes, YouTube summary, competitor scan, setup, регулярных аналитических обзоров. Не работает для разовых задач с высокой неопределенностью, где еще непонятна сама рамка.
- **На чем держится:** В лекции разбирались setup-навык, шаблоны заметок, конкурентный обзор и сохранение результатов в нужные папки.

6. Автономность - отдельная проблема, а не обязательное свойство полезного агента.

- **Почему важно:** Можно получить много пользы от агента, который действует полуавтономно: спрашивает разрешение, читает файлы, готовит черновики и просит валидацию.
- **Где работает / где не работает:** Работает для делегирования под присмотром. Не работает как безопасная стратегия, если сразу отдавать агенту долгие автономные задачи с записью, удалением и внешними API.
- **На чем держится:** Лекция заканчивается мостиком к М6: как отпускать агента в более свободное плавание и какие защитные механизмы для этого нужны.

Ключевые идеи для обсуждения

1. Агент как "модель + цикл", а не как новая форма разума

Суть: Агент производит впечатление самостоятельности, потому что сам выбирает следующий шаг внутри задачи. Но в основании остается та же механика LLM: продолжение контекста токен за токеном. Поэтому полезнее смотреть не на "личность" агента, а на то, какой цикл, контекст и инструменты вокруг модели собраны.

Почему это важно: Это защищает от двух крайностей: паники "все уже автоматизировано" и наивной веры "агент сам разберется".

Пример / кейс из лекции: В начале лекции агент противопоставляется workflow: если шаги заранее задает человек, это workflow; если следующий шаг выбирает модель внутри цикла, мы уже ближе к агенту.

Где не сработает / ограничение: Если задача требует строгого маршрута, проверяемого процесса и минимальной вариативности, workflow может быть лучше агента.

Вопрос для обсуждения: В каких ваших задачах действительно нужно, чтобы модель сама выбирала следующий шаг, а где достаточно хорошо описанного workflow?

2. Минимальный агент показывает и силу, и хрупкость подхода

Суть: Простой агент может быть очень маленьким: хранить переписку, отправлять ее в API модели и получать ответ. Когда добавляется инструмент чтения файлов, агент уже получает зачаток рабочего контекста. Но как только появляется редактирование, запись и самоизменение, хрупкость резко возрастает.

Почему это важно: Демонстрация полезна именно потому, что снимает магию: агент не загадочен, но сырой агент ненадежен.

Пример / кейс из лекции: В лекции минимальный агент сначала работал как чат, затем получил чтение файлов; попытка заставить его спроектировать безопасное редактирование показала сбои и галлюцинации.

Где не сработает / ограничение: Минимальный агент хорош как учебная модель, но плох как рабочий инструмент без бэкапов, ограничений, валидации и нормальной обвязки.

Вопрос для обсуждения: Что в вашей работе было бы опасно отдавать "минимальному агенту" даже на эксперименте?

3. Обвязка превращает модель в рабочую систему

Суть: Claude Code, Codex и похожие инструменты - это не просто "модель умнее". Это mature harness: файлы, сессии, разрешения, режимы доступа, вызовы инструментов, структура проекта, команды и ограничения. Большая часть надежности возникает именно там.

Почему это важно: Это меняет фокус настройки: не только промпт, а среда, правила, доступы и артефакты.

Пример / кейс из лекции: После демонстрации минимального агента лекция переходит к тому, что готовые агентные инструменты уже прошли путь ошибок и встроили часть контрактов в продукт.

Где не сработает / ограничение: Обвязка не отменяет проверку: текстовые правила могут размываться, контекст может загрязняться, а внешние инструменты могут быть опасны.

Вопрос для обсуждения: Какие правила у вас сейчас записаны как "пожелания", но должны быть встроены в обвязку или процесс?

4. Инструменты и MCP расширяют возможности агента - и площадь риска

Суть: Пока агент только пишет текст, ошибка остается текстом. Когда он читает почту, ходит в интернет, пишет файлы или дергает API, он начинает действовать во внешнем мире. MCP удобен как универсальный разъем, но каждый новый разъем добавляет и полезность, и угрозы.

Почему это важно: Это помогает проектировать доступы: начинать с чтения, затем ограниченная запись, затем подтверждения, затем хуки и внешние проверки.

Пример / кейс из лекции: В лекции MCP объясняется как "USB для модели"; демонстрируется подключение почты с запретом на отправку и удаление писем.

Где не сработает / ограничение: MCP-сервер с сотнями методов может разнести контекст и увеличить риск ошибки; почта и API требуют защиты от prompt injection и деструктивных действий.

Вопрос для обсуждения: Какие три инструмента дали бы вашему агенту 80% пользы, и какие действия он не должен уметь делать без человека?

5. Папка-экзокортекс: агенту нужен устойчивый рабочий контекст

Суть: Папка с файлами, шаблонами и правилами становится внешней памятью. Агент не "знает вас" автоматически; он считывает документы, инструкции и артефакты по мере работы. Чем точнее организован контекст, тем меньше приходится каждый раз объяснять заново.

Почему это важно: Это особенно важно для продуктовой и управленческой работы, где качество ответа зависит от контекста компании, аудитории, рынка, договоренностей и критериев.

Пример / кейс из лекции: В лекции используется Obsidian-папка с README, templates, inbox, meeting notes, импортированными материалами и инструкциями для агента.

Где не сработает / ограничение: Агент не читает все файлы сам по себе. Он должен иметь повод вызвать чтение. Слишком большой или противоречивый контекст снижает следование правилам.

Вопрос для обсуждения: Какие 5-7 документов должны лежать в вашем рабочем экзокортексе, чтобы агент перестал отвечать "вообще"?

6. CLAUDE.md / AGENTS.md задают "личность" и рабочие правила

Суть: Постоянный файл инструкций - это не просто справка. Он каждый раз формирует поведение агента: роль, карту папок, порядок чтения, правила цитирования, допустимые действия, fallback и запреты. Это похоже на операционный контракт помощника.

Почему это важно: Так можно убрать часть повторяющихся объяснений из промпта и сделать поведение стабильнее между сессиями.

Пример / кейс из лекции: В лекции меняется описание роли агента, и он начинает вести себя иначе; отдельно обсуждается ограничение на размер постоянных инструкций.

Где не сработает / ограничение: Это все еще текстовая инструкция. Ее нельзя считать надежной защитой от опасных действий; важные запреты должны дублироваться техническими механизмами.

Вопрос для обсуждения: Что должно быть в постоянных правилах агента, а что нельзя оставлять только в тексте?

7. Навыки - это маленькие рабочие протоколы

Суть: Skill описывает, когда его вызывать, какие инструменты разрешены, какой формат результата нужен и какие шаги выполнить. В отличие от разового промпта, навык можно запускать снова и снова, накапливая качество процесса.

Почему это важно: Это переводит работу из "попросить модель красиво" в "дать ей повторяемый протокол".

Пример / кейс из лекции: В лекции показываются setup-навык, meeting notes, YouTube summary, competitor scan и шаблоны сохранения заметок в нужные папки.

Где не сработает / ограничение: Навык не гарантирует истинность результата. Он задает маршрут, но факты, ссылки, гипотезы и выводы все равно нужно проверять.

Вопрос для обсуждения: Какая одна повторяемая задача в вашей работе заслуживает отдельного skill уже сейчас?

8. Память требует гигиены: не все надо тащить в одну сессию

Суть: Длинная сессия раздувает контекст, снижает управляемость и увеличивает стоимость. Иногда можно продолжить сессию по resume hash, но часто лучше сохранить важные артефакты в файлы и начать свежий контекст.

Почему это важно: Это помогает не путать "память" с мусорным баком. Хорошая память - это отобранные артефакты, а не весь поток разговора.

Пример / кейс из лекции: В лекции разбирается resume, compact и идея сохранять важные файлы в Obsidian или другую рабочую папку.

Где не сработает / ограничение: Compact - не точная память, а сжатие с потерями. Его можно направлять, но нельзя считать надежным архивом.

Вопрос для обсуждения: Когда в ваших задачах лучше продолжать сессию, а когда лучше закрыть ее и оставить только артефакты?

9. Задачи для агента находятся через практику, а не через абстрактный список идей

Суть: Самая сложная часть - не запустить агента, а увидеть, какую работу ему действительно можно делегировать. Часто сначала человек чинит самого агента, а потом постепенно замечает повторяющиеся участки: редактура, сбор цифр, проверка источников, подготовка заметок, конкурентный обзор.

Почему это важно: Это снижает завышенные ожидания: агент не обязан сразу заменить процесс, он может снимать маленькие регулярные куски нагрузки.

Пример / кейс из лекции: В вопросах после лекции обсуждается, что задачи для агента могут быть неочевидны; примером становятся редакционное ревью и маркетинговая аналитика по Метрике и Telegram.

Где не сработает / ограничение: На границе высокой неопределенности агент легко начинает правдоподобно придумывать. Там нужно спрашивать, каких данных ему не хватает, и валидировать факты.

Вопрос для обсуждения: Какая ваша регулярная "рутинная, но контекстная" задача стоит первой проверки на агенте?

Что здесь новое или спорное

- Агент может быть очень маленьким по коду. Сложность не в факте существования агента, а в надежной обвязке вокруг него.
- Провал демонстрации - не баг лекции, а часть смысла: сырой агент легко ломается, галлюцинирует и делает обходные предложения.
- Промпт не исчезает, но перестает быть главным уровнем управления. Важнее становятся папка, постоянные правила, навыки, инструменты и разрешения.
- Текстовые инструкции полезны, но слабы как защита. Для опасных действий нужны технические границы: read-only, подтверждения, хуки, запреты на команды, бэкапы.

- Больше контекста не всегда лучше. Раздутый контекст может ухудшать следование правилам и увеличивать стоимость.
- Агент не читает все файлы в папке автоматически. Он вызывает чтение, когда в его текущем ходе появляется причина.
- MCP выглядит как универсальный разъем, но универсальность увеличивает площадь атаки и может перегружать контекст сотнями доступных методов.

Вопросы для группы

- Где в вашей работе нужен именно агент, который выбирает следующий шаг, а где достаточно workflow с заранее заданным маршрутом?
- Какие действия агента можно оставить на текстовом уровне правил, а какие должны быть запрещены или ограничены технически?
- Какой минимальный рабочий контекст нужен агенту, чтобы он отвечал про вашу реальную работу, а не "вообще про рынок"?
- Какие повторяемые задачи стоит превратить в skill, чтобы перестать писать один и тот же промпт?
- Как вы поймете, что агент усилил работу, а не просто создал правдоподобный черновик, который все равно надо полностью переделывать?
- Какие данные агент должен запрашивать у вас, если не может уверенно решить задачу, вместо того чтобы додумывать?

Что попробовать на практике

Собрать папку-экзокортекс на одну задачу.

- **Кейс:** одна регулярная задача - например, обзор конкурентов, подготовка заметки по встрече, ревью текста или сбор цифр по каналу.
- **Проверка:** агент использует нужные файлы, задает вопросы о пробелах, отделяет факты от гипотез и выдает результат в заданном формате.

Превратить один повторяемый промпт в skill.

- **Кейс:** взять шаблон, который вы уже используете 3+ раза, и описать триггер, входы, разрешенные инструменты, формат результата и обязательную проверку.
- **Проверка:** один и тот же skill можно применить к трем разным материалам без переписывания инструкции с нуля.

Сделать карту разрешений агента.

- **Кейс:** выписать инструменты, которые агенту нужны: чтение файлов, запись файлов, интернет, почта, API, терминал. Для каждого определить read-only / ask / deny.
- **Проверка:** для деструктивных действий есть подтверждение, бэкап или внешний запрет, а не только фраза в инструкции.

Поймать границу неопределенности.

- **Кейс:** дать агенту задачу, где он может начать правдоподобно додумывать: рыночное сравнение, бизнес-модель, вывод из неполных данных.
- **Проверка:** агент явно пишет, каких данных не хватает, какие выводы являются гипотезами и что нужно проверить человеком.

Открытые вопросы

- Какие защитные механизмы должны быть вне модели, чтобы агент мог безопасно работать дольше и автономнее?
- Как измерять реальную экономику агента: время, качество, стоимость токенов, стоимость проверки и риск ошибки?
- Где проходит граница между "хорошим делегированием" и "я теперь обслуживаю своего агента"?
- Какие типы продуктовых и управленческих задач остаются слишком неопределенными для агентного режима без плотного диалога с человеком?

- Как проектировать память так, чтобы она помогала, но не превращалась в накопление шума?
- Как переносить навыки и контекст между разными агентными средами, если стандарты CLAUDE.md / AGENTS.md / skills различаются?

Короткий итог для обсуждения

M5 предлагает не верить в агентов и не бояться их, а раскладывать на слои: что выбирает модель, какие инструменты ей доступны, какой контекст она видит, какие навыки запускает, где стоят границы и как проверяется результат. Полезный агент - это не "умная модель сама все сделала", а собранная рабочая система делегирования.

Подготовлено как учебный конспект-опора: тезисы, рамки, вопросы и проверки вместо линейного пересказа.