КУРС «СИГНАЛ» · МОДУЛЬ 4

# Workflow

## Как автоматизировать повторяющуюся работу с LLM, не превращая процесс в неуправляемого автономного агента

**Формат:** конспект-опора для обсуждения, а не линейный пересказ лекции.
**Основание:** транскрипт живого модуля от 30 апреля 2026.
**Назначение:** помочь участникам спорить, переносить идеи в свои процессы и формулировать первые workflow-кандидаты.

Нейтральная версия без рекламной вставки и без дословного транскрипта.

# Главный вопрос лекции

## Как понять, где LLM должна быть частью предсказуемого рабочего процесса, а где попытка «дать ИИ свободу» превращает задачу в рискованный агентский эксперимент?

Главный сдвиг модуля: мы смотрим не на «магического помощника», а на конструкцию из входа, шагов, проверок и результата. В этой конструкции LLM может быть мощным компонентом, но не обязана управлять всем процессом.

**Рабочая формула модуля:** вход + шаги + гейты + готово. Если хотя бы одна часть не определена, это еще не workflow, а надежда на то, что модель сама разберется.

| Task | Workflow | Agent | Skill |
|---|---|---|---|
| Один запрос, один ответ, обычно внутри чата. | Порядок шагов заранее задан человеком или кодом. | Модель сама выбирает следующий шаг на ходу. | Переиспользуемая инструкция, папка с правилами, примерами и иногда скриптами. |

# Ключевые тезисы

1. **Workflow - это не длинный чат.** Большой промпт и длинная переписка не дают внешних проверок. Модель может забыть часть требований, галлюцинировать выполнение или подстроиться под контекст.
2. **Главное различие - кто задает порядок.** В workflow порядок предопределен; в агенте порядок выбирает модель. Это разные режимы доверия и разные риски.
3. **LLM лучше зажимать между твердыми гранями.** Код, маршрутизация, ограничения инструментов и проверки удерживают результат лучше, чем просьба «будь точной».
4. **Гейт превращает «кажется сделал» в проверяемое «готово».** Без гейта модель может убедительно отчитаться о выполнении, хотя комментарии не исправлены или файл не загружен.
5. **Skill полезен, но не заменяет workflow.** Скилл помогает повторять качество и не писать промпт заново, но жесткий контроль появляется только через внешнюю обвязку и проверки.
6. **Хороший кандидат на workflow должен окупаться.** Нужны повторяемость, декомпозиция, критерий результата, приемлемая цена ошибки и экономический смысл.
7. **Человек не исчезает, а меняет роль.** Он часто становится консультантом или контролером: задает рамку, принимает результат и отвечает за последствия.

# Ключевые идеи для разбора

### 1. Пять слов, которые нельзя смешивать

**Суть:** LLM, tool, task, workflow, agent и skill часто звучат как взаимозаменяемые слова, но описывают разные уровни системы. LLM отвечает и рассуждает, tools дают ей действия, task живет в чате, workflow задает цепочку шагов, agent сам выбирает следующий ход, skill подгружается как повторяемый способ действия.
**Ограничение:** границы не всегда чистые: в реальных продуктах agent может запускать workflows, а workflow может содержать маленький agent loop.
**Вопрос:** какие из этих слов сейчас у вас в команде используются слишком широко и мешают договориться?

### 2. Workflow начинается с известной последовательности

**Суть:** workflow можно собрать, когда человек уже понимает, какие шаги обычно приводят к хорошему результату. Если последовательность неизвестна, вы не автоматизируете процесс, а просите модель придумать процесс за вас.
**Пример:** встреча команды, подготовка описаний спикеров, разметка подкаста, дайджест чата - повторяющиеся задачи с похожей структурой, но меняющимися входными данными.
**Вопрос:** где у вас задача кажется «творческой» только потому, что ее еще никто не разложил на шаги?

### 3. Гейт - это граница доверия

**Суть:** гейт останавливает процесс, если результат не прошел проверку. Это может быть программа, человек, сравнение файлов, проверка формата, наличие обязательных блоков, сверка комментариев или другой внешний критерий.
**Пример:** в подготовке спикера Олег сначала уверенно закрывал комментарии, которые считал выполненными, но фактически не всегда исправлял. После этого появился программный барьер: сравнить изменения и закрывать только подтвержденное.
**Вопрос:** какой один гейт сильнее всего снизит риск в вашем первом workflow?

### 4. Тупая маршрутизация может быть умнее «умной» LLM

**Суть:** на входе в workflow не всегда нужно ставить модель. Иногда лучше договориться о командах: «дайджест», «правки», «англицизмы», «подкаст» - и распознавать их программно.
**Почему это важно:** человеческий язык неоднозначен. Если нужна предсказуемость, лучше меньше просить модель «понять намерение» и больше задавать жесткие входные формы.
**Вопрос:** где ваша команда пытается решить маршрутизацию промптом, хотя хватило бы явной команды или кнопки?

### 5. Skill - это способ упаковать повторяемое мастерство

**Суть:** для LLM skill - это не «навык человека», а папка с инструкцией, примерами, правилами и иногда скриптами. Он подгружается по необходимости и позволяет не пересобирать сложный промпт каждый раз.
**Ограничение:** чем объемнее skill, тем выше риск контекстного шума и забытых требований. Скиллы надо дорабатывать на реальных прогонах, а не считать завершенными после первого текста.
**Вопрос:** что в вашем повторяемом процессе стоит вынести в skill, а что должно остаться во внешнем коде?

# Кейсы из лекции: что обсуждать

### Кейс 1. Олег как «почти агент»

**Снаружи:** бот сидит в Telegram, ему пишут, он отвечает, значит выглядит как агент.
**Внутри:** большая часть поведения задается заранее: команды распознаются, данные скачиваются, LLM вызывается в нужном месте, результат проверяется и только потом публикуется.
**Обсуждение:** почему интерфейс чата создает иллюзию агентности даже там, где на самом деле работает workflow?

### Кейс 2. Дайджест редакционного чата

| Шаг | Что происходит | Где риск |
|---|---|---|
| Команда | Пользователь просит Олега сделать дайджест за период. | Неверно распознать намерение или даты. |
| Сбор данных | Бот берет сообщения из базы. | Пустой или неполный диапазон. |
| LLM | Группирует темы, решения, ссылки, открытые вопросы. | Добавить то, чего в чате не было. |
| Гейт | Проверяет базовую структуру и пригодность результата. | Слабый гейт не ловит смысловые галлюцинации. |
| Публикация | Результат уходит обратно в чат. | Пользователи могут принять уверенный текст за факт. |

**Ключевой эпизод:** Олег однажды сообщил о запланированной рассылке, которой не было. Это хороший пример того, почему «гладкий дайджест» не равен «верный дайджест».

### Кейс 3. Подготовка спикера

**Схема:** сообщение «подготовь спикера» + ссылка на Yonote -> маршрутизатор -> скачивание документа и комментариев -> headless LLM со skill -> запись результата -> программное сравнение изменений -> загрузка обратно и закрытие подтвержденных комментариев.
**Смысл кейса:** LLM делает сложную редакторскую часть, но не отвечает за весь процесс. Скачивание, сравнение, загрузка и закрытие комментариев лучше держать вне модели.
**Обсуждение:** какие части этой цепочки можно доверить модели, а какие должны оставаться механическими?

### Кейс 4. Разметка подкаста

**Схема:** ссылка на Dropbox -> скачивание аудио -> локальная транскрибация через Whisperflow -> skill с шаблоном выпуска -> черновик описания, названий, упоминаний и справки -> редакторское чтение.
**Смысл кейса:** большая часть времени уходит не на «думание модели», а на обвязку: скачать, распознать, подготовить вход, сохранить результат. LLM появляется внутри процесса, а не вместо процесса.
**Обсуждение:** где в вашей работе «умная» часть занимает 20%, а 80% - это механическая подготовка входа и выпуск результата?

### Кейс 5. Live-demo: YouTube summary

**Схема:** Obsidian + Code -> установка инструмента для субтитров -> создание skill -> скачать не видео, а субтитры -> выделить неординарные идеи, применимые действия, вопросы для размышления и тайм-коды -> переиспользовать skill на другом видео и даже в другом экземпляре Code.
**Смысл кейса:** «консольная магия» оказывается не магией, а повторяемой процедурой, которую можно закрепить в skill и гонять снова.

# Паттерны workflow

В лекции они нужны не как терминологический справочник, а как набор ментальных шаблонов: какой тип процесса я вообще пытаюсь собрать?

| Паттерн | Когда подходит | Пример из модуля |
|---|---|---|
| Цепочка шагов | Есть последовательность: взять вход, преобразовать, проверить, выдать результат. | Подготовка спикера; разметка подкаста. |
| Маршрутизация | Один входной канал, много разных сценариев обработки. | Олег распознает «дайджест», «правки», «подкаст», «англицизмы» и отправляет запрос в нужный skill. |
| Параллельная работа | Можно выполнить несколько независимых действий и потом собрать общий результат. | Скачать документ и комментарии; параллельно обработать части материалов. |
| Главный и помощники | Один компонент распределяет задачи между несколькими исполнителями и собирает итог. | Похоже на agent/subagent-архитектуры, но порядок может быть задан заранее. |
| Проверка и доработка | Нужен цикл: результат -> оценка -> исправление -> повторная проверка. | Редакторы ругаются с Олегом, правятся skills и глоссарии, качество постепенно стабилизируется. |

# Критерии: когда задача годится для workflow

- **Повторяемость.** Задача возвращается регулярно. Если она разовая, автоматизация может не окупиться.
- **Декомпозируемость.** Ее можно разложить на понятные шаги. Если шаги неизвестны, сначала нужен анализ процесса.
- **Критерий «готово».** Есть понятный выход: файл, список, описание, дайджест, проверенный документ, набор тайм-кодов.
- **Предсказуемость качества.** Можно хотя бы частично проверить результат: формат, полноту, соответствие правилам, наличие обязательных блоков.
- **Цена ошибки.** Ошибка не должна сразу разрушать процесс. Риск нужно хеджировать человеком, версионированием или гейтом.
- **Экономический смысл.** Автоматизация должна освобождать заметное время или снижать операционную боль.

## Что здесь новое или спорное

- **Workflow не равен промпту.** Даже очень подробный промпт остается рекомендацией для модели, а не жесткой последовательностью действий.
- **Иногда лучший вход в AI-систему - не естественный язык, а жесткая команда.** «Напиши как хочешь» удобнее человеку, но хуже для предсказуемости.
- **Чтобы автоматизировать, сначала надо уметь делать самому.** Иначе нечего формализовать: нет шагов, нет гейта, нет определения «готово».
- **Творческую работу можно частично автоматизировать.** Если она годами повторяется и имеет устойчивый формат, творческая часть может оказаться упакованной в skill и проверки.
- **Полная автономность не всегда прогресс.** Для бизнеса ценнее предсказуемый результат, чем красивое обещание «агент сам все сделает».
- **Снаружи система может выглядеть как агент, а внутри быть workflow.** Это не плохо. Плохо - не понимать, где именно проходит граница свободы модели.

## С чем слушатель, вероятно, поспорит

- **«Я не хочу писать схемы, пусть агент сам разберется».** Тогда вы выбираете не workflow, а агентский режим с другой ценой ошибки.
- **«Скилл же уже описывает порядок, значит это workflow».** Порядок в тексте - это инструкция модели; workflow появляется, когда часть порядка исполняется внешней системой.
- **«Регекс и команды выглядят примитивно».** Примитивность на входе может быть ценой надежности.
- **«Если все равно нужен человек, в чем автоматизация?»** Человек переходит из режима исполнителя в режим ревьюера/контролера и тратит время на другие действия.

## Вопросы для группы

1. Какая повторяющаяся задача в вашей работе уже имеет понятный вход, шаги и результат, но до сих пор делается руками?
2. Где в этой задаче LLM должна получить свободу, а где свободу лучше забрать кодом, шаблоном, кнопкой или проверкой?
3. Какой минимальный гейт отделит «модель красиво ответила» от «результат действительно можно использовать»?
4. Что дешевле и надежнее на входе: попросить модель понять намерение или договориться с командой о явной команде/формате?
5. Какая ошибка в вашем workflow будет приемлемой, а какая должна останавливать процесс?
6. Как изменится роль человека: оператор, соавтор, консультант, контролер или наблюдатель?
7. Какие части вашей работы выглядят творческими, но на самом деле повторяются по одному и тому же ритуалу?
8. Где вы сейчас называете систему «агентом», хотя на практике вам нужен более жесткий workflow?

---

**Мини-диагностика:** если у задачи нет хотя бы трех признаков - повторяемость, декомпозиция, критерий «готово», проверяемое качество, приемлемая цена ошибки, экономический смысл - workflow пока, скорее всего, не нужен.

## Что попробовать после лекции

1. **Найти одного кандидата на workflow.** Выберите задачу, которую вы делали минимум 5-10 раз: дайджест, письмо, подготовка описания, проверка документа, разметка видео, обработка заявок, сбор статуса.
2. **Нарисовать схему.** Не писать сразу промпт. Сначала нарисовать: вход -> шаг 1 -> шаг 2 -> место для LLM -> гейт -> публикация/сохранение.
3. **Сформулировать «готово».** Запишите 3-5 проверяемых признаков результата: обязательные блоки, формат файла, отсутствие пустых разделов, список измененных комментариев, тайм-коды, ссылки, длина, язык.
4. **Собрать маленький skill.** Вынесите в skill конкретный повторяемый способ работы: входной формат, цель, правила, пример хорошего выхода, запреты, куда сохранить результат.
5. **Провести ручной прогон и записать сбои.** Запустите workflow на нескольких реальных примерах и соберите карту ошибок: где модель выдумывает, где забывает, где нужен гейт.

## Открытые вопросы

- Какие проверки можно сделать программными, а где все равно нужен человеческий смысловой ревью?
- Как измерять доверие к workflow после нескольких ручных прогонов?
- Когда длинный skill становится слишком шумным и начинает ухудшать результат?
- Как версионировать skills, глоссарии и правила, чтобы команда понимала, почему поведение изменилось?
- Где проходит граница между workflow с маленьким agent loop и настоящим агентом?
- Как тестировать регрессии, если модель, skill или внешний инструмент обновились?
- Какая роль человека оптимальна: консультант, контролер или наблюдатель?
- Как не превратить первый workflow в чрезмерно сложную инженерную систему?

## Короткое закрытие для обсуждения

**Workflow - это не способ дать ИИ больше свободы. Это способ забрать у модели лишнюю свободу там, где нужна повторяемость, и оставить ей только ту часть неопределенности, где она действительно усиливает работу.**

**Мост к следующему модулю.** Если workflow держится на заранее заданном порядке, то что меняется, когда порядок начинает выбирать агент? Для этого понадобятся понятия agent loop, harness/обвязка, ограничения инструментов и новые уровни контроля.