Увольнение, найм и обучение команд в эпоху ИИ-агентов

Николай Шейко

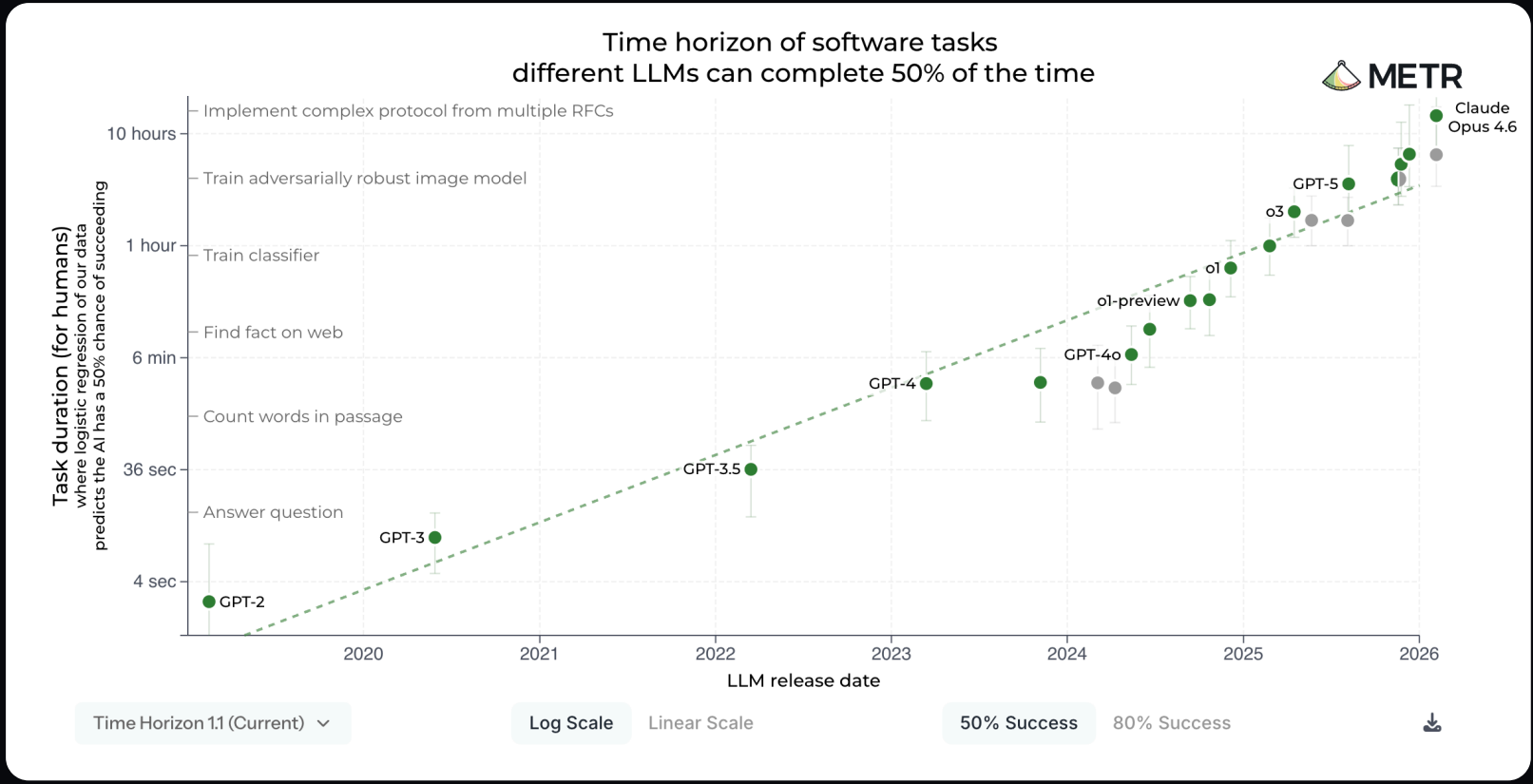
- Раньше AI Engineer в US HRTech
- Делаю кастомные ИИ инструменты для неайтишного бизнеса
- Для айтишного – обучаю команды использовать AI в разработке
- Организовываю AI конфы (entropy.talk)
- Пишу в AI и грабли (@ai_grably)

AI и грабли
t.me/ai_grably

План

1. Тренды в индустрии
2. Найм и увольнение
3. Сдвиг в обучении
4. Разбор кейсов внедрения ИИ (и факапов)
5. Todos

Тренды



metr.org/time-horizons/

Переход от копайлота к ИИ-работникам, работающим автономно

Я – менеджер или инструмент?

Проблемы вайбкодинга

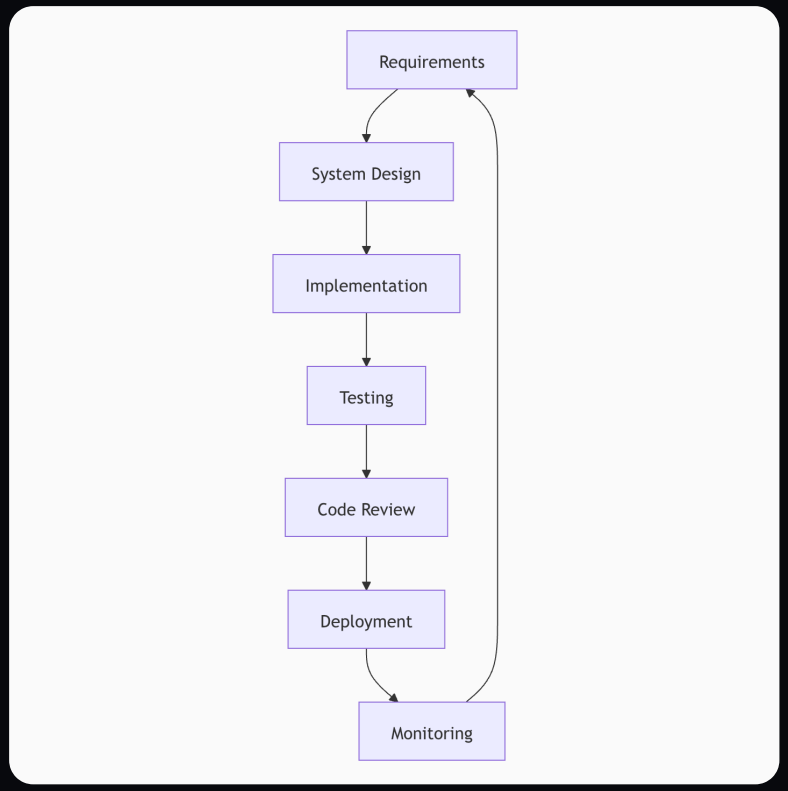
- Новые проекты – быстро упираются в границу сложности
- Старые проекты – много tacit knowledge

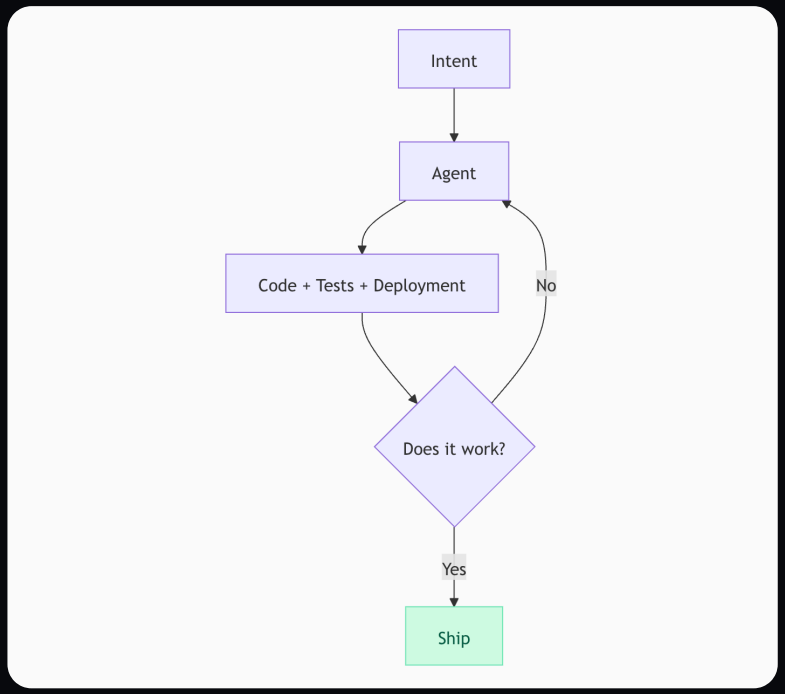
Границы применимости определяются контекстом на входе и контролем на выходе

Распределение времени

ИИ присутствует на каждом из этапов, но больше всего экономит время разработки

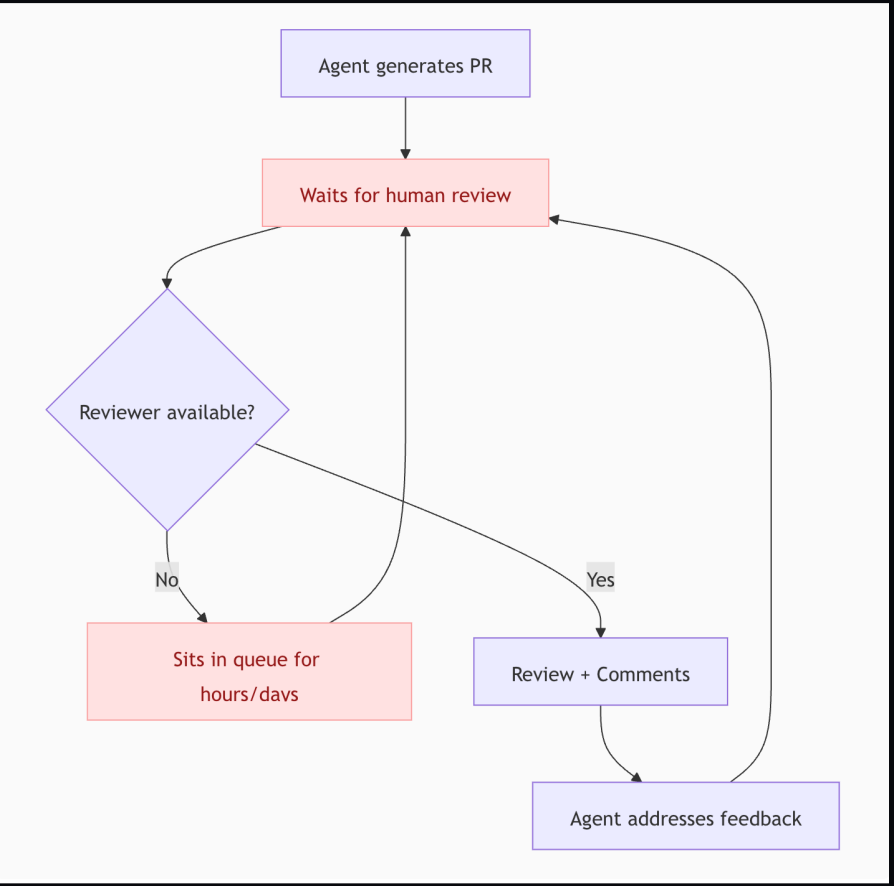
SDLC is dead





Boris Tane: The Software Development Lifecycle is Dead

SDLC is dead



Review Curse

Суммируем

Схлопывание этапов разработки

Developer → Engineering Manager

Команды должны и будут меняться

Что меняется

- Изменение состава (найм и увольнение)
- Обучение
- Другая структура и процессы

Ключевые навыки

- Агентность сотрудника > Агентность агента
- Архитектура, паттерны
- Опыт работы с ИИ

Найм

- Запрещать ИИ
- Требовать ИИ

Проблема – как найти задачи

а) Достаточно сложные, чтобы не решаться в один промпт

б) Достаточно простые, чтобы успеть решить за собеседование

Кейс 1. Команда системных аналитиков

- Света — лид в крупном российском банке
- Уже делают отдельный AI-native этап

Проблема 1

Слишком похоже на обычный этап системного дизайна

Проблема 2

Все, кто хорошо прошел первый этап без ИИ – хорошо проходят и с ИИ

Вопросы к Свете

- А как вообще в компании уже используется ИИ?
- Как ты использовала на последней неделе?
- Где использовала команда?
- Про что говорили как про вау-момент?
- Какие самые геморные задачи перестали быть такими?

Два кластера задач

1. Формирование и исследование архитектурных концепций

   масштабный дипресерч и куча итераций сверху – для собесов не подходит

2. Генерация драфтов API контрактов – то, что нужно

Предложение

Один этап: 40 мин систем дизайна + 20 мин генерация драфта контракта из спроектированной системы

Кандидат уже в контексте — удобно

- Грин-флаги → чеклисты для интервьюера

Докручиваем

- Не давать делать драфт с нуля
- Дать готовый, но плохо спроектированный
- Попросить добавить что-то, что просто так не влезет
- Наблюдаем за подходом к решению

> *"Хороший специалист – знает, как не косячить. Отличный – знает, как исправить, когда кто-то уже накосячил"*

Решение – рефакторинг

- LLMки слишком цепляются за то, что у них уже есть в контексте
- Опытный разраб отличается ~~умом и сообразительностью~~ ленью и скептицизмом

Три типа кандидатов

а) Консерваторы

Не используют ИИ, считают, что сами пишут код лучше, а ИИ работает только для пет-проектов

б) Вайбкодеры

Не напрыгались на граблях, пойдут кодить в режиме ковбоя

в) AI native инженеры

Разберут задачу и архитектуру с агентом, поймут в чем проблема, объяснят ее собеседующему

TL;DR

1. Делаем небольшой проект, но с кривой архитектурой
2. Про рефакторинг ничего не говорим
3. Даем фичу, которую в текущую архитектуру без костылей не встроить
4. Наблюдаем

Мои спекуляции про найм

Снова появятся тестовые задания

Раньше

- Простая задачка не показывает реальные навыки
- Сложная задачка требует слишком много времени и режет конверсию

Сейчас

- Можно давать сложные задачки на один вечер
- Рыночек у работодателя

Кейс 2. Найм AI инженера

US-based startup — Оффлайн агенты

Выполненных тестовых заданий	30
Прошли AI ревью	10
Собеседований	8
Офферов	3

Критерии для увольнения

- Чистый исполнитель (не может ставить задачи и прорабатывать требования)
- Требует постоянного контроля и проверки результатов
- Любит писать код

ВАЖНО: Не увольняем

- Душнил-консерваторов! — они будут балансировать энтузиастов
- Людей с tacit knowledge

Что делать с Джунами???

Хорошего ответа нет

Что делать с Джунами

Плохой ответ: использовать как тренажер менеджерских навыков

Плюсы	Минусы
Больше обратной связи, чем от машины	Обратная связь иногда сильно дольше

Обучение

Тупеем или умнеем?

Примеры вопросов

"Какие есть идиоматичные способы решить эту задачу?"

"Что в твоих инструкциях привело к тому, что ты сделал X?"

"Где в твоем решении могут быть переусложнения, от которых можно избавиться?"

"Если я спрашиваю или предлагаю какой-то бред, так и скажи – я не очень в этом разбираюсь"

Note: /fork

Кейс 3. Внедрение ИИ в команду

- Компания оплачивает ИИ для команды
- Команда пользуется, но кто как умеет, централизированной настройки почти нет
- Перевозят фронт на новый стек, хотят ускорить за счет ИИ, заодно прокачаться

Мой план

Сначала

- Пообщаться с командой
- Процессы разработки
- Какие тулы используют
- Как настраивают

Потом

- Скорректировать поведение людей
- Скорректировать поведение агента (CLAUDE.md, Skills, субагенты)
- Протестировать, сделать еще итерацию, научить делать самим

Первые попытки

- Cursor -> Claude Code
- Опросник для команды (для CLAUDE.md)
- Замыкание Feedback Loop (линтер, typescript, тесты, playwright)
- Участие в синках, корректировка решений команды

Что могло пойти не так?

Что могло пойти не так?

Ошибка 1. Делать самому

Проблема 1: Tacit Knowledge (неявные знания)

Проблема 2: Магический артефакт

Что могло пойти не так?

Ошибка 2. Пытаться работать сразу со всеми

Проблема 1: Разная степень сопротивления

Проблема 2: Разная степень "ИИ-нативности"

Что могло пойти не так?

Ошибка 3. Учить пользоваться ИИ на примере переноса фронта на новый стек

Плюсы

- Можно граундиться об старую реализацию
- Все edge cases уже известны

Минусы

- Тащим старые костыли
- Очень много архитектурной работы

Результат 0

Решение

1. Видео, где я сам реализовываю одну фичу
2. Скрипт выгрузки сессий Claude Code
3. Отсматриваю вместе с codex
4. Пишу фидбэк и планы для команды

Повторяющиеся лайфхаки

- Plan vs Act
- Mermaid диаграммы
- Feedback loop (линтер, typescript, тесты, playwright)
- Повторяющиеся шаги → ~~Скиллы~~ Скрипты
- Ревью вместе с агентом (волшебное слово "инварианты")
- Тесты == спека (всегда sync)

gist.github.com/toolittlecakes/7c15b1304070aa36cd39355db4545c92

Рекомендации для менеджмента

- Прозрачность == Ответственность
- 2-3 основных "чемпиона"
- Отсутствие срочных релизов
- Все изменения делают сотрудники сами
- Начинать с простых задач с низкой ценой ошибки

Итоги

Тренд на автономность → Process engineering

- AI – ваш сотрудник
- Ваши сотрудники – теперь менеджеры
- Но не микро-менеджеры!

Сдвиг в SDLC

- Схлопывание этапов и ролей
- Синхронность → асинхронность
- Написание кода → планирование и ревью

Начинать с низкой цены ошибки

- Рисерч, планирование
- Бизнес логика
- E2E тесты (по спеке!)

Найм меняется

- Тестовые задания снова ок
- Требовать ИИ на собесах
- Менеджерские навыки

Процессы и люди первичны, инструменты вторичны

- Играем в долгосрок == важна производная
- Нельзя оптимизировать то, что не измеряем
- Внедрение без обучения – не работает

План: Найм

- Задачи на 30-40 минут с агентом
- Требует экспертности (рефакторинг!)
- Анализ сессий
- "Какие инструменты использовали и для каких задач за последний месяц?"

План: Разработка / Обучение

- Выделить отдельную команду (2-3 человека)
- Задачи с низкой ценой ошибки, запретить писать код вручную
- Выгрузка сессий и их анализ (опционально: семинары с best practices)
- Итерации (human & agent)

План: NO-DOs

- Архитектурные задачи
- Отдавать ревью агенту
- Делать ревью вручную
- Нанимать "исполнителей"
- Нанимать "у нас в компании ИИ было нельзя"

Почитать

OpenAI: Harness Engineering

The Software Development Lifecycle is Dead

Вопросы к самому себе

1. Что отдавать ИИ, а что делать самому?
2. На какие грин-флаги обращать внимание на собесе?
3. Какие навыки именно мне нужны для роста?
4. А что отличается, если мы сами делаем ИИ продукты?
5. Как дать ИИ-агента не разработчикам?
6. Зачем нам еще быстрее делать фичи?

Все доклады
AI Hard Fork