🤖 Тандем AI-моделей: автономная разработка без предвзятости

🔁 Loop Engineering: Тандем моделей для автономной разработки

Ключевые тезисы:

- Использование двух разных AI-моделей в ролях **тимлида** и **разработчика** создаёт эффективный цикл разработки.
- **Главная проблема** — предвзятость одной модели, которая сама пишет и проверяет код.
- **Решение** — разделение ролей: одна модель планирует и ревьюит, другая — пишет код.
- **Критерии приёмки и тесты** — объективная основа для принятия решения, а не мнение агента.
- Такой подход экономит время, повышает качество и позволяет уверенно двигать проект вперёд.

🎯 Суть подхода: Разделение ролей

Автор переносит принципы командной разработки на работу с AI. Вместо одной модели, которая делает всё, используется **тандем**:

- **Тимлид (Claude Opus):** Отвечает за проектирование, документацию, постановку задач и, самое главное, за **строгое ревью** на основе критериев.
- **Разработчик (GPT-4o / Cursor):** Пишет код по заданию тимлида.

💡 **Ключевая идея:** Разные модели (от Anthropic и OpenAI) обеспечивают **непредвзятый взгляд**. Модель, написавшая код, уже считает его правильным, а

"внешний" ревьювер ищет ошибки.

🔄 Как работает цикл (Loop Engineering)

Нормальный цикл разработки, в отличие от наивного "написано — значит готово", выглядит так:

1. 📋 **Спецификация и критерии приёмки:** Чёткое описание того, **что** нужно сделать и **как** понять, что задача выполнена.
2. ⚙️ **Имплементация:** Модель-разработчик пишет код.
3. 🧪 **Ревью и тестирование:** Модель-тимлид проверяет код, прогоняет тесты, сверяет с критериями.
4. ⚖️ **Верификация:**
   - ✅ **Если всё ОК:** Работа принимается, цикл завершён.
   - 🔄 **Если есть ошибки:** Код отправляется на доработку разработчику с конкретными замечаниями.

⚠️ **Роль человека:** Нести окончательную ответственность, но **не делать ревью вручную**, особенно в сложных системах.

💰 Экономический вопрос: Зачем две подписки?

Использование двух платных моделей (Claude + Cursor/GPT) увеличивает расходы. **Аргументы "за":**

- **Высокая цена ошибки:** Если сроки горят, а результат должен быть качественным.
- **Нехватка компетенций:** Человеку может потребоваться неделя, чтобы разобраться в сложном коде, а модель сделает это за минуты.
- **Объективность:** Разные модели — это разные "взгляды", что снижает риск пропустить баг.

> *Личный выбор автора: Он готов платить за вторую подписку, чтобы получить эффективный тандем и не тратить время на ручное ревью.*

🛠️ Техническая реализация на примере

Задача: Добавить кнопку экспорта лидов в CSV.

Критерии приёмки (для Claude):

1. Есть кнопка.
2. CSV-файл содержит нужные поля.
3. Обрабатывается пустой список.
4. Формат валидный.

Процесс:

1. **Claude (тимлид)** получает промт с задачей и критериями.
2. **Claude вызывает агента-кодера (Cursor)** через команду `go` в терминале, передавая ему детальный план.
3. **Cursor** работает в режиме `go`, пока не выполнит все условия.
4. **Claude** проверяет результат: смотрит diff (разницу в коде), запускает тесты, ищет нарушения.
5. Если тесты не проходят или есть замечания — **Reject** с конкретным списком доработок.
6. Цикл повторяется, пока все критерии не будут **зелёными**.

🧪 Почему тесты — это всё

Тесты (юнит-, end-to-end, smoke-тесты) — это объективные факты, а не мнение.

- Пока функционал не подтверждён тестами, всё остальное — лишь предположение о его работоспособности.
- Тесты дают **измеримое доказательство** выполнения задачи (скриншоты, логи, отчёты о покрытии).

🗂️ Организация проекта

Для работы тандема используется:

1. `claude.md`: Главный файл с описанием проекта, критериями приёмки и **запретом для Claude писать код самостоятельно**.
2. **Skill-файл (например, `cursor_teamlead.md`):** Пошаговое описание всего цикла loop engineering:
    - Как делегировать задачи Cursor.
    - Как настроить **гейты приёмки** (gate) в каждый `go` prompt.
    - Как проводить ревью на предмет "AI slop" (мусора в коде).
    - Шаблоны промтов и вспомогательные bash-скрипты.

🚀 Результаты и выводы

- **На практике:** Автор написал ~100 000 строк кода для большого бизнес-проекта, используя DeepSeek в качестве кодера и Cursor в качестве тимлида, потратив менее $30.
- **Распределение задач:**
    - **Claude (Anthropic):** Для задач, требующих глубокого анализа и ревью, несмотря на более строгие лимиты токенов.
    - **Cursor/OpenAI:** Для ресурсоёмких задач по генерации кода, где важна экономия токенов.
- **Главный итог:** Такой подход позволяет **автономно** получать качественный результат с **доказательной базой** (тесты, артефакты), что ускоряет разработку и повышает уверенность в коде.
