



От идеи к коду: системный подход с AI

Схема разработки с AI

>_ От клиента к коду: как AI помогает превращать идеи в работающий продукт

Клиент → Бриф → ТЗ → План → Задачи → Changelog



по Сергей Стародубцев

2 урок курса

Проблема хаотичной разработки

Современная разработка начинается с системного подхода — от идеи до релиза. AI помогает автоматизировать каждый этап: анализирует требования, генерирует спецификации, подсказывает архитектуру и контролирует прогресс.



Типичные проблемы без системы:

- Постоянные "срочные" изменения
- Превышение бюджета в 2-3 раза
- Сроки срываются регулярно
- Непонятно, что делать дальше
- Выгорание команды

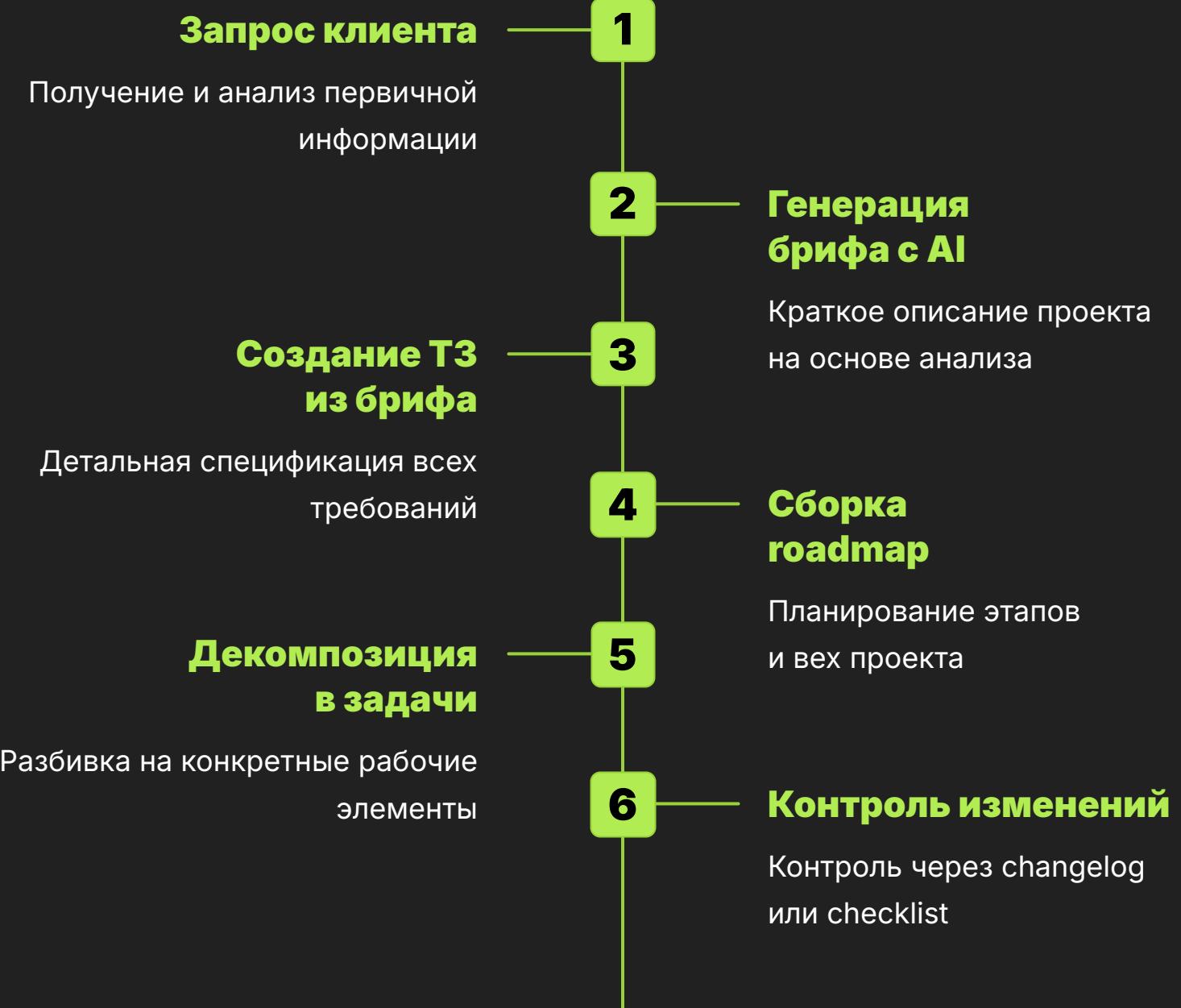


Почему традиционные методы не работают с AI?

AI ускоряет процессы, но без системы хаос тоже ускоряется!



Этапы разработки с AI



AI — не просто помощник по кодингу, а интегратор всех этапов!



Этап 0: Что хочет клиент?

Любой проект начинается с запроса клиента.

Цель:

Понять, что хочет клиент.

Необходимо глубоко разобраться в бизнес-целях, описать результаты простыми словами.

AI может:

- Анализировать сообщения и письма
- Формулировать основные боли и задачи
- Предлагать варианты решений

Промпт-инженер помогает "перевести" язык клиента на язык технологий!

Почему этап 0?

Если клиент не знает, что он хочет или мы его не понимаем, то разработка так и не начнется!



Этап 1: От клиента к брифу

Цель:

Понять реальные потребности и ограничения

Что включает бриф:

- Цели и задачи проекта
- Ключевые требования
- Ограничения и ожидания
- Целевая аудитория
- Бюджет и временные рамки
- Особенности бизнеса

Обычно бриф выглядит как список с вопросами, на которые исполнитель предлагает ответить заказчику перед стартом работ. Он помогает заказчику и исполнителю понять друг друга и прийти к общему видению конечного результата.

Бриф (от англ. *brief* — «инструкция, сводка») — документ, краткая письменная форма согласительного порядка между планирующими сотрудничать сторонами, в которой прописываются основные параметры будущего проекта.

Что должно быть в брифе?



Описание бизнеса

- Сфера деятельности
- Уникальные особенности
- Конкурентные преимущества



Целевая аудитория

- Демография и поведение
- Боли и потребности
- Предпочтения в интерфейсах



Цели проекта

- Бизнес-метрики успеха
- Функциональные требования
- Ожидания от результата

Роль AI на этапе брифа



Генерация вопросов

- AI создает список вопросов для глубокого интервью с клиентом
- AI задает уточняющие вопросы

Анализ конкурентов

Поиск и структурирование информации о конкурентах

Структурирование

Генерирует структурированный бриф из хаотичных пожеланий

Этап 2: Создание технического задания

Цель:

Перевести бизнес-требования в техническую спецификацию

AI помогает:

Структура ТЗ:

- Архитектура системы
- Функциональные требования
- Пользовательские сценарии
- Технические ограничения
- Критерии приемки

Пример промпта:

>_ Сгенерируй ТЗ для интернет-магазина аудиокниг на основе брифа: {текст}.

ТЗ — "руководство" для всей команды разработки: подробно описывает функционал, фиксирует требования к интерфейсу, архитектуре, API

1

Генерировать техническую спецификацию из брифа

Трансформация бизнес-языка в технические требования

2

Предлагать архитектурные решения

Варианты структуры системы на основе требований

3

Создавать пользовательские истории

Разработка сценариев использования продукта

Структура качественного ТЗ

1 Функциональные требования

- Пользователь может добавить товар в корзину
- Система отправляет email после заказа
- Админ может управлять каталогом

2 Нефункциональные требования

- Время загрузки страницы < 2 сек
- Поддержка мобильных устройств
- Шифрование платежных данных

3 Критерии приемки

- Конверсия в покупку > 3%
- Поддержка 1000+ одновременных пользователей
- Uptime > 99.9%

Аптайм (uptime) — это метрика, позволяющая вычислить количество времени, в течение которого сайт, хостинг или сервер может работать без перерывов.



Этап 3: План разработки (Roadmap)

Цель:

Создать последовательность реализации с учетом приоритетов

Принципы планирования:

- MVP сначала, затем расширение
- От простого к сложному
- Итеративная доставка ценности
- Баланс скорости и качества

MVP (minimum viable product — минимально жизнеспособный продукт) – это продукт, который разрабатывается с максимальной экономией денег и ресурсов

AI в планировании:



Анализ зависимостей

Определение связей между задачами



Оценка времени

Прогнозирование сроков разработки



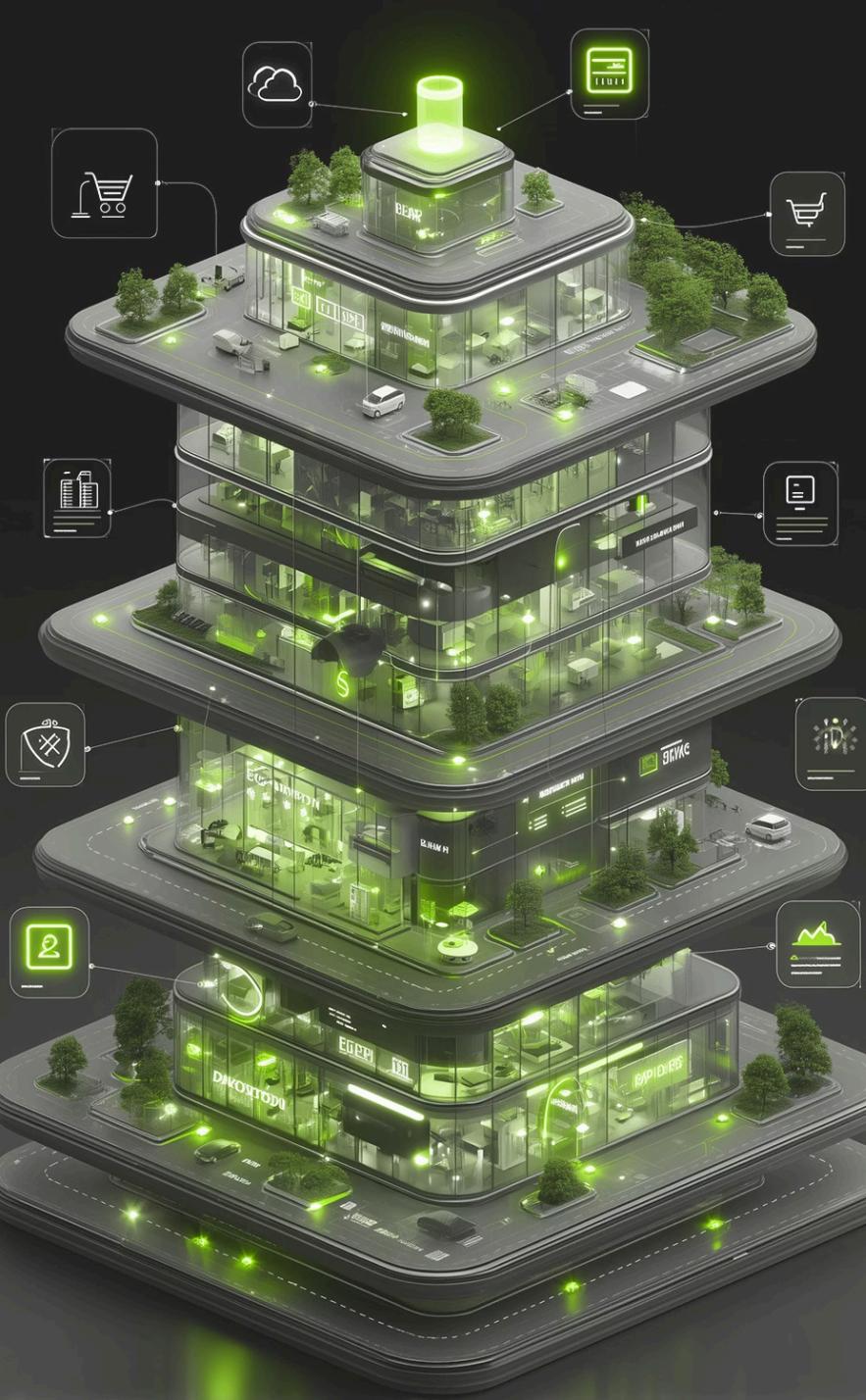
Оптимальная последовательность

Планирование порядка выполнения



Выявление рисков

Обнаружение потенциальных проблем



Пример плана интернет-магазина

1

Фаза 1: MVP (2 недели)

- Статичные страницы
- Простой каталог
- Базовая корзина

2

Фаза 2: Основной функционал (4 недели)

- Регистрация пользователей
- Система оплаты
- Email уведомления

3

Фаза 3: Продвинутые функции (6 недель)

- Рекомендации с AI
- Аналитика
- Расширенная админка

Этап 4: Декомпозиция на задачи

Цель:

Разбить план на конкретные выполнимые задачи

Роль AI:



Дробление целей

AI помогает дробить большие цели на понятные и измеримые подзадачи



Формулировка

Формулирует задачи так, чтобы любой разработчик понял их с первого взгляда



Результат

Готовый список задач для трекера зада

Структура задачи:

Название	Критерии готовности	Зависимости
"Создать страницу каталога"	Что должно работать	"Настроить базу данных"
Описание	Оценка	
Детальные требования	4 часа	

Как AI помогает создавать задачи



Автоматическая декомпозиция

>_ Разбей функцию 'корзина покупок' на задачи по 2-4 часа каждая



Анализ зависимостей

>_ Какие задачи должны быть выполнены перед реализацией системы оплаты?



Оценка времени

>_ Сколько времени займет создание адаптивной верстки каталога товаров?



Генерация критериев готовности

>_ Создай чек-лист готовности для функции регистрации пользователей

>_ - так мы будем помечать примеры промптов



Этап 5: Система контроля через Changelog

Цель:

Прозрачное отслеживание прогресса и изменений

Что фиксируем в Changelog:

- Выполненные задачи
- Изменения в требованиях
- Найденные и исправленные баги
- Метрики производительности
- Принятые технические решения

Когда использовать:

Для анализа прогресса и отчетности по проекту

Формат записи:

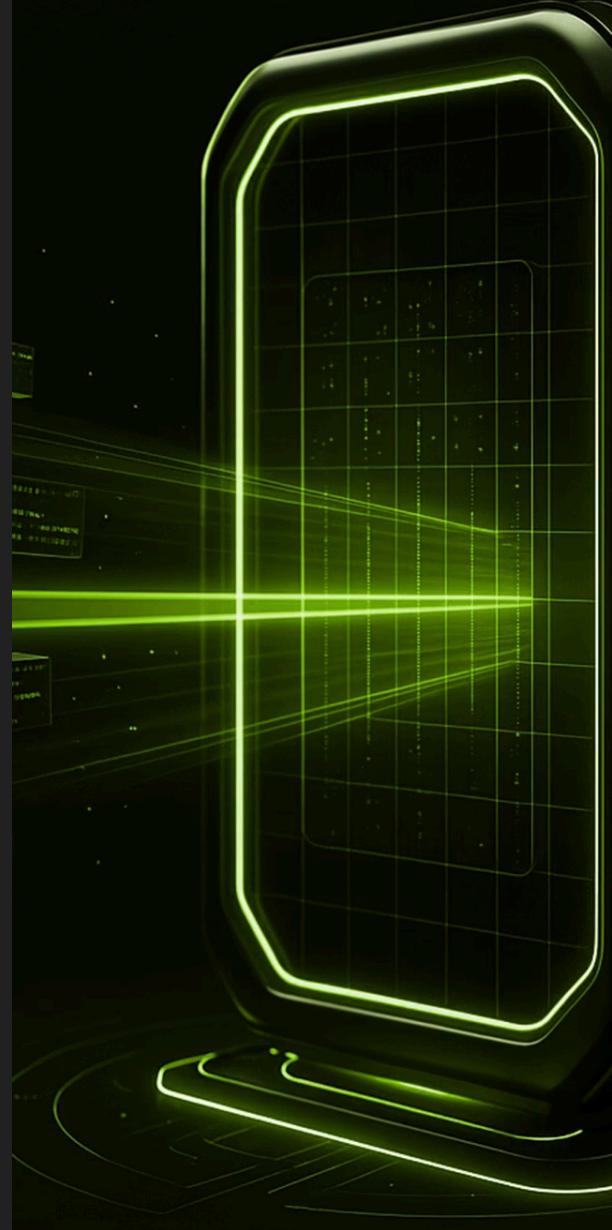
```
## [1.2.0] - 2024-01-15
```

Добавлено

- Фильтрация товаров по категориям
- Сортировка по цене и рейтингу

Исправлено

- Ошибка в расчете скидки
- Проблема с мобильной версией корзины



Этап 5: Система контроля через Checklist

Цель:

Четкое выполнение задач и контроль текущего прогресса

Что фиксируем в Checklist:

- Задачи на текущий этап.
- Критические шаги для завершения этапа.
- Зависимости между задачами.
- Статус выполнения.

Преимущества:

- Фокусировка команды на приоритетных задачах
- Быстрое выявление блокеров и узких мест
- Наглядность прогресса для клиента и команды

Формат записи:

Система пользователей

Backend

Создать модель User

API регистрации/авторизации

Валидация email

Восстановление пароля

Unit тесты

Frontend

Форма регистрации

Форма входа

Личный кабинет

Адаптивная верстка

E2E тесты

Готовность к релизу

Code review завершен

Все тесты проходят

Документация обновлена



Когда использовать:

Для ежедневного контроля задач и удержания фокуса

Changelog vs Checklist: выбираем инструмент контроля

CHANGELOG

- Фиксирует каждое изменение: дата, суть, автор
- Отслеживает историю проекта
- Удобен для анализа прогресса и поиска ошибок
- Формируется автоматически через AI из git-коммитов

Для чего подходит:

- Документирование выполненных изменений
- Отслеживание версий и релизов
- История принятых решений
- Коммуникация с клиентом и командой

CHECKLIST

- Список конкретных задач "что сделать" и "что готово"
- Обеспечивает общую картину хода проекта
- Дает быстрый отклик о статусе задач
- AI автоматизирует отметки о завершении задач

Для чего подходит:

- Контроль выполнения текущих задач
- Быстрая проверка готовности
- Визуальный прогресс работы
- Планирование ближайших шагов

КОМБИНИРОВАННЫЙ ПОДХОД

Рекомендация: Интегрируйте оба инструмента в единую систему контроля с помощью AI-ассистентов для максимальной эффективности

- Checklist для планирования и контроля спринта
- Changelog для документирования результатов
- Реализуйте задачи согласно чек-листу (Checklist), отмечая прогресс (Changelog)

Роли AI-команды на каждом этапе



Промпт-инженер:

- Составляет запросы к AI для каждого шага
- Проверяет корректность ответов AI
- Настраивает автоматизацию

Вайб-кодер:

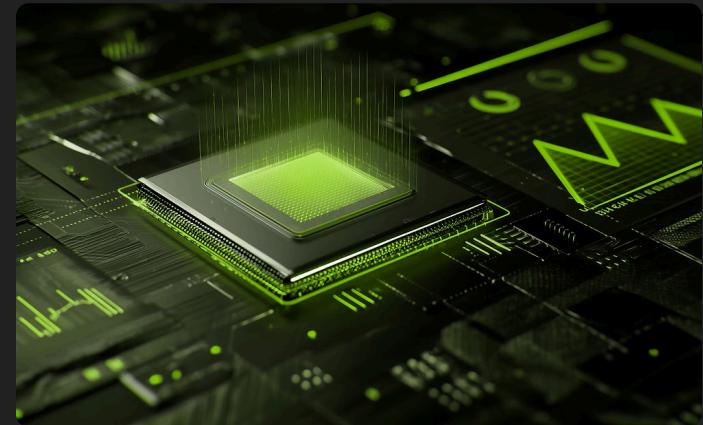
- Участвует в мозговых штурмах
- Тестирует новые идеи и сценарии
- Помогает сделать процесс "живым" и творческим



AI-архитектор:

- Проектирует структуру системы
- Определяет интеграцию AI в ключевые процессы
- Обеспечивает согласованность между всеми этапами и ролями
- Балансирует скорость и надежность решений

Преимущества системного подхода



Для команды

- Понятные цели и приоритеты
- Прозрачный прогресс
- Повышенная мотивация
- Быстрая адаптация

к изменениям

Для клиента

- Контроль бюджета
- Предсказуемые сроки
- Видимость процесса
- Гарантия результата

Для AI

- Структурированные промпты
- Четкий контекст задач
- Последовательная работа
- Улучшение качества ответов

Частые ошибки и как их избежать

Типичные ошибки:



Пропуск этапов

"Давайте сразу к коду!" → Хаос в середине проекта



Размытые формулировки

"Сделать красиво" → Бесконечные доработки



Игнорирование ограничений

"Добавим еще функций" → Превышение бюджета

Как избегать:

- Никогда не пропускайте бриф
- Все требования в письменном виде
- Фиксированные временные рамки этапов
- Используйте AI для проверки полноты



Главное: зачем нужна схема AI-разработки?



Системность

Систематизирует работу — меньше хаоса, больше предсказуемости



Эффективность

Экономит время — AI "делает тяжёлое"



Прозрачность

Все шаги прозрачны для команды и заказчика



Ясность

Маршрут понятен: от идеи до готового продукта

Ваша задача — научиться запускать свои проекты именно по такой схеме, используя силу современных AI-инструментов!

