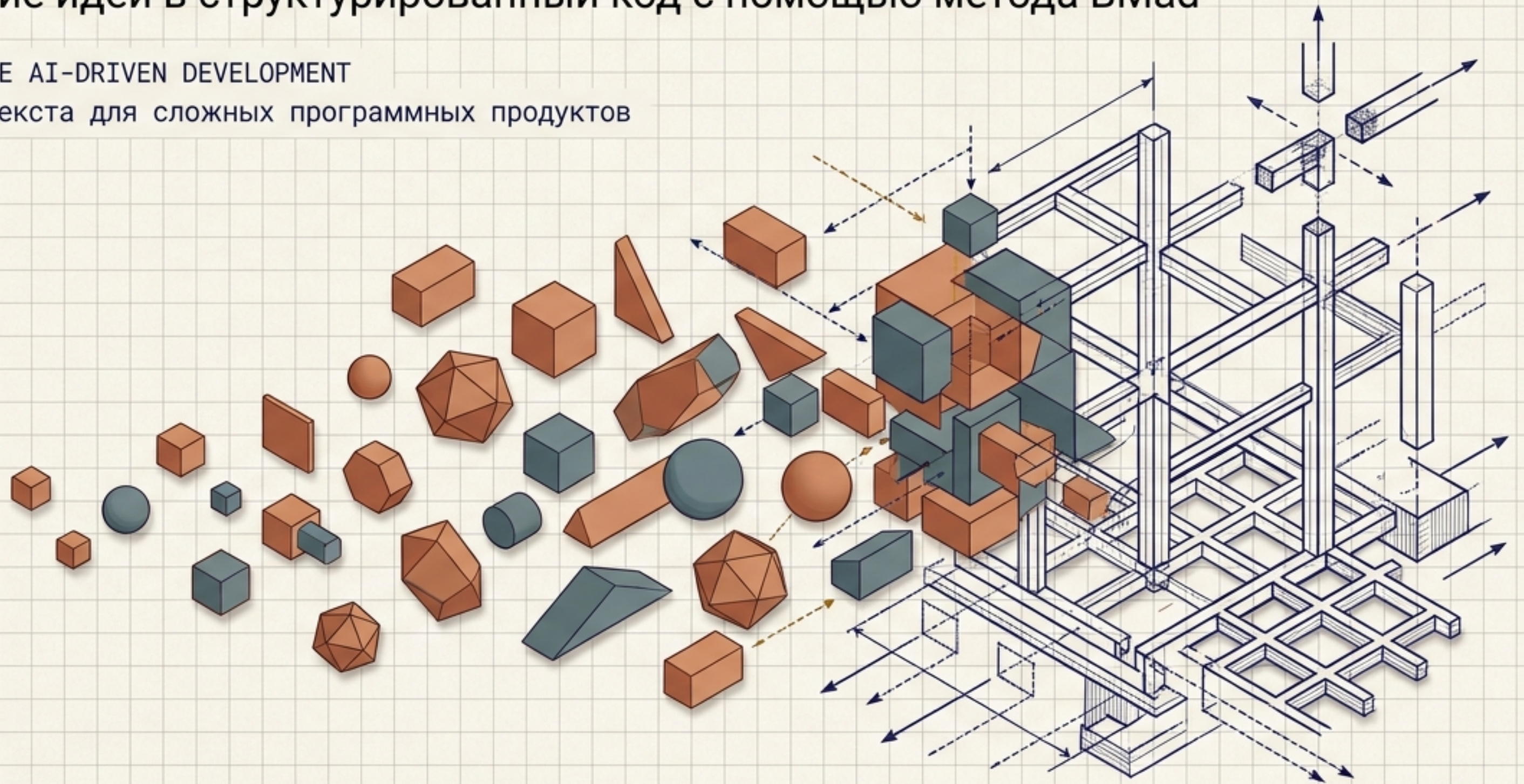


Модуль 1. Аналитика и работа с требованиями

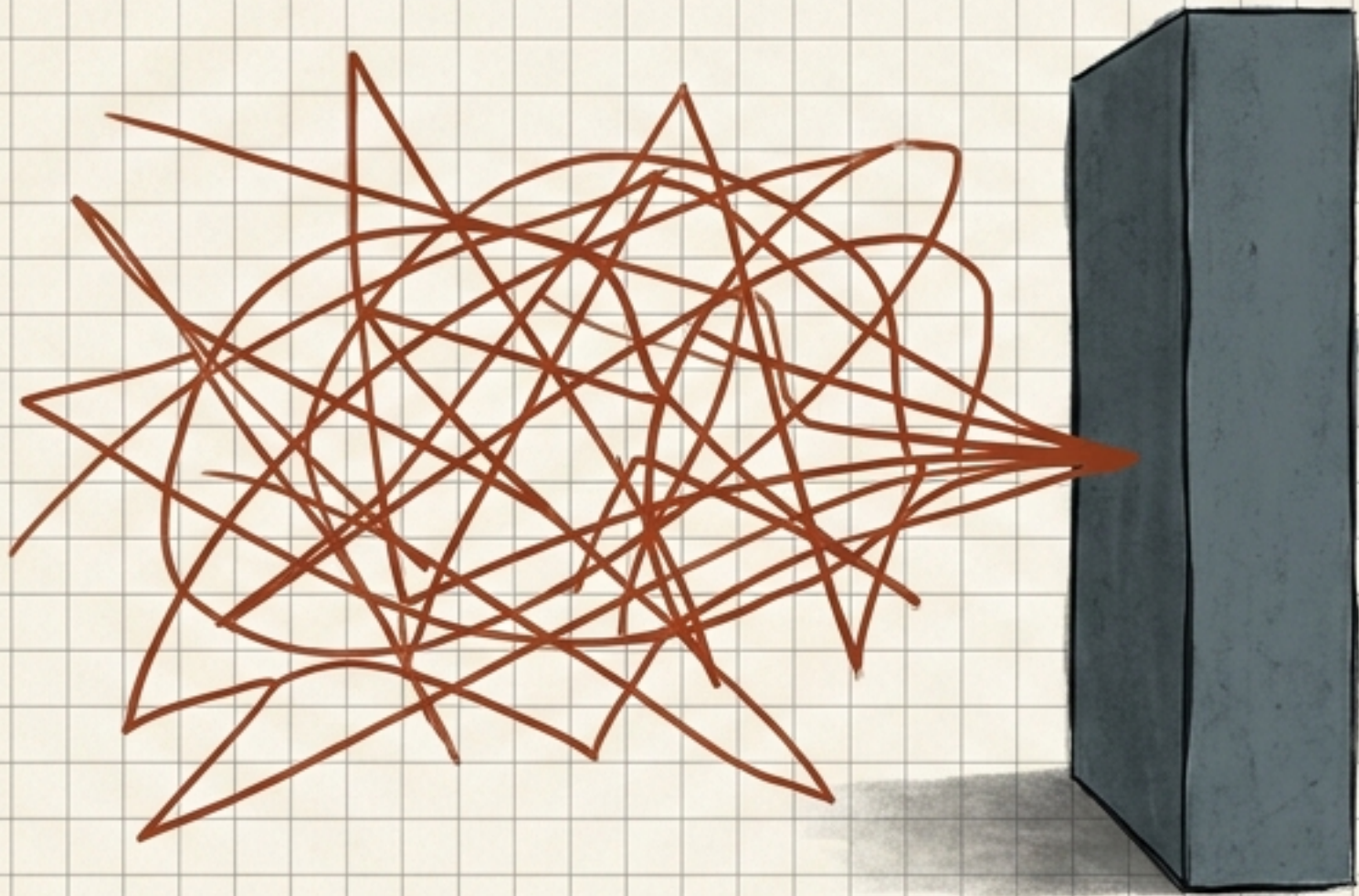
Превращение идей в структурированный код с помощью метода BMad

v6.12.0 // AGILE AI-DRIVEN DEVELOPMENT

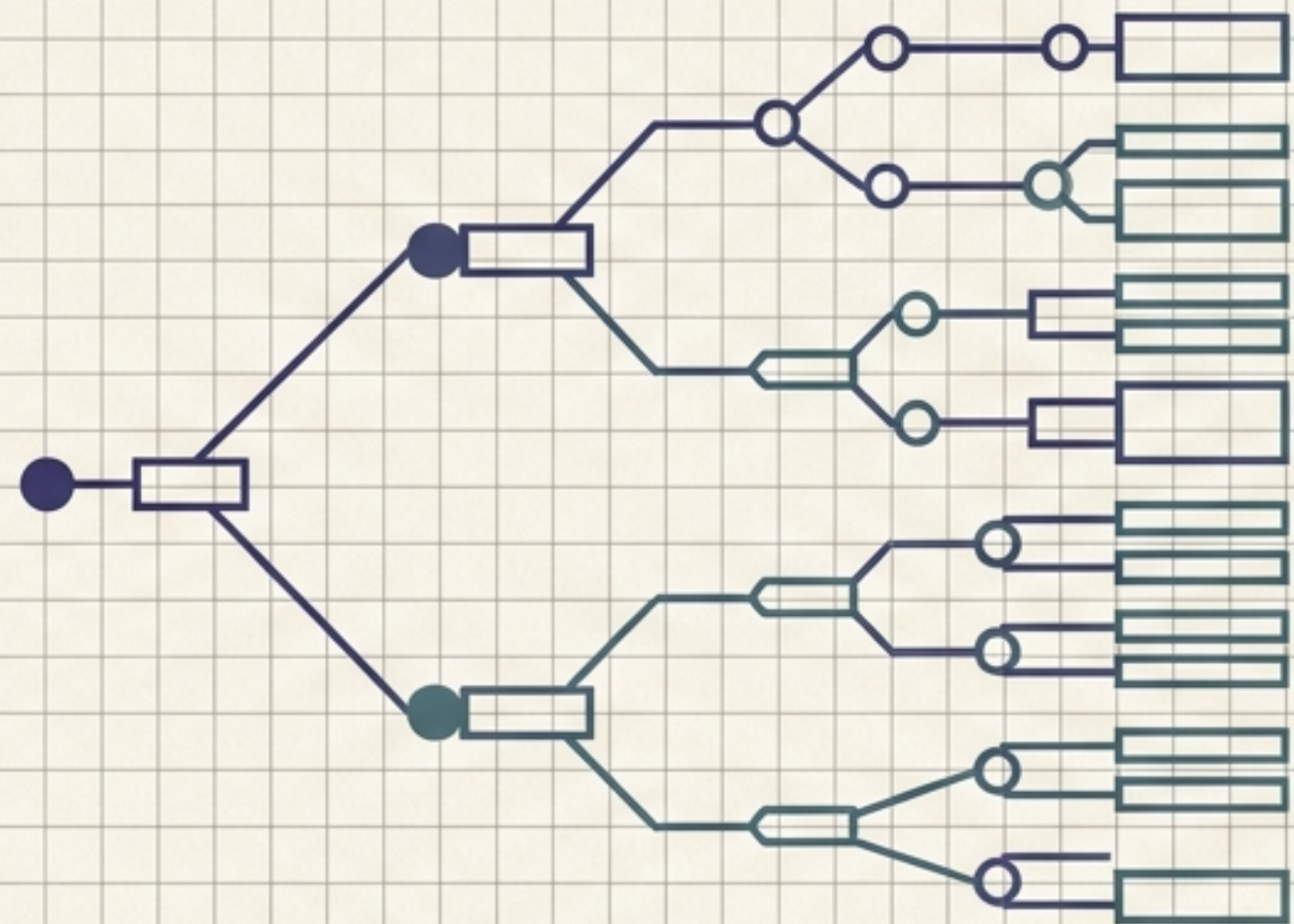
Инженерия контекста для сложных программных продуктов



«Почему мы начинаем с аналитики?»



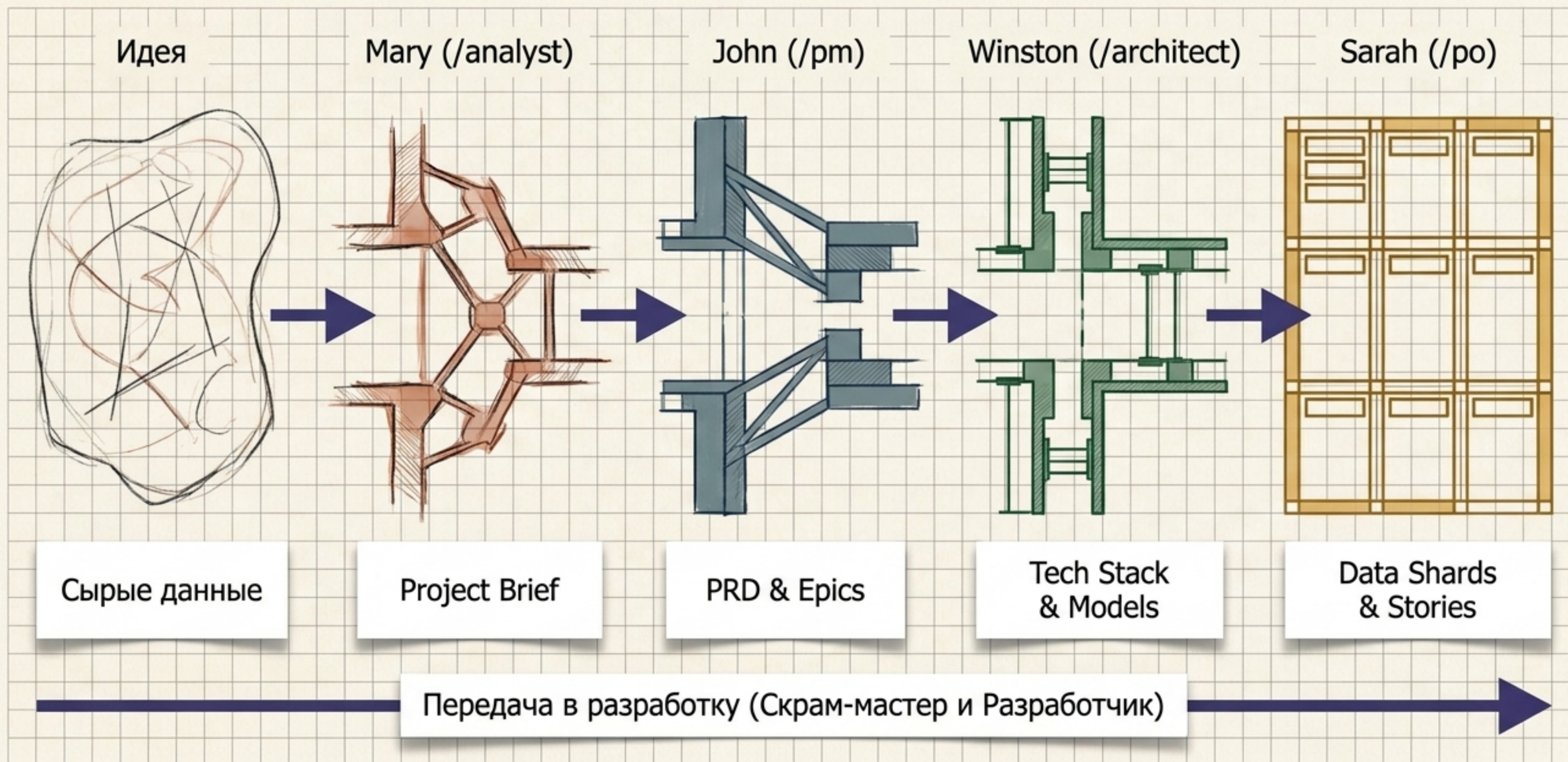
Vibe coding: общение с ИИ наугад.
Работает для прототипов, ломается на масштабе.
В итоге вы упираетесь в стену.



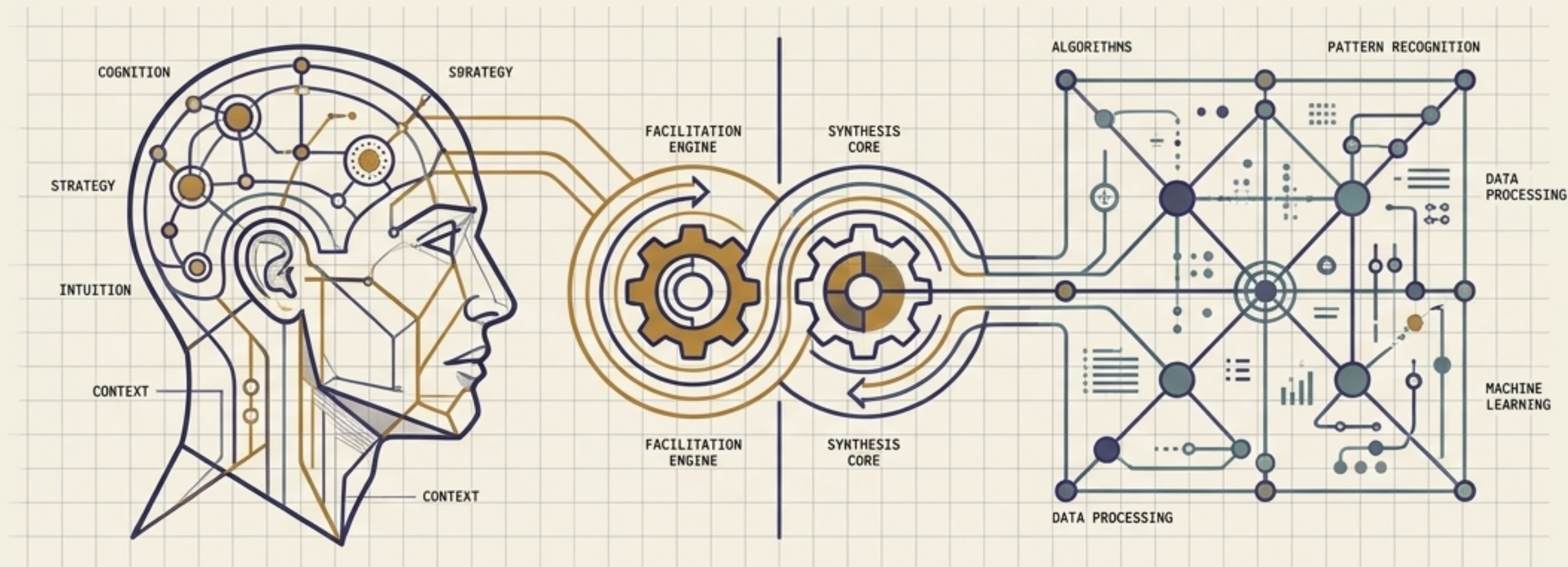
Метод VMad: Проектирование контекста.
ИИ понимает, ЧТО строить, до того, как напишет
первую строчку кода. Структура бьет хаос.

ГЛАВНОЕ ПРАВИЛО: Если вы потратите время на фазу планирования, разработка пройдет без сучка и задоринки.

Воронка планирования VMad



«ИИ – это фасилитатор, а не Google»



Не оставляйте свой мозг за дверью

ИИ не делает работу за вас, он делает её вместе с вами. Вы управляете процессом.

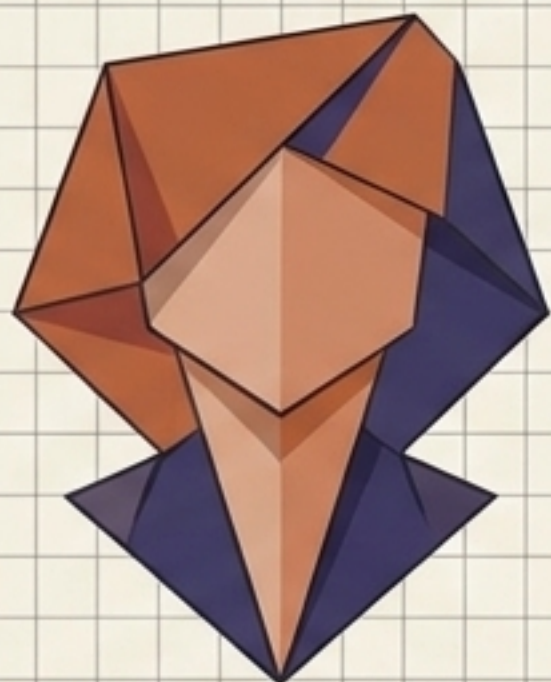
Симбиоз, а не автопилот

Метод бросает вам вызов, заставляя вытаскивать на поверхность лучшие идеи.

Отказ от усредненности

Стандартный промпт дает средний результат. Фасилитация поднимает вас на уровень, которого вы бы не достигли в одиночку.

Шаг 1: Формирование видения



Mary

> /analyst

ИМЯ: Mary

РОЛЬ: Бизнес-аналитик

ЦЕЛЬ: Распаковка сырых идей, мозговой штурм и создание Project Brief.

ИДЕАЛЬНО ДЛЯ: Размытых концепций, поиска слепых зон и конкурентного анализа.

Mary помогает понять, ЗАЧЕМ мы это строим и КТО наша целевая аудитория, прежде чем мы перейдем к функциям.

«Арсенал креативного хаоса»

5 «Почему»

МЕХАНИКА: Глубокий анализ первопричин через цепочку вопросов.

РЕЗУЛЬТАТ: Докапывается до истинной ценности фичи.

Шесть шляп

МЕХАНИКА: Оценка идеи с разных точек зрения (факты, эмоции, риски).

РЕЗУЛЬТАТ: Выявляет скрытые риски и слепые зоны проекта.

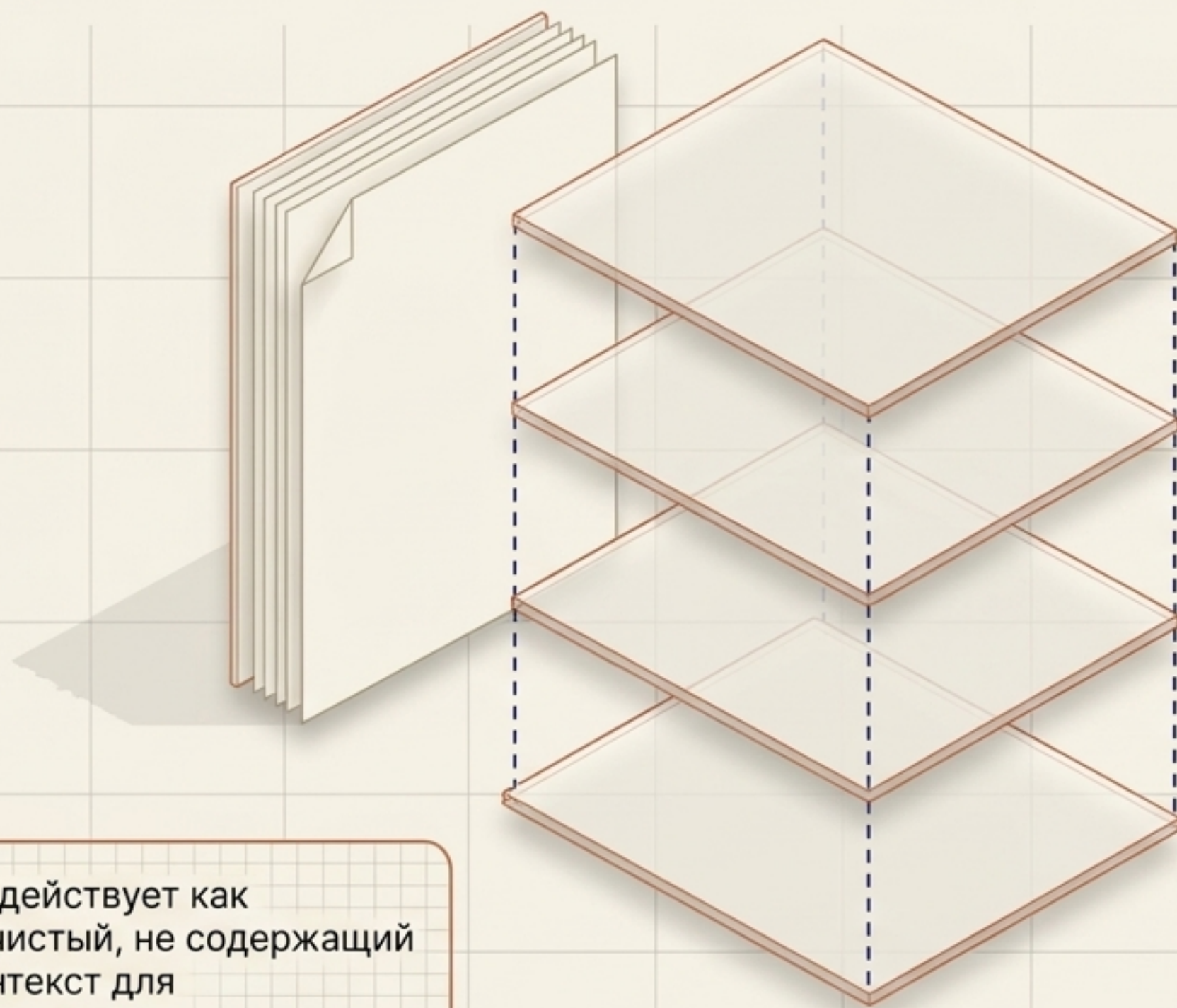
Ролевые игры

МЕХАНИКА: Тестирование идеи на конкретных персонах (напр., «Перегруженный фрилансер»).

РЕЗУЛЬТАТ: Фокус на UX и решении реальных болей пользователей.

ИНФОРМАЦИЯ: Доступно более 20 встроенных техник фасилитации.

Выходной артефакт: Project Brief



Этот бриф действует как идеально чистый, не содержащий мусора контекст для следующего агента.

1. Executive Summary

Суть проекта и главная бизнес-ценность.

2. Problem Statement

Проблемы и боли, которые решает продукт.

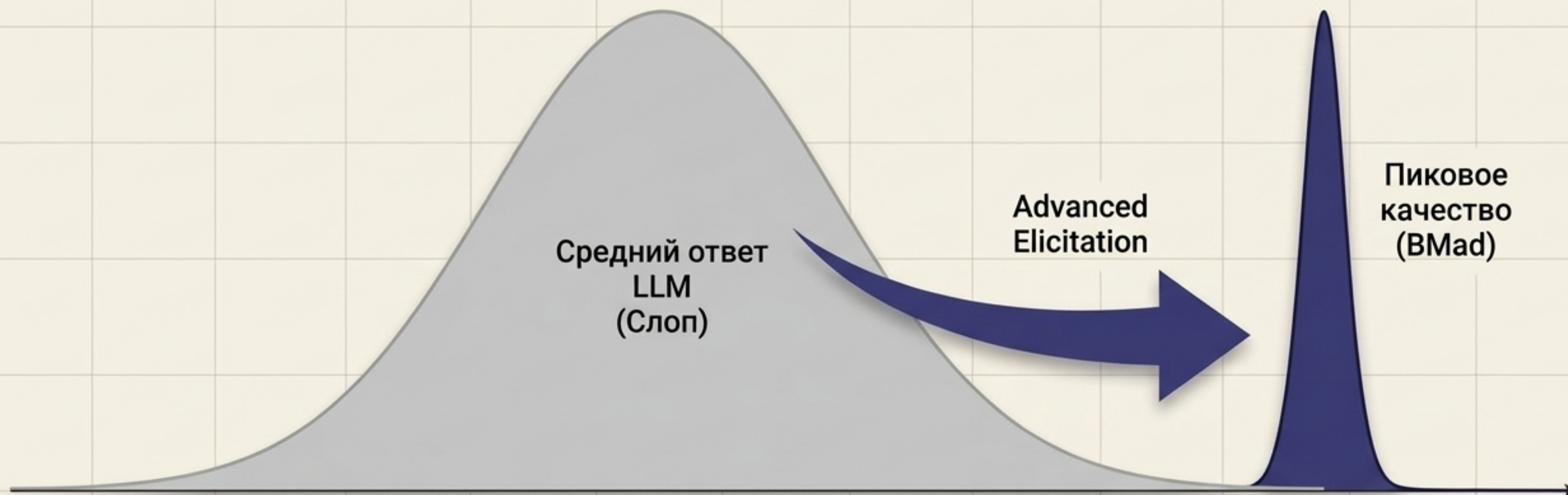
3. Target Users

Целевая аудитория: демография и паттерны использования.

4. Proposed Solution

Конкретное предлагаемое решение.

«Как заставить LLM работать на пределе»



Не принимайте первый ответ агента.



Используйте встроенные техники (Critical Assumption Testing) в конце каждого раздела.



Бросьте вызов ИИ, чтобы он переосмыслил свои же предложения и нашел уязвимости.

«Шаг 2: От идеи к требованиям»



> /pm

ИМЯ:

John

РОЛЬ:

Продакт-менеджер

ВХОД:

Project Brief (от Mary)

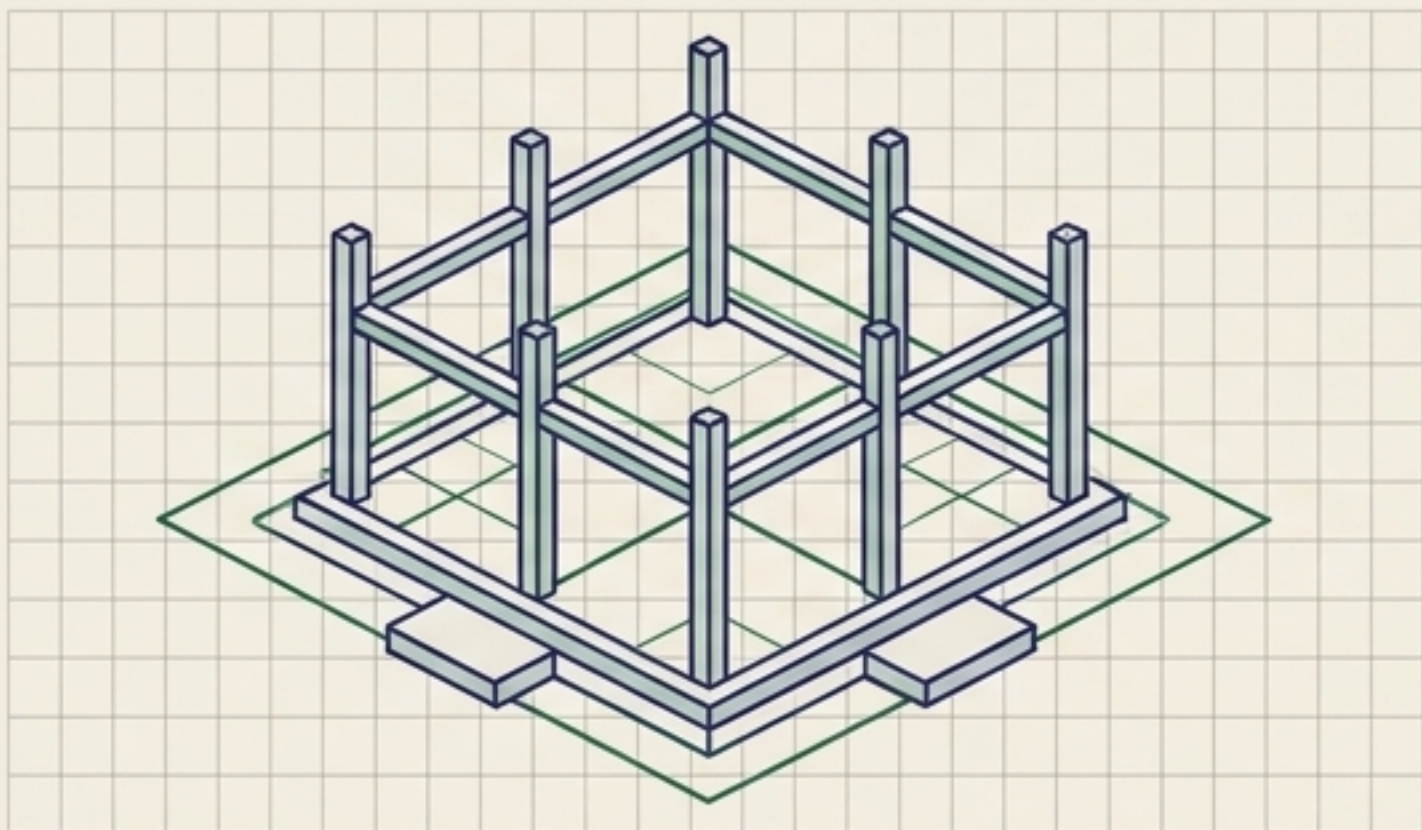
ЦЕЛЬ:

Создание PRD, выделение MVP,
нарезка Эпиков.

John превращает человеческие желания в жесткие функциональные требования, готовые для разработки.

Адаптация под масштаб проекта

«Greenfield (Чистый лист)»



Создание архитектуры с нуля. Фокус на новых рыночных требованиях, свободный выбор идеального технологического стека.

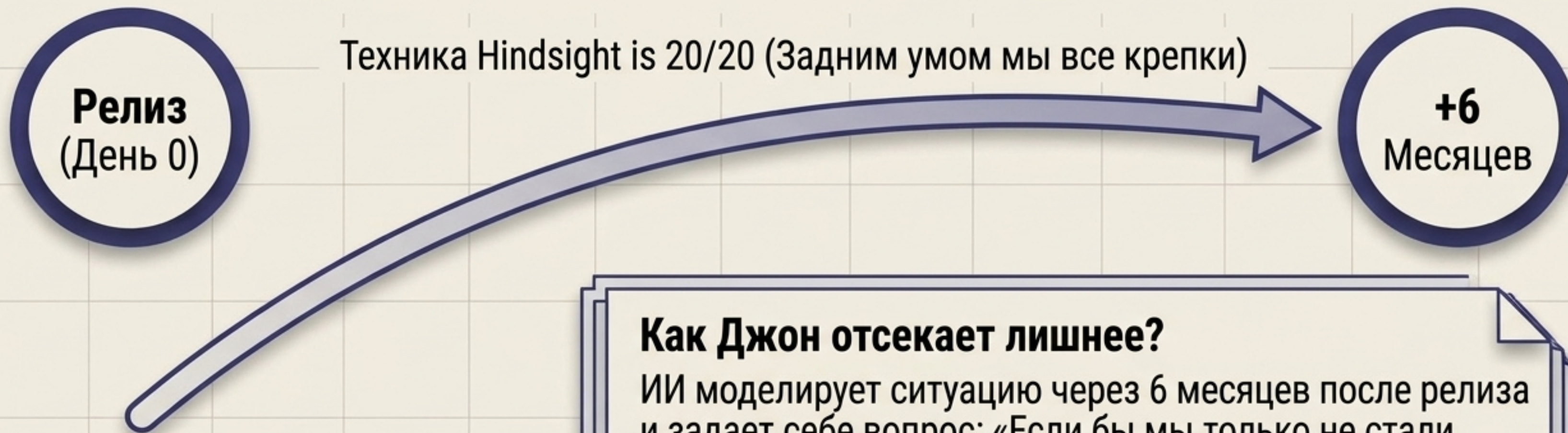
«Brownfield (Существующий код)»



Внедрение в существующую кодовую базу. Учитывает legacy-ограничения, зависимости, риски интеграции. Начинается с анализа текущего контекста.

ВMad автоматически адаптирует размер и сложность PRD под ваш сценарий.

Защита MVP от разрастания



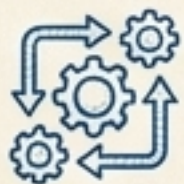
Как Джон отсекает лишнее?

ИИ моделирует ситуацию через 6 месяцев после релиза и задает себе вопрос: «Если бы мы только не стали внедрять эту сложную фичу в первой версии...»

РЕЗУЛЬТАТ: Выявляет тяжелые нефункциональные требования, которые убьют бюджет, и безжалостно переносит их в Post-MVP. Защищает бережливую разработку.

Анатомия PRD (Product Requirements Document)

Functional Requirements



Что система должна делать
(Действия пользователя).

Non-Functional Requirements



Как система должна работать
(Производительность, безопасность).

Scope Management



Четкое разделение на MVP
и Post-MVP.

CORE OUTPUT: Epics (Эпики)



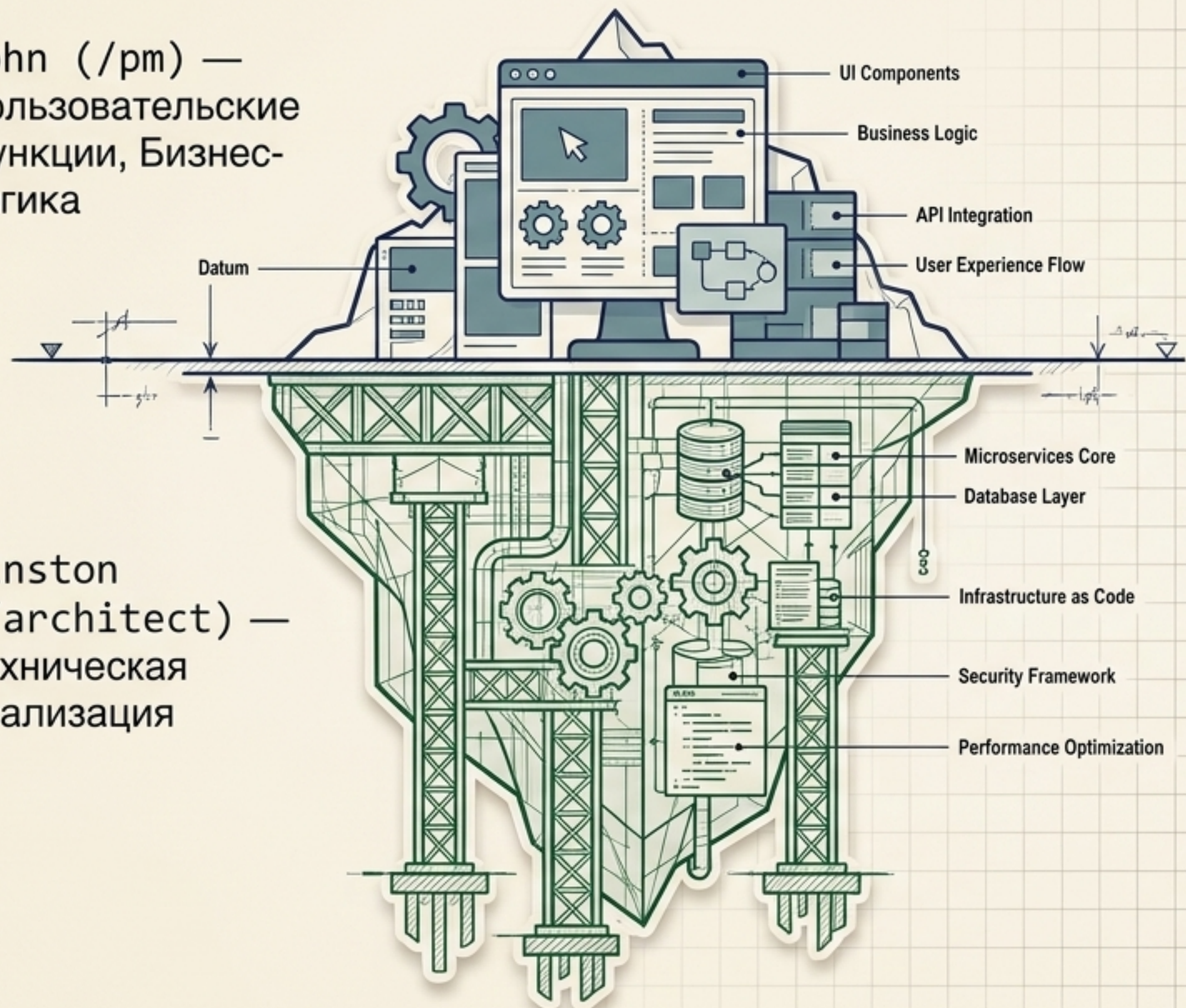
Крупные функциональные блоки
(Например: Epic 1 - Core CRUD Operations).

ОШИБКА В PRD СТОИТ В 10 РАЗ ДОРОЖЕ, ЧЕМ ОШИБКА В КОДЕ.

«Шаг 3: Технический фундамент»

John (/pm) —
Пользовательские
функции, Бизнес-
логика

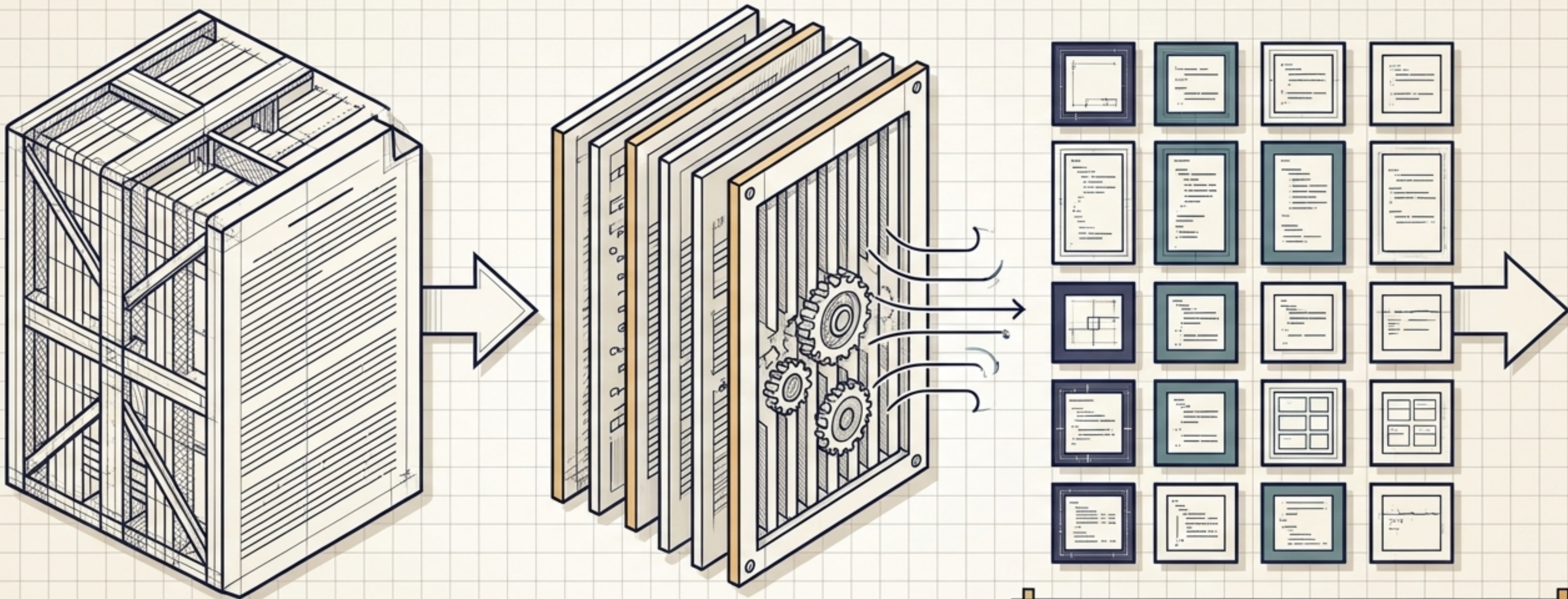
Winston
(/architect) —
Техническая
реализация



ВЫХОДНЫЕ ДАННЫЕ АРХИТЕКТОРА:

- **Tech Stack:** Строгая фиксация версий библиотек.
- **Data Models:** Type-safe интерфейсы и схемы БД.
- **Coding Standards:** Правила документации и структуры (Source Tree).

«Контекстная инженерия: Дробление (Sharding)»



Кодящие агенты (Claude Code, Cursor) ненавидят огромные документы. Избыток текста ведет к галлюцинациям и потере фокуса.

Sarah (Product Owner) автоматически дробит большие документы на крошечные, сфокусированные файлы.

РЕЗУЛЬТАТ: Максимальный контекст для текущей задачи, нулевой информационный шум.

«Идеальная User Story & Итоги Модуля 1»

Формат:

Как [роль], я хочу [действие],
чтобы [причина].

Acceptance Criteria:

☐ Что значит «сделано»
(чек-лист для QA).

Context:



Прикрепленные шарды
(Tech Stack, Data Models).

Итоги Модуля 1



1. Идеи отфильтрованы (Mary).



2. Требования и MVP
зафиксированы (John).



3. Архитектура скрыта под
капотом (Winston).



4. Документы раздроблены
для ИИ-разработчика (Sarah).

Ваш проект готов к передаче Бобу (Скрам-мастеру) и Джеймсу (Разработчику). Переход к Модулю 2.