



Как организовать гибридную команду людей и ИИ-агентов



Максим Цепков

IT-архитектор и бизнес-аналитик

Навигатор и эксперт по миру Agile,
методов самоуправления и моделей soft skill

 [mtsepkov](https://t.me/mtsepkov)  mtsepkov.org

Joker | Санкт-Петербург
14-15 октября 2026

О чем будет рассказ?



AI? Not! Super Intellect!

- Концептуальный обзор про agentic engineering: практика против теории
- Не будет конкретики про модели: **модели – ничто, harness – все!**
- Разработка: ИИ-помощники дают от 30% до 10-15 раз производительности
 - Теория (SDD, Disrupt PDCL) не работает или работает ограниченно – почему?
 - Есть альтернативная успешная практика – какая?
 - Как встраивать ИИ-агентов для конкретных операций (review, типовые задачи, доки)
- Изменяется не только pipeline разработки, но и место ИТ – **что именно?**
 - Бизнес вайбкодит прототипы и решения, приносит их вместо идей
 - Если гипотеза проверяется за 3-5 дней вместо 2-3 месяцев – процесс надо менять
 - Взаимодействие с ИИ дает полный лог работы сотрудника – это можно использовать!
- ИИ-агенты будут работать не только в ИТ, но и в бизнесе: встройка и риски

Итоги: от вайбкодинга – к agentic engineering, от контекста – к harness

Немного обо мне

Создание и внедрение больших корпоративных систем (30+ лет)

- Знание практик операционного управления и ведения проектов в коммерческих и государственных организациях и банках
- Опыт управления проектами в IT: от инженерного подхода и PMBOK – к современным Agile-методам (с 2007 года)
- Опыт перестройки организаций при внедрении систем

Навигация в менеджменте цифрового мира

- Agile и самоуправление: бирюзовые организации, холакратия и социократия ([книга, статьи и выступления](#))
- Модель [спиральной динамики](#) (с 2013) и другие [модели soft skills](#), [модели личности](#) ([книга](#)) и [самоопределения](#) ([книга](#))

Корпоративная культура Китая – лидера новой волны технологических революций ([книга](#))

На моем сайте mtsepkov.org – выступления, блог и много другого



SDD не работает...

Проблема внедрения SDD – проверка спецификации

Для получения хорошего результата спецификация должна быть подробной и технической – и **она становится плохо проверяемой**

- Ее точно не может проверить бизнес и она ему не нужна
- Ее тяжело проверяют сами разработчики
- Проверка не гарантирует, что будет сделано то, что нужно бизнесу
- Проверка не гарантирует, что в коде будет так, как требует спецификация
- У разработчиков теряется мотивация

Почему это так и что делать?

Проверка спецификации и создание кода – **разное**

- Разработчики прокачивали умение эффективно создавать код – переводить задачу на язык программирования, используя шаблоны
- Проверка спецификации – принципиально иное умение: хороший писатель редко бывает качественным редактором и наоборот
- Смена обязанностей меняет ориентиры в профессии и демотивирует

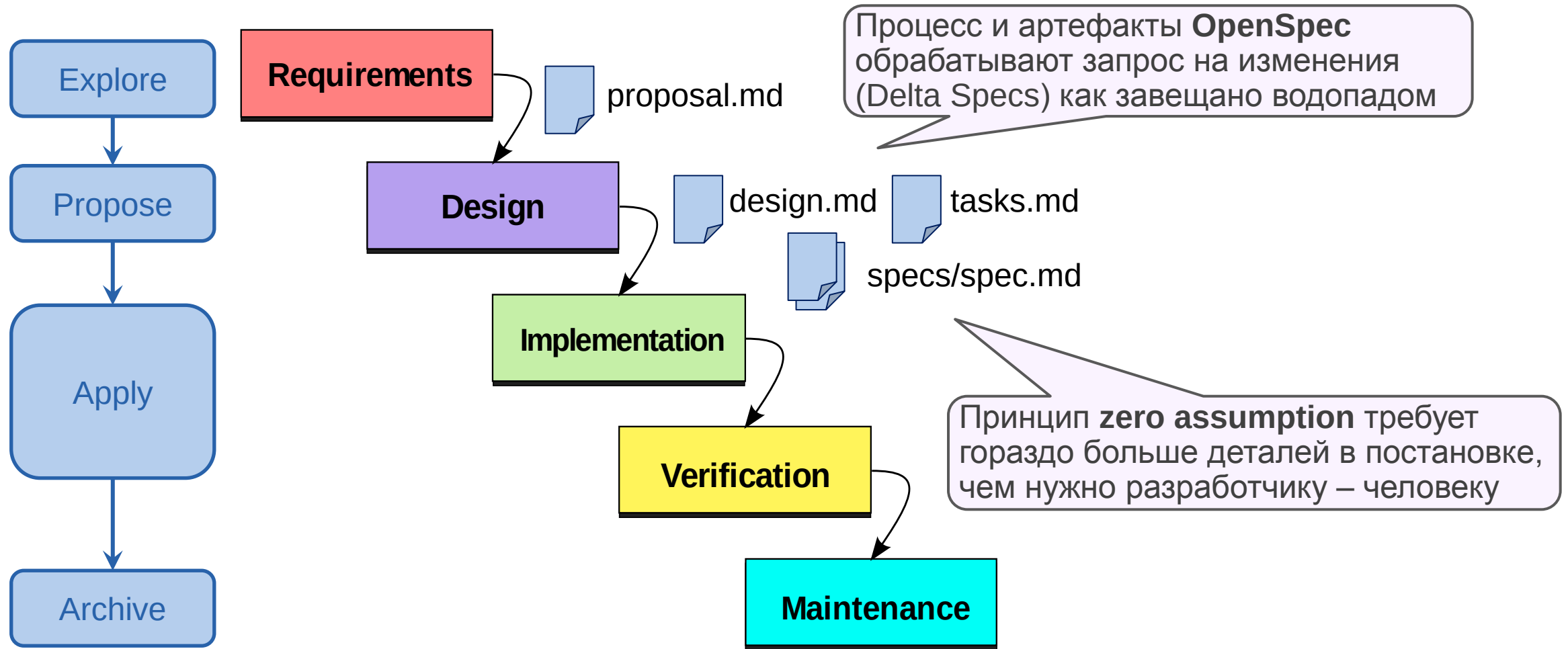
Как облегчают проблему проверки?

- Вместо проверки можно писать чек-лист правил хорошей спецификации, и пусть ИИ проверяет – но это тоже другой характер работы
- Еще hint: **проверяем не спецификацию, а точки принятия решений**, в которых ИИ делал выбор из альтернатив – это ближе и проще

SDD работает на потоке типовых задач – рельсы

- Таких задач много: сайт-визитка, типовой интернет-магазин, типовой документооборот, типовая CRM небольшого бизнеса
- Раньше делали базовое решение с точками кастомизации, ее делала пара аналитик и разработчик, иногда докручивая базу
- Теперь можно иначе, с помощью ИИ
 - Создаем harness для скелета решения
 - Аналитик может вайбкодить специализированное решение
 - А разработчик смотрит, где в решениях возникают проблемы и докручивает harness
- Есть ряд кейсов, например, на Merge-2026 рассказывали: в компании enterprise разработки из 10 проектов разработчики остались на двух, остальные вайбкодят аналитики с прежним качеством и темпами, руководитель держит руку на пульсе и докручивает harness
- **Сложность таких решений ограничена, но гибкость – увеличилась**

SDD – очередная инкарнация водопада/RUP



Модель водопада – http://en.wikipedia.org/wiki/Waterfall_model

Процессы в ИТ не обеспечивают результата!



Идея RUP – гарантировать успех проекта, убрав неопределенность на этап проектирования. Она **провалилась**. Подробности – в моем выступлении [«Почему проектный подход не работает в ИТ» \(Teamlead-2022\)](#)

Проектирование – НИОКР, результат не гарантирован

Исполнение – производство, результат обеспечивают регламенты, включая контроль рисков

Concept

Requirements and Architecture

Проект

Detailed Design

Спецификация проекта устраняет неопределенность

Maintenance

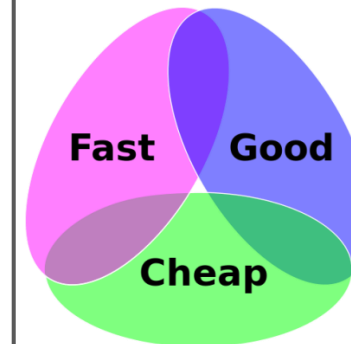
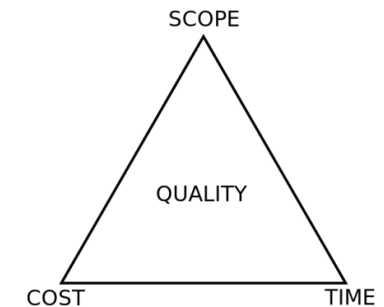
System Verification

Integration and Test

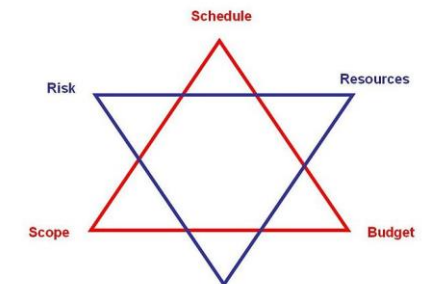
Implementation

РМ контролирует результат проектирования, а затем управляет исполнением в проектном треугольнике

Проектный треугольник



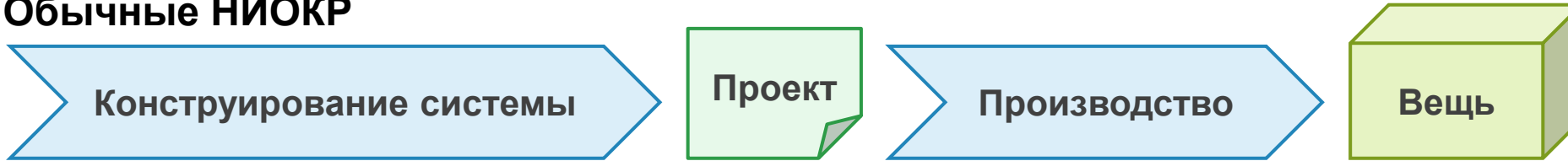
"Triple Constraint" in Project Management



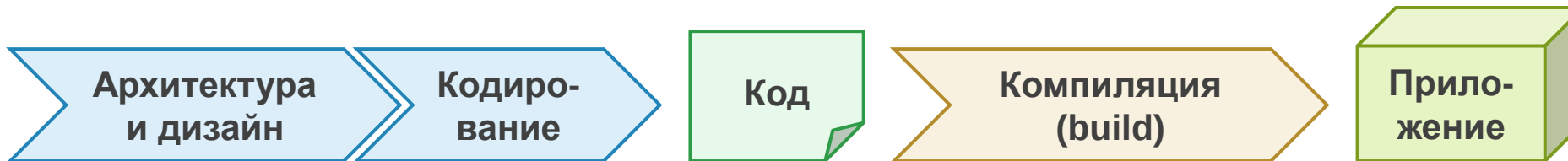
Спецификация не убирает неопределенность

[Jack W. Reeves. What is software design](#), 1992 ([перевод](#))

Обычные НИОКР



ИТ-разработка



А еще для уверенного успеха НИОКР в проектировании нельзя использовать технологии с уровнем зрелости ([TRL](#)) ниже 8 (опыт Боинг). Если бы так действовали в IT, смартфоны выходили бы без софта 😊 По факту используют технологии уровня 4–5, доводя их внутри проекта, при этом сроки и трудозатраты уверенно предсказать нельзя

Два разделения труда: division и sharing

- Мы можем делить труд на операции, а можем объединяться для их совместного выполнения
- Объединение важно, если есть большая неопределенность в решении, тогда бизнес (аналитик) и разработчик совместно ищут решение
 - Вместо сложной системы шаблонов документов – каталог образцов для копирования
 - Вместо конструктора отчетов – средство легко вывести SQL-запрос на интерфейс
- Кроссфункциональные команды и T-shape – про sharing вместо division
- Универсализм ИИ-агентов позволяет делать **tiny-team**: бизнес + технарь, но для них надо обеспечить bus factor больше 1
- Набор компетенций сильно расширяется
- Одна из новых компетенций – быстрая смена фокуса на задачах

Альтернативы SDD

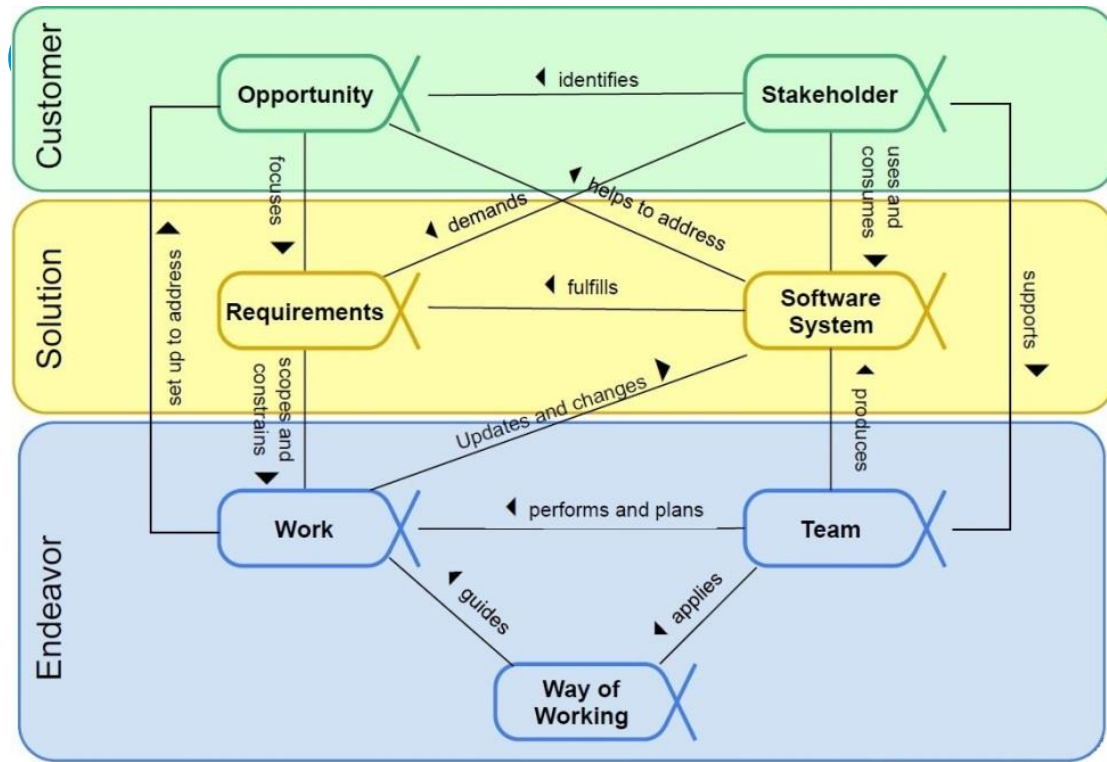
Рабочие альтернативы SDD

- Развитие TDD/BDD: acceptance criteria и тест-кейсы проверять легче, бизнес их понимает – задаем их и архитектуру, дальше – агенты
 - **Виталий Левченко** «[SDD, или Как кратко ускорить разработку с LLM](#)» (Codefest-2026): в названии SDD, но по сути в описании – архитектурные принципы и тесты, по ним спецификацию и код итеративно доводят агенты
- Если бизнес не готов формулировать, что нужно – надо делать прототипы
 - Прототипы показываем и доводим, разработка прототипов ускоряетсякратно
 - По мере стабилизации – формируем архитектурные описания и тест-кейсы
- Кейсы:
 - Андрей Сотник и Родион Лунев «[AI-помощник. Как эксперимент вышел из-под контроля](#)» (Saint Teamlead-2026) – серийное изготовление прототипов в X5-tech
 - Вадим Герасимов «[От хаоса к системе: Почему LLM не отменяет анализ, а требует его эволюции](#)» (AnalystDays-2026) – создание и развитие небольшой CRM

Вайбкодим проект – а что с документацией?

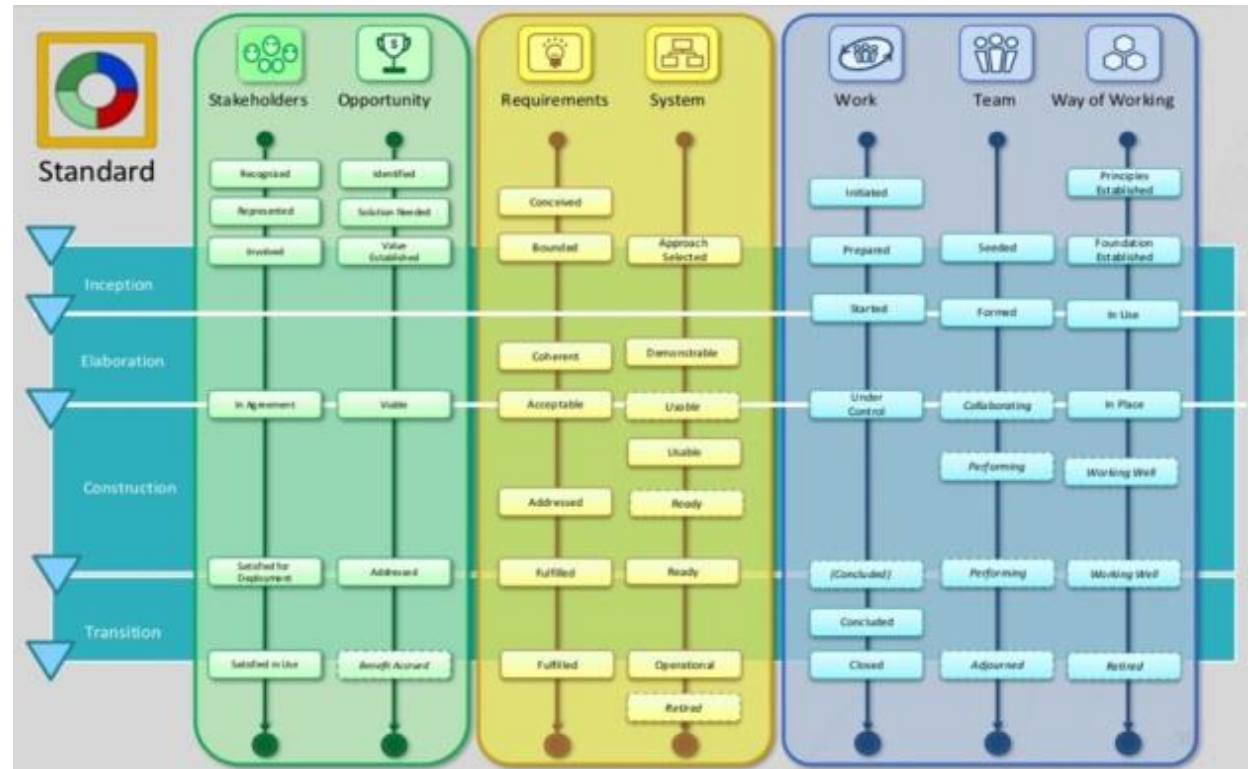
- Описание архитектуры необходимо, но может появляться не сразу
- Acceptance criteria и тест-кейсы тоже необходимы, но тоже не сразу
- Агенты хорошо порождают описания и тесты, когда есть чек-листы качества
- Должен быть способ в частном решении контролируемо нарушить правила – эксперимент или расширение функционала
- Нужен отдельный процесс: **мониторинг качества решений**, которые обеспечивает harness и его развитие – доработка шаблонов и правил

OMG Essence: фокусы внимания, а не линейный проект



Принцип: описание назначения и архитектуры, тесты и реализация развиваются параллельно и соответствуют друг другу

Нужно описание архитектуры артефактов, чек-листы и механизмы проверки каждого артефакта отдельно и проверки их соответствия между собой и с реализацией



Бизнес вайбкодит, ему – нравится!

- Вайбкодинг доступен бизнесу и он приносит прототипы вместо идей
 - С точки зрения бизнеса, задача – сделана, осталось запустить на прод
 - Он не готов обсуждать постановки, а поведение в сложных случаях – готов
- **Круглый стол по вайбкодингу на секции маркетинга на Merge-2026:**
 - Легче самому за пару вечеров сделать, чем две недели объяснять разработке
 - ИИ понимает лучше и не сопротивляется, когда требуешь переделок
- Бизнес ждет, что ИТ сможет подхватить сделанное и развивать дальше, это точно изменит процесс, но это пока не осознано

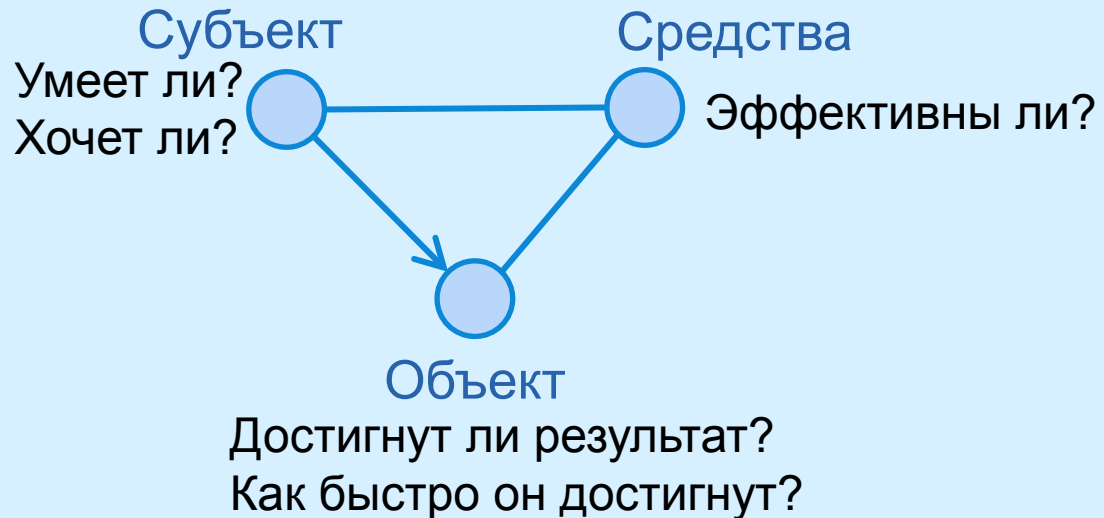
Бизнес не сможет вайбкодить все

- Развитие этой идеи – **Disrupt-PDCL** от Сбера: разработчики делают платформу, на которой бизнес сам делает приложения
- Применение – Андрей Неведин из Райффайзенбанк [«Context is a must: как мы системно управляем контекстом»](#) (Saint Highload-2026)
- Подход переносит на ИИ-агентов старую идею CASE-средств: разработка создает платформу, на которой бизнес делает приложения
- У идеи те же ограничения: платформа будет слишком сложной, и при этом все равно не поддержит необходимые кейсы, решения для которых надо искать, **решая задачу совместно с бизнесом**

Организация труда ИИ-агентов

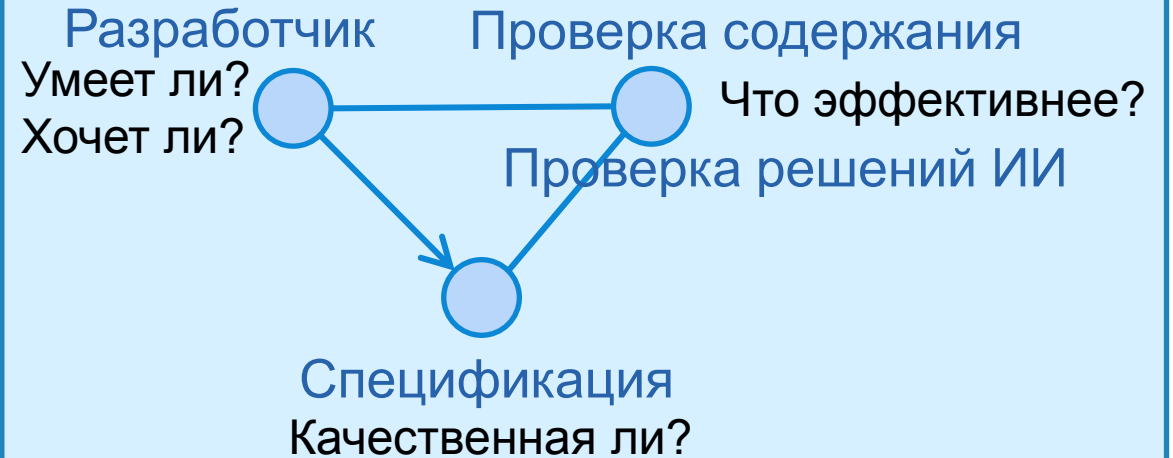
Субъект – Объект – Средства

Выполнение операции



Вопросы определяют метрики, с помощью которых мы проверяем результативность и эффективность выполнения операции

Проверка спецификации



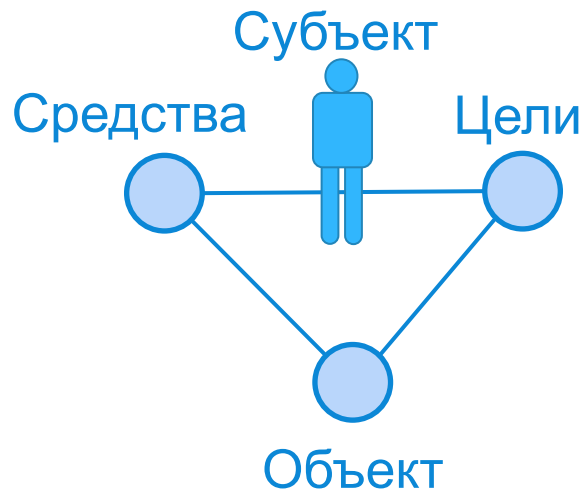
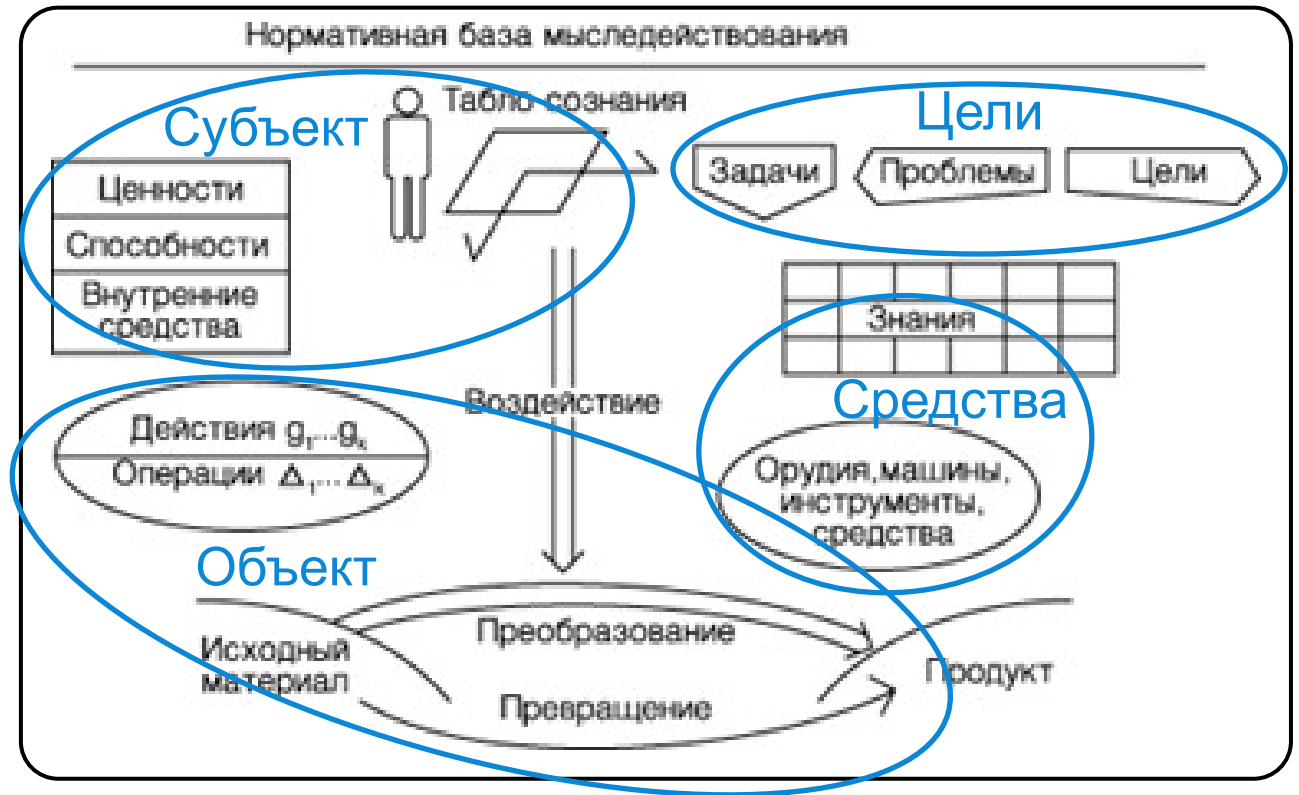
Сменив средство – передав проверку ИИ, мы повлияли на мотивацию и способность разработчика выполнить операцию – качество результата повысилось



Средство – не только LLM, среда разработки или фреймворк, но и **знания**, каким именно образом их использовать.
А для ИИ-агента – обеспечение его знаниями о проекте и методах работы (harness)

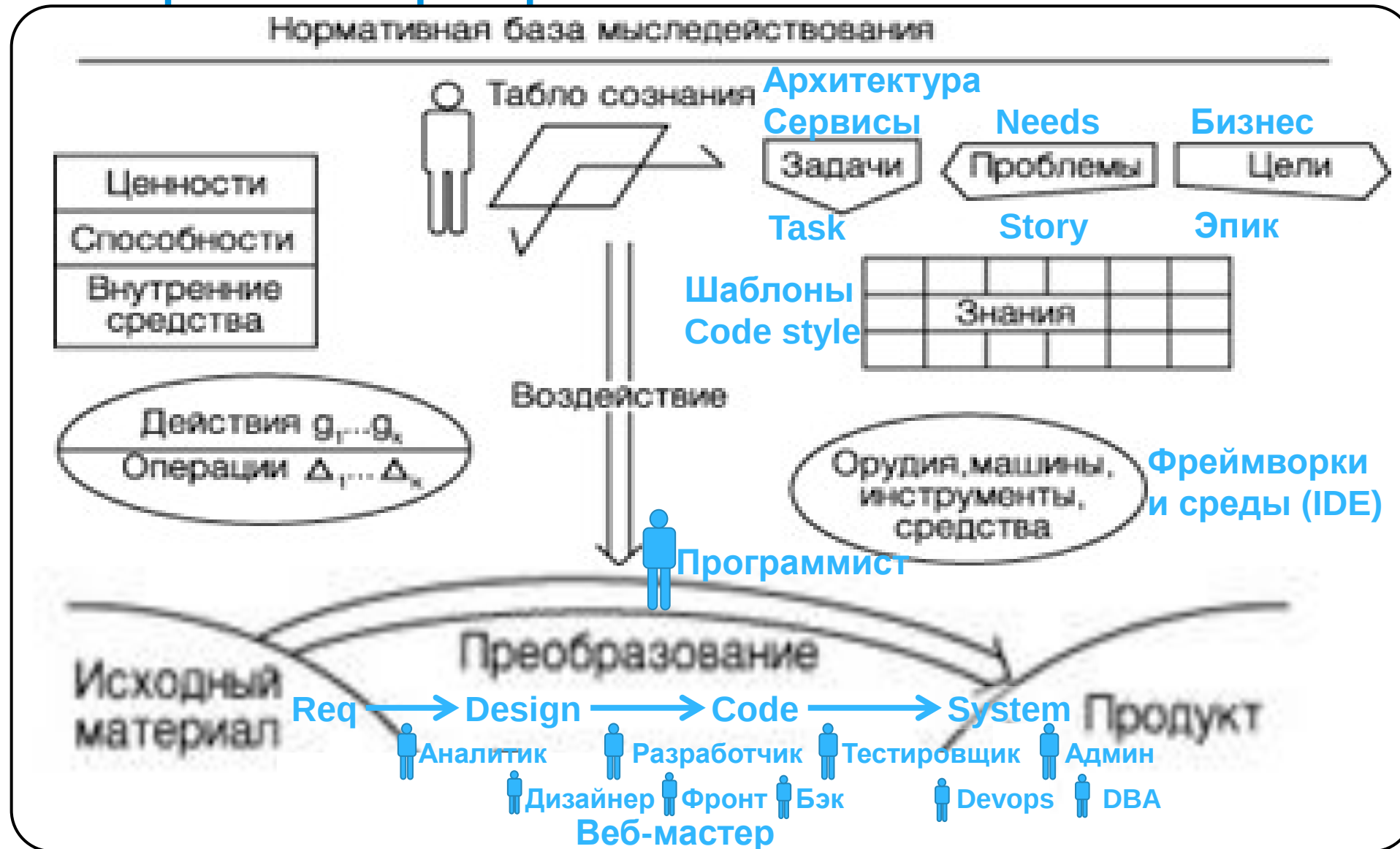
Схема Акта Деятельности (СМД-методология)

Различает структурные элементы Деятельности, это позволяет ее понимать и конструировать

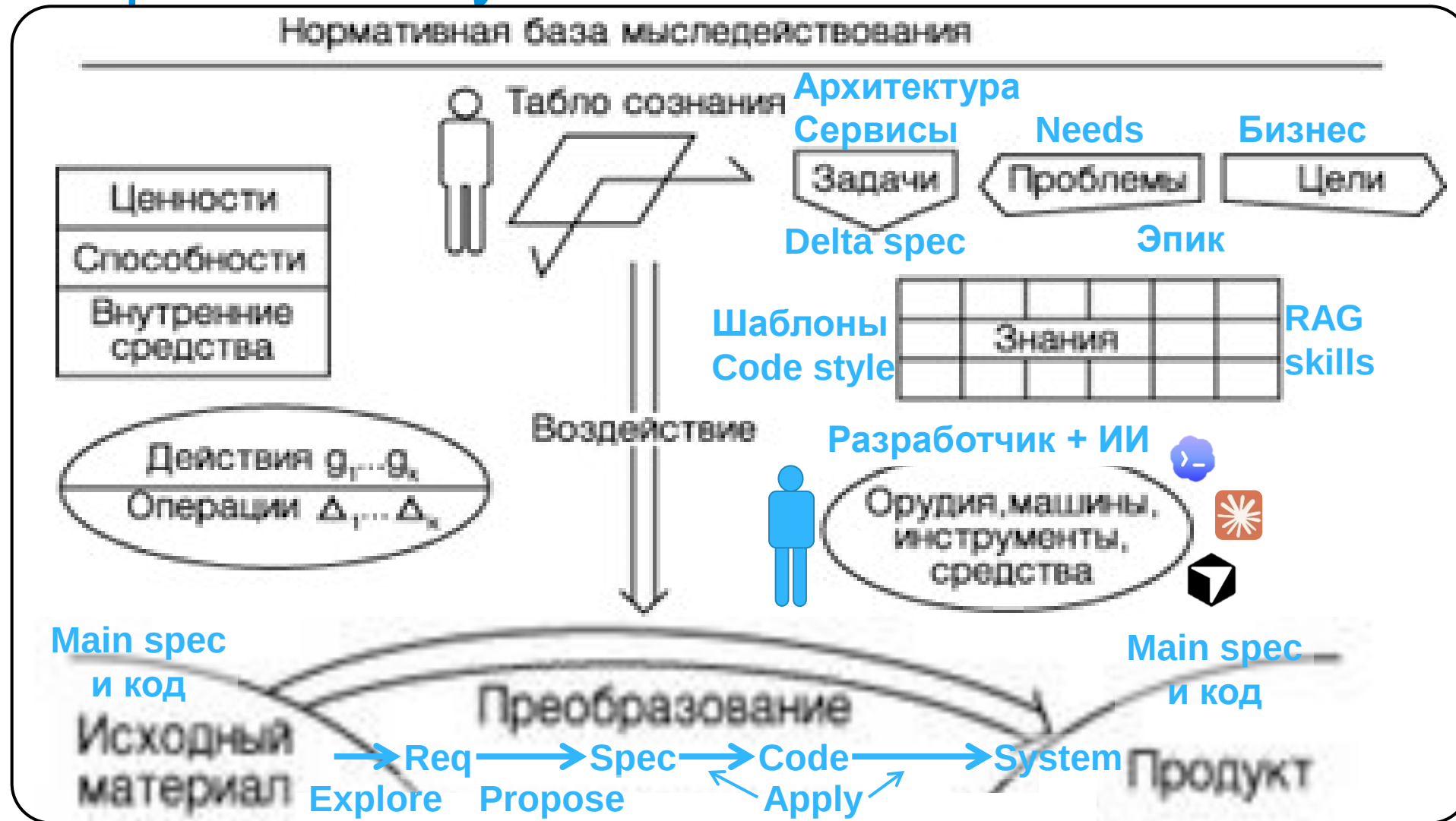


Развернута от тройки **Цели – Объект – Средства** в лекции 3 серии **Петра Георгиевича Щедровицкого** в МИСИС PUZZLES: Как современный управленец формирует картину мира: [pdf \(fondgp.ru\)](http://fondgp.ru), [youtube](https://www.youtube.com), [rutube](https://rutube.ru)

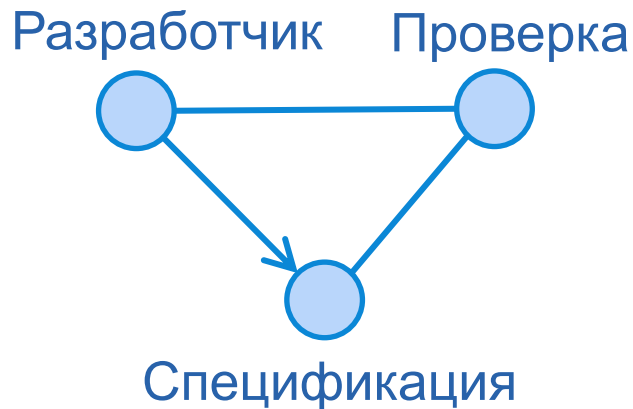
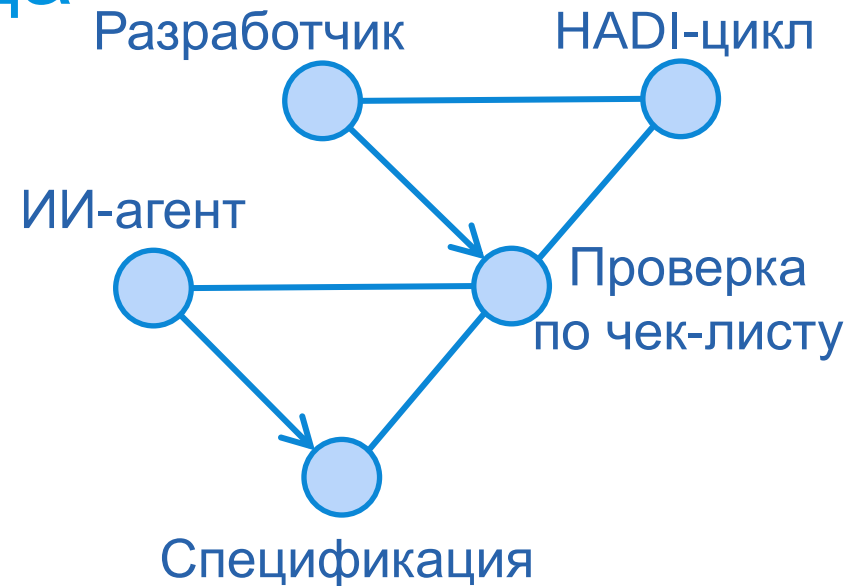
История ИТ-разработки



Разработка с участием ИИ



Вертикальное разделение труда



- Проверку разработчиком заменили проверкой ИИ-агентом по чек-листу
- Чек-лист создает разработчик, используя HADI-цикл: оценивает текущую работу и ставит эксперименты, накапливая массив задач для отладки

Вертикальное разделение труда – создание знаний для обеспечения выполнения отдельных операций, а также разделение работы на операции

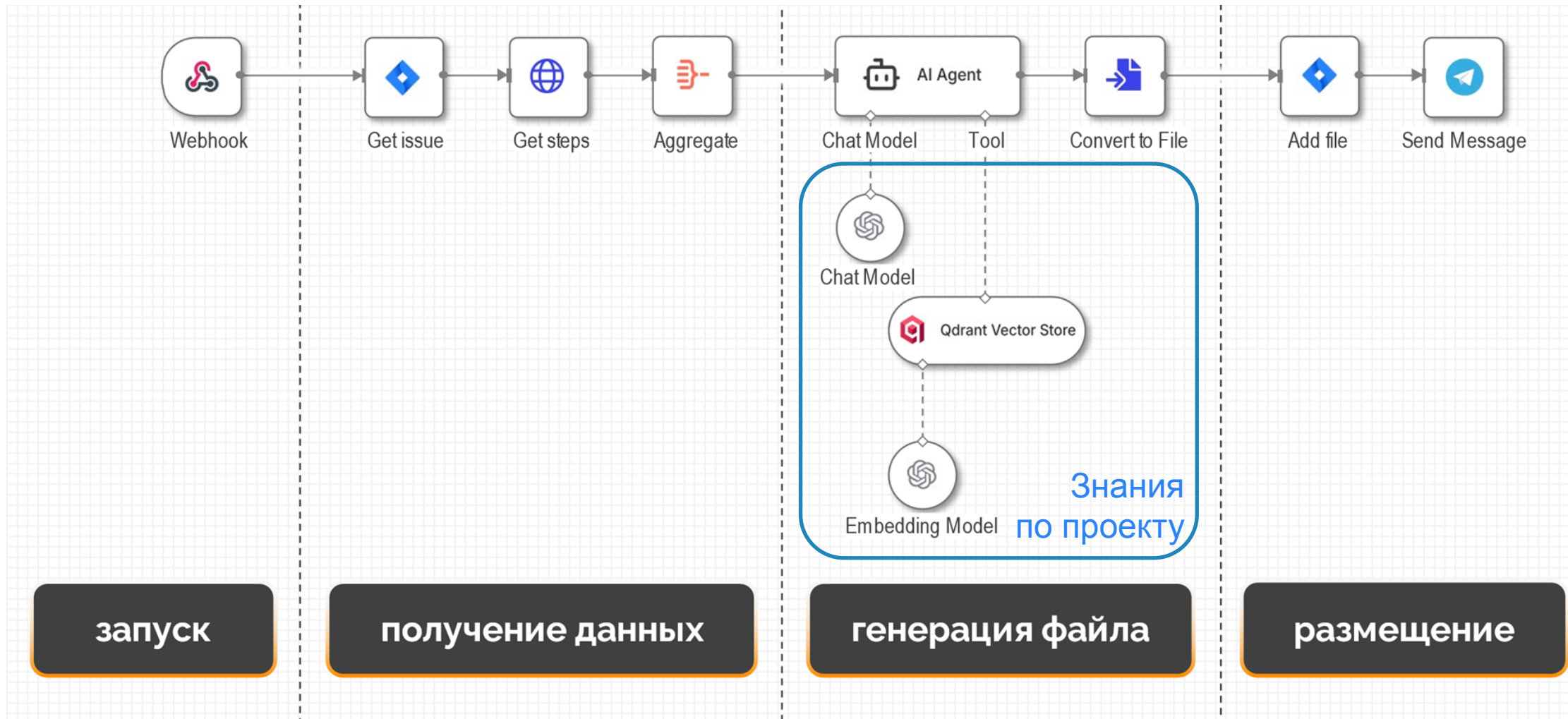
В ИТ это погружено внутрь команд, а не выделено на позицию технолога

Включение ИИ-агентов

- ИИ-агенты классно делают точечные задачи – но процесс становится сложнее, в нем больше шагов
- **Описания pipeline как workflow задач в Jira** – недостаточно, нужны средства типа **n8n** с деталями и высокой формальностью
- Агентов надо снабдить информацией, это – не только принципы и архитектура, но и **живое знание о ходе проекта** и его коде
- Доступа к коду и записям встреч – недостаточно, надо проектировать **систему артефактов** и поддерживать структурные описания в ней
- Нужен процесс **мониторинга качества решений**, которые обеспечивает harness и его развитие, но затраты на это необходимо ограничивать
- Все это касается не только разработки: бизнес будет использовать агентов в своих основных процессах, и их работу так же надо поддерживать

ИИ-агента надо снабдить знаниями

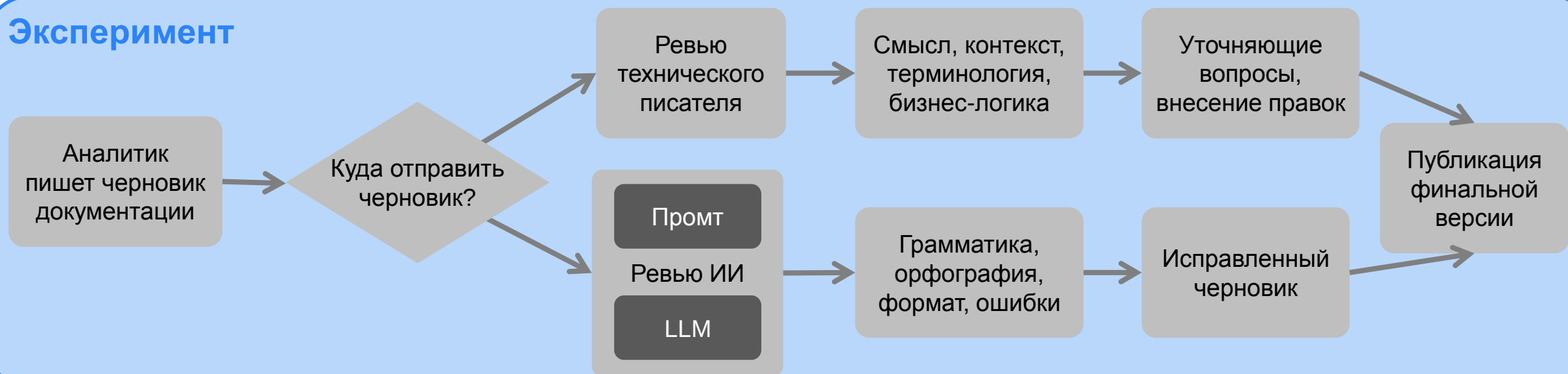
Александр Ткачев «No-code/Low-code в тестировании: ускоряем процессы с помощью n8n» (SQAdays-2026)



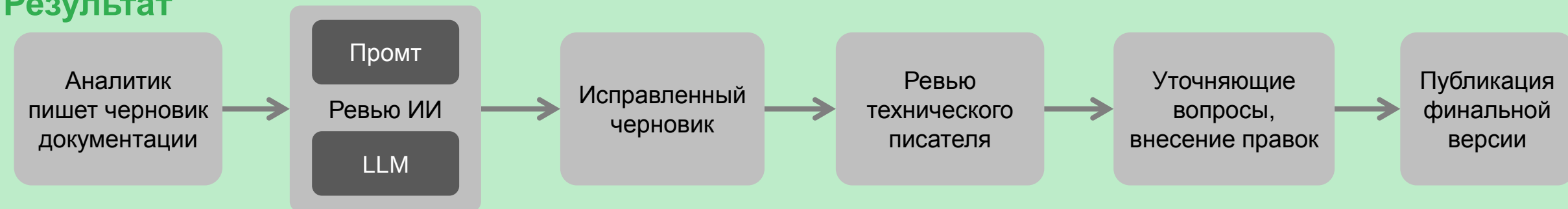
Надо адекватно оценить возможности

Камила Мазаева «Кому доверить ревью API – техпису или искусственному интеллекту» (TechWriterDays-2026)

Эксперимент



Результат



Социократия: вся деятельность – эксперименты



- Цель – снимать напряжения, а не делать задачи
- Напряжение – источник мотивации для действий
- Драйвер трактует напряжение как возможность создания ценности или препятствие, мешающее потоку ценности
- Решение – не идеально, оно лишь **достаточно хорошо**, чтобы уменьшить напряжение и **достаточно безопасно**, чтобы попробовать

Quality gate или барьер для разработки?

- ИИ-агенты хорошо делают проверки по чек-листам
- Проверки могут обеспечить качество кода, соответствие архитектурным принципам, требования безопасности, качество документации и так далее
- Проверки по чек-листам частично выносим в автомат, кодируемый агентом (правила для linter и т.п.) – это дает жесткость проверок
- Важно предусмотреть гибкие механизмы для особых случаев – раньше это обеспечивала коммуникация с ревьюером
- Нужен отдельный процесс: **мониторинг качества решений**, которые обеспечивает harness и его развитие – доработка шаблонов и правил

Отдельные агенты образуют сложную систему

[Дарья Шатко. «Как мы внедрили LLM-судей в автоматизациях клиентского сервиса» \(AIconf-2026\)](#)

е2е-пайплайн чат-бота в клиентской поддержке



Управление разработкой и место ИТ

Внедрять ИИ по площади бессмысленно

- Теория ограничений говорит: ускорение в конкретном месте только тогда дает бизнес-эффект, когда это место является ограничением для бизнеса
- Внедрение по площади приводит к тому, что **ИИ в точках ограничений** появляется **позднее**, чем в других местах – там операционка зашкаливает
- Поэтому проекты массового внедрения ИИ не дают бизнес-эффекта

Правильный подход – **фокусировка: пилоты** для апробирования, **точечные внедрения** в критических участках с ожидаемым результатом, и лишь затем – распространение



ИИ – инфраструктура, как электричество или интернет:
освоить необходимо, применять надо там, где уместно,
и он не даст конкурентного преимущества, потому что будет у всех

ИИ в разработке – лишь начало изменений

- Разработка перестала быть ограничением – надо работать с новой точкой
- Ограничением стала работа владельца продукта по созданию гипотез
- Если гипотеза проверяется за 3-5 дней вместо 2-3 месяцев, **надо перестраивать весь процесс** и роль ИТ в нем
- Раньше бизнес говорил: «куча идей, разработка не успевает», теперь вопрос к **качеству идей**
- Работу владельца продукта ИИ-помощники ускоряют меньше – пока нет технологии
- Для проверки бизнес может и будет вайбкодить прототипы – это быстрее, чем доносить идею до команды, и надо **уметь прототипы подхватывать**

Меняется процесс – меняются компетенции людей

- Проверка спецификации – совсем не то, что написание кода: писатель редко бывает хорошим редактором и наоборот
- Компетенция **быстрого кодирования прототипов** раньше была отдельной, теперь большинство работают в близком режиме
- Агенты работают долго (5-10-20 минут), на это время **человек переключается** на другую задачу – это тоже новое требование **для всех**
- Требования должны быть разумны: заранее оценить, что делает ИИ, а что лучше сделает разработчик – отдельная компетенция



Помните: человек не может все.

Если бы разработчик мог быть предпринимателем, руководителем компании – он бы не работал разработчиком

Лог работы позволяет совершенствовать процесс

- Работа сотрудников с агентами дает лог реальной деятельности, гораздо более подробный чем раньше в task-трекере, мессенджерах и почте
- Такой лог дает возможность **анализа и совершенствования процесса** и конкретных сотрудников, показывая точки возможного развития
- Об этом – интересное выступление Авенира Воронова [«AI-копилоты управления разработкой»](#) (Saint Teamlead-2026)
- Это – новое направление в системе вертикального разделения труда
- Но опыт говорит, что использовать будут не продуктивно, для контроля

Будущее: ИИ-агенты в бизнесе и жизни

- Если ты взаимодействуешь с ИИ, он знает твою жизнь: здоровье, типовые активности и задачи, и может использовать это, решая запросы
 - Подобрать обувь или зубную щетку и пасту с учетом индивидуальных особенностей
 - Планировать отпуск, выходные или вечер отдыха, опираясь на твою историю
 - Подстроить информации по профессиональным вопросам
 - Обеспечить контекстом, когда ты подхватываешь задачи коллеги
- Если выбирает и покупает ИИ-агент, маркетинг надо строить иначе
- Надо строить мост из нескольких стадий обработки, соединяя данные
 - Проектировать, строить и держать контекст твоей жизни, работы команды и компании
 - Выделять задачи для ИИ-агентов и организовывать взаимодействие в глобальной сети
 - Маркетплейсы и агрегаторы пробуют затянуть в свою сеть, а люди сопротивляются
 - Вместо универсальных таск-трекеров и CRM можно делать специализированные
- ИТ-разработка опережает в освоении ИИ и может быть драйвером

Об этом хорошие выступления у **Гарретта Джонстона**, мои конспекты:

[Живая компания-2025](#) (там есть ссылки на запись), [ПИР-2024](#), [ПИР-2025](#), [ПИР-2026](#)

ИИ в Китае: помощь и партнер, а не замена человека

- Китай активно создает ИИ-модели, **выкладывая в открытый доступ**, [программа развития до 2035 года](#) вдохновляет
- Там не боятся, что ИИ лишит работы, он помощник и партнер
 - Xiaomi: роботы будут делать нашу работу, и у нас будет больше времени
 - Миссия SenseTime: пусть ИИ возглавит наш прогресс
 - Нет проблем в том, чтобы дружить с ИИ-помощником, хотя он не человек
- ИИ не заменяет человека, а дополняет – это **дает поэтапное развитие**
 - Обучение танцам человек повторяет за учителем, а аватар показывает разницу
 - Роботы будут ухаживать за больными и стариками – уже **идет апробация**: робот помогает чем может и умеет звать на помощь, его умения можно увеличивать
 - Поставлена задача снижения цены робота до 20 тысяч долларов (сейчас 40-100)
- ИИ не глуп – он учится, и с ним надо обсуждать рабочие вопросы, чтобы улучшить решение, и чтобы сам **ИИ научился и лучше помогал**



Итоги: от вайбкодинга к agentic engineering

Joker<?>

- ИИ перестал быть просто помощником человека, а стал членом команды
- **ИИ нужен онбординг**, как любому новичку: контекст проекта и принятый способ работы – нужно создать **harness**
- Как и человека, **ИИ-агента надо оценивать и развивать**: выявлять проблемы и дорабатывать harness, чтобы работ была эффективной
- А также – **менять разделение труда** между ИИ-агентами и человеком: одни операции ИИ делает лучше, другие хуже – **перестраиваем pipeline**

Методы аналогичны применяемым для людей, но объект отличается



Максим Цепков



<http://mtsepkov.org>



[Tg: @MaximTsepkov](https://t.me/MaximTsepkov)

maks.tsepkov@ya.ru

