Для собственников и руководителей, которые запускают проект

Как ~~не~~ заказать ~~не~~ то

Как понять, что вам нужно,
и быстро проверить гипотезу

Константин Потапов

20 лет опыта создания и развития проектов

potapov.me/ru · constantin@potapov.me · @potapov_me

МИНИ-КНИГА ПЕРЕД РАЗРАБОТКОЙ

Как ~~не~~ заказать ~~не~~ то

Как понять, что вам нужно, и быстро проверить гипотезу

Константин Потапов

potapov.me/ru · constantin@potapov.me · Telegram: @potapov_me

О книге

Эта книга не про бюрократию, брифы и толстые ТЗ. Она про короткий путь от туманного «хочу» к первому проверяемому результату — до того, как деньги уйдут в разработку и переделки.

Как не заказать не то

Как понять, что вам нужно, и быстро проверить гипотезу

Автор: Константин Потапов

© Константин Потапов, 2026

Оглавление

Введение

Я разработчик. Больше двадцати лет я пишу код, запускаю продукты, чиню застрявшие проекты и вхожу в чужой хаос, когда «почти всё готово», но почему-то не работает.

Со временем стало ясно: дорогая ошибка редко начинается в коде. Обычно она начинается раньше — в момент, когда человек путает форму будущего проекта с результатом, ради которого этот проект вообще нужен.

«Хочу сайт». «Нужна автоматизация». «Нужно что-то с заявками». «Хотим личный кабинет». «Надо внедрить ИИ».

Всё это может быть правильным направлением. А может быть аккуратным, современным и дорогим способом не трогать настоящую проблему.

Эта книга не про бюрократию, брифы и толстые ТЗ. И даже не про то, как «правильно структурировать требования». Структура успокаивает, но за вас не работает.

Речь о более практичной вещи: как до разработки начать движение к результату — что делаем, кто делает, зачем, как проверяем и что не оплачиваем сейчас.

Это не теория процессов. Перед нами живой человек: у него есть деньги, боль, надежда, страх ошибиться и право уйти. Ему не нужен красиво описанный бизнес. Ему нужен разговор, после которого появляется оплачиваемое действие.

Можно заполнить десять таблиц и всё равно заказать не то. Ценность появляется не там, где у вас красивый документ, а там, где после разговора кто-то делает правильное действие и приближает результат.

Как читать эту книгу. Если у вас 15 минут — идите сразу в главу 7: там одна страница, которую можно заполнить и пустить в работу. Если у вас идея продукта — шесть вопросов в главе 4. Если завтра подписывать бюджет — пять решений в главе 8. Остальное объясняет, почему это работает и как не платить за туман.

1. Вам не нужен проект. Вам нужен сдвиг

Собственник редко просыпается с мыслью: «Мне бы заказать проект».

Обычно внутри уже что-то давит.

Заявки приходят, но менеджеры теряют часть обращений. Экспертиза есть, но клиенту трудно понять, за что платить. Команда делает руками то, что давно пора собрать в понятный контур. Продукт почти готов, но не продаётся. Есть канал, материалы, опыт, кейсы — а в продажу это не сложено.

И тогда появляется первая формулировка: «Нужен сайт», «нужна система», «нужно автоматизировать», «нужен ИИ-агент».

Первая формулировка почти всегда описывает инструмент. Она не обязана быть глупой. Она просто ранняя.

Техническая подкованность, кстати, не спасает: заказчик с опытом разработки тоже путает инструмент с целью — просто его инструмент звучит солиднее. Не «нужен сайт», а «нужен микросервис с очередью».

Сайт может быть нужен. Но если проблема в том, что услуга непонятна, сайт лишь красивее покажет туман. Автоматизация может быть нужна. Но если процесс каждый раз

придумывают заново, автоматизация закрепит хаос. ИИ может быть полезен. Но если никто не понимает, какое решение сотрудник должен принять после подсказки, модель будет производить убедительный шум.

Проект стоит начинать не с вопроса «что сделать?», а с вопроса «какой сдвиг должен произойти?».

Сдвиг — это наблюдаемое изменение в бизнесе:

- клиент быстрее понимает услугу и оставляет заявку;
- менеджер не теряет обращение между чатом и таблицей;
- отчёт собирается за двадцать минут, а не за три часа;
- собственник видит статус заказов без пяти звонков;
- новая услуга продаётся до разработки сложного сервиса;
- команда перестаёт держать критичный процесс в голове одного человека.

Вот это уже предмет разговора.

Если сдвиг не назван, разработка превращается в ставку на удачу. Можно сделать аккуратно, современно, технически правильно — и всё равно не туда.

Нормальный разговор о проекте начинается с простого вопроса: **что должно заработать лучше и какое действие делаем первым?**

Если ответа нет, рано обсуждать функции.

Сделайте сейчас: запишите одной фразой, какой сдвиг вам нужен. Без слов «сайт», «система», «приложение» и «ИИ».

2. Желание клиента нельзя презирать. Но его нельзя принимать за задачу

Желание клиента — важная энергия. Человек видит образ будущего: красивый сайт, удобный кабинет, автоматическую обработку, умного помощника, продукт, который наконец-то «сам работает».

Плохой консультант ломает это сверху: «Вам это не нужно, я умнее». После такой фразы доверие обычно заканчивается. И правильно заканчивается.

Но есть другая ошибка — принимать желание за готовую задачу.

Человек говорит: «Хочу новый сайт». За этим может стоять несколько разных деловых причин:

- заявки стали хуже;
- текущая упаковка выглядит слабее реального уровня бизнеса;
- менеджеры устали объяснять одно и то же голосом;
- клиентам непонятно, чем услуга отличается от дешёвых альтернатив;
- собственнику стыдно отправлять ссылку крупным партнёрам.

В каждом случае «сайт» будет разным. Иногда это одна посадочная страница. Иногда — новая упаковка услуг. Иногда — серия кейсов. Иногда — вообще не сайт, а коммерческое предложение, нормальный процесс первичного контакта и один понятный канал связи.

Есть и второй слой. Человек может рационально говорить про продажи, а скрыто хотеть подтверждения уровня. Представьте клиента, который пришёл на встречу обновлять сайт. Он не будет читать методичку — ему важны живой разговор, взгляд, доверие к человеку напротив. Вслух он говорит: «нужны заявки, конверсия, понятные услуги». Это правда. Но внутри есть ещё одно желание: чтобы новый клиент открыл сайт и почувствовал «вау, эти ребята сильнее остальных».

Это не глупость и не стыд. Это социальная выгода: доверие, статус, ощущение масштаба, право стоить дороже.

Ошибка — выбирать между продажами и образом. Плохой исполнитель скажет: «вам шашечки или ехать?» Хороший разложит оба слоя: сайт должен продавать и при этом создавать нужное впечатление у правильного клиента.

Хорошая работа не спорит с желанием. Она переводит его в действие, которое можно сделать и проверить.

Не так:

> Сделайте красивый современный сайт.

А так:

> После первого экрана клиент должен понять, какую дорогую проблему мы решаем, почему нам можно доверять, почему мы не дешёвый исполнитель и какое действие сделать дальше.

Это уже можно обсуждать, проверять и улучшать.

Правильная фраза перед проектом звучит спокойно:

> Я понял, чего вы хотите. Теперь давайте посмотрим, какой результат это должно дать, что сделать первым и за что пока не надо платить.

Иногда после такого разговора проект становится меньше. Иногда меняется. Иногда откладывается. Иногда, наоборот, становится ясно, что делать надо серьёзнее, чем казалось.

Во всех случаях клиент выигрывает: он перестаёт платить за чужие догадки.

Сделайте сейчас: запишите своё желание как есть, без стеснения. Рядом — что оно должно изменить в деньгах, времени или доверии клиентов.

3. Карта полезна только тогда, когда ведёт по территории

Карта — это то, как проект выглядит в голове до начала работы.

На карте всё обычно просто. Клиент оставляет заявку. Менеджер её обрабатывает. Пользователь заходит в кабинет. Система отправляет уведомление. ИИ помогает сотруднику. Данные красиво попадают в отчёт.

Территория — это как оно живёт на самом деле.

Клиент пишет не туда. Менеджер держит часть процесса в голове. Важная информация приходит голосом в Telegram. Один отчёт нужен каждый день, второй никто не открывает. Данные есть, но в трёх местах. Есть сотрудник, без которого «простая схема» перестаёт работать. Есть исключение, которое случается редко, но именно на нём обычно теряются деньги. А в некоторых отраслях территория — это ещё и то, что нельзя: персональные данные, лицензии, отраслевые системы, из которых не уйти. Такие ограничения не обсуждаются — их можно только узнать заранее или оплатить потом.

Проект дорожает не потому, что карта плохая. Карта всегда неполная. Он дорожает, когда эту неполноту принимают за реальность.

Структурное мышление говорит: «Давайте опишем разделы: цели, аудитория, функции, роли, интеграции, этапы».

Это может быть полезно. Но само по себе ничего не двигает.

Результатное мышление спрашивает иначе:

- где сейчас теряются деньги, время, данные или доверие;
- какая ошибка повторяется чаще всего;
- какой ручной обход все считают нормальным;
- какое исключение ломает красивую схему;
- кто будет пользоваться результатом и что изменится в его рабочем дне;
- как мы поймём, что стало лучше.

Разница жёсткая.

Структура может дать аккуратный документ. Работа от результата даёт действие, за которое не стыдно платить.

Вот почему я не люблю разговоры в стиле «просто сделайте как у них». У «них» виден интерфейс. Не видны ограничения, экономика, ручные обходы, роли, привычки команды, цена ошибки и причина, по которой именно такое решение у них вообще появилось.

Копировать карту без территории — удобный способ заказать не то.

Вот как это выглядит в жизни. 2013 год, крупный бренд оборудования для лодок заказывает интернет-магазин. Бюджет — 700 тысяч, три подрядчика: данные, дизайн, движок. Управление владелец отдал менеджеру проекта, а сам устранился: «мне некогда вникать, я в этих ваших сайтах не разбираюсь».

Критерии результата никто не зафиксировал — и их место тихо заняла карта из чужой территории. Менеджер спроектировала каталог по образцу любимого магазина одежды. Оттуда переехали фильтры по цвету — для лодочных запчастей. Сами запчасти легли в дерево категорий на восемь уровней вглубь. Покупателю предлагали выбирать помпу по цвету глянцевой картинки, а конкретную деталь — искать на глубине восьмого клика.

Дело не в том, что менеджер была глупа или злонамеренна. Она просто не видела территорию: не знала, сколько существует деталей, как они устроены и по каким признакам их ищет живой лодочник. Этот вопрос даже не осознавался как вопрос. И красивое здесь не враг — врагом стало красивое, которое никто не проверил использованием.

Подрядчики подстроились. Дизайн рисовали под вкус менеджера, потому что менеджер принимала работу. Движок натянули на готовый шаблон, потому что кастомную разработку делать не умели. Каждый сдал свой кусок и получил деньги. О покупателе не думал никто.

Я делал данные и дошёл до владельца с предупреждением: проект идёт не туда. Ответ был честный: «менеджер управляет, мне некогда». Свою часть я выполнил безупречно — и это не утешает до сих пор. Живого магазина не появилось. Деньги сгорели, виноватыми назначили подрядчиков, и проект поехал на второй круг — с новыми исполнителями и теми же вопросами без ответов.

Что стоило сделать иначе? Не «слушать инженера» — это лозунг. Конкретно: взять одну самую горячую категорию в продажах, заточить магазин под неё и открыть его для живых покупателей. По ценам 2013 года это стоило бы около 150 тысяч — в разы дешевле и быстрее — и сразу показало бы, находит ли человек нужную запчасть. Дальше добавлять категории с прибыли. Признаюсь: тогда я и сам так не мыслил — все работали «водопадом»: сделать всё целиком и сдать. Урок был оплачен чужими деньгами и моим временем.

И правило, которое я оттуда вынес: оценке нельзя верить. Когда человека просят оценить, он начинает играть в арт-директора — оценивает красоту и вплетает свои вкусы. Правду говорит только использование. А чтобы было чем пользоваться, нужен маленький работающий кусок, не презентация.

Перед проектом нужна не идеальная схема будущего. Нужно действие в реальности: что делаем первым, почему это даст эффект и какую часть фантазии пока не оплачиваем.

Сделайте сейчас: опишите путь одной заявки как он есть — с чатами, ручными шагами и исключением, которое все считают нормой.

4. Разработка стала дешевле. Ошибка направления — нет

Сейчас код писать проще. Модели помогают быстрее собрать черновик, сверстать экран, написать типовой кусок, найти ошибку. Низкоуровневая работа дешевеет.

Из-за этого появляется соблазн: раз делать стало быстрее, можно меньше думать перед стартом.

На практике наоборот.

Когда производство дешевеет, дороже становится выбор направления. Код можно написать быстрее. Но если вы быстрее написали не то, пользы не прибавилось. Вы просто раньше добрались до переделки.

Это главная проблема ИИ-генерации в проектах: она любит структуру вместо движения. Ей легко выдать план, список экранов, таблицу требований, дорожную карту, набор пользовательских историй. Всё выглядит убедительно. Всё похоже на работу.

Но результата может не появиться.

Например, можно быстро сгенерировать «структуру личного кабинета»: профиль, заказы, уведомления, документы, поддержка. Красиво. Только потом выяснится, что клиентам не нужен кабинет. Им нужно за тридцать секунд понять статус заказа и получить один правильный документ без звонка менеджеру.

Кабинет был структурно красивым решением. Для результата — лишним.

Раньше плохая идея часто умирала на этапе дорогой оценки: «слишком сложно, слишком долго, вернёмся позже». Сейчас её можно быстро собрать, красиво показать, не-

много доработать, ещё чуть-чуть поправить — и незаметно потратить недели на то, что не двигает бизнес.

То же часто происходит с основателем стартапа. В голове уже живёт продукт: аудитории, функции, монетизация, личная боль, будущая архитектура. И рядом тревога: «если я сейчас это упрощу, я убью замысел». Саму идею раскрывать не нужно. Важно другое — как превратить большое видение в проверяемый первый ход.

Хороший разговор в такой точке не начинается с экранов и оценки разработки. Сначала нужно развести слои: для кого первый запуск, какое одно поведение должно измениться, где сейчас больнее всего, какой опыт можно проверить за неделю вручную или полуавтоматически, а что в первую версию не попадает. После этого большая идея не исчезает. Она перестаёт быть облаком и получает первый край, за который можно взяться.

На выходе получается не «стартап целиком», а короткая карта: первый пользователь, первый сценарий, обещание результата, дешёвая проверка, критерий успеха и список того, за что пока не платим. Это и есть упаковка видения: не красивее рассказать мечту, а превратить её в следующий ход.

Как упаковать большое видение, не убив идею

Шесть вопросов. Лучше отвечать письменно и по одному, иначе мозг снова соберёт красивое облако.

1. **Чья жизнь или работа должна улучшиться первой?** Не «все, кому это полезно» — один конкретный человек.
2. **Какое одно поведение должно измениться за 7 дней** — так, чтобы человек сам сказал: «мне помогло, хочу продолжать»?
3. **Где этот человек сейчас ломается?** Один момент, не вся жизнь.
4. **Какой минимальный опыт можно проверить без продукта** — вручную, ботом, таблицей, живым оператором?
5. **Как поймём, что сработало?** Не «понравилось», а поведение: сколько дней человек возвращался, что довёл до конца, готов ли заплатить за продолжение.
6. **За что пока не платим?** Архитектура, интеграции, масштаб, красивая оболочка — это ставки, их рано оплачивать.

Если на второй вопрос ответа нет — большую систему строить рано. Если ответ есть — можно начинать проверку, и мечта впервые получает шанс стать бизнесом, а не презентацией.

Как выглядит переход:

> **Было:** «Хочу платформу, которая изменит то, как малый бизнес управляет финансами».
>
> **Стало:** «За 7 дней проверяем на пяти владельцах кофеен: если каждый вечер присылать им одну цифру и один вопрос, начнут ли они принимать решения по деньгам — и попросят ли продолжения».

Идея не стала меньше. У неё появился первый край, который можно проверить.

Поэтому ценность разработчика смещается. Просто написать функцию — всё менее редкая способность. Гораздо важнее понять, какая функция вообще нужна, какой первый

шаг даст эффект, где готовый сервис лучше разработки, где нужен прототип, а где проект надо остановить до того, как он начнёт есть бюджет.

Я не хочу продавать клиенту много работы. Я хочу найти место, где моя работа создаст заметно больше ценности, чем стоит.

Для этого сначала нужна не структура документа, а цепочка:

> боль → нужный результат → первое действие → дешёвая проверка → видимый эффект.

Если этой цепочки нет, любая спецификация украшает туман.

Сделайте сейчас: ответьте письменно на шесть вопросов выше. Застряли на втором — значит, платить за архитектуру рано.

5. Холостой ход: когда все поговорили, но проект не сдвинулся

Есть вид встреч, после которых всем кажется, что работа идёт.

Обсудили стратегию. Поговорили про сайт, продукт, автоматизацию, ИИ, личный кабинет. Нарисовали несколько стрелок. Вспомнили конкурентов. Сказали «надо упаковать», «надо упростить», «надо сделать удобно». Разошлись с ощущением движения.

А потом выясняется, что никто не может ответить на простые вопросы:

- что делаем первым;
- зачем именно это;
- какой результат считаем полезным;
- где границы;
- что не делаем;
- какой риск главный;
- какое решение надо принять сейчас.

Это не работа. Это холостой ход.

Мотор шумит. Топливо горит. Люди заняты. Машина стоит.

Холостой ход приятен, потому что создаёт ощущение заботы о проекте. Но собственнику от него обычно не легче. Время ушло, внимание потрачено, ясности мало, решения нет.

Если его не прервать, финал обычно похож. Декорации меняются, механизм — нет.

Компания устала терять заявки и решила: нужна CRM под себя. Подрядчик приехал, послушал, покивал и посчитал функции: карточки клиентов, воронка, права, отчёты, интеграция с телефонией, мобильная версия. Смета — полтора миллиона, срок — четыре месяца. Все довольны: клиент — что его услышали, подрядчик — что посчитал.

Встречи шли тепло. Обсуждали интерфейсы, спорили о цветах статусов, добавляли «а ещё было бы удобно». Один вопрос не прозвучал ни разу: какой результат мы хотим увидеть первым и когда?

Через восемь месяцев — сроки поехали, они почти всегда едут, когда нет границ, — система была готова. Формально. Менеджеры в неё не пошли: заявки по-прежнему летели в WhatsApp, потому что так быстрее, а обязательное поле «источник заявки» заполняли дефисом, лишь бы карточка сохранилась. Половина функций не открылась ни разу. Полтора миллиона превратились в интерфейс, которым никто не пользуется, и в усталый вопрос «и что теперь?».

Виноватых искали долго. Подрядчик показывал подписанное ТЗ: всё сделано по списку. Клиент показывал пустую систему: пользы нет. Формально правы оба. Проиграли оба.

Остановиться надо было до разработки — на вопросе, который так и не прозвучал. Если бы его задали, ответ был бы: «менеджер не теряет заявку между чатом и таблицей». И быстро выяснилось бы, что для этого не нужна система за полтора миллиона. Нужен один канал приёма заявок и простое правило работы с ним. Такая проверка заняла бы пару недель и почти не стоила денег. А уже на живом процессе стало бы видно, какая система нужна на самом деле — и нужна ли вообще.

Система есть, пользы нет — это не невезение. Это оплаченный входной туман.

Работа отличается от холостого хода не количеством обсуждённых тем, а изменением состояния проекта.

До разговора было: «нам нужен сайт».

После разговора должно стать, например:

> Нам нужна страница для дорогой услуги. Первый экран должен отсечь дешёвые заявки и объяснить, за что платят. До дизайна собираем три кейса, формулируем критерии подходящего клиента и проверяем предложение на пяти живых разговорах.

Или так:

> Нам не нужен личный кабинет в первой версии. Клиентам нужен статус заказа и документы. Делаем уведомления, единый канал связи и простую страницу статуса. Кабинет вернём в обсуждение, если появятся повторные сценарии.

Или так:

> ИИ здесь пока не нужен. Сначала убираем ручной перенос данных между таблицей и CRM. Это даст эффект быстрее и снизит количество ошибок без новой магии.

Вот это результат разговора. Не потому что появился документ, а потому что кто-то теперь знает, что делать.

Главный признак пользы простой: после встречи начинается действие, которое уменьшает неопределённость или создаёт ценность.

Если этого нет, вы не двигаете проект. Вы гладите его по голове.

Сделайте сейчас: после ближайшей встречи по проекту запишите одно предложение: что теперь сделаете точнее. Не пишется — встреча была холостым ходом.

6. Как запустить первый результат без анкеты на сто вопросов

Перед проектом не нужно отвечать на сто вопросов. Это утомляет и быстро превращается в анкету ради анкеты.

Достаточно короткого маршрута из пяти шагов.

Шаг 1. Назвать результат

Не «сделать сайт», а «получать заявки от клиентов, которые уже понимают услугу и примерный бюджет».

Не «внедрить ИИ», а «сократить подготовку отчёта с трёх часов до двадцати минут».

Не «автоматизировать», а «убрать ручной перенос данных, из-за которого теряются заказы».

Пока результат не назван, вы обсуждаете форму.

Шаг 2. Посмотреть, как всё работает сейчас

Без красивой схемы. Как есть.

Кто что делает. Где ждут. Где ошибаются. Где клиент уходит. Где сотрудник держит процесс на себе. Где данные переписываются руками. Где все говорят «ну это редко», хотя именно там потом горит бюджет. И где ограничения — не привычка, а закон: персональные данные, отраслевые правила, системы, из которых нельзя уйти.

Шаг 3. Отделить желаемое от необходимого

Желаемое сохраняем как образ направления. Необходимое попадает в первый шаг.

Клиент может хотеть личный кабинет. Не спорим. Но в первую версию может попасть не кабинет, а статус заказа, уведомления и понятная выдача документов. Если это закрывает боль, проект стал точнее.

Шаг 4. Запустить дешёвую проверку

Не всё надо сразу строить.

Иногда достаточно прототипа. Иногда таблицы. Иногда посадочной страницы. Иногда разбора текущего процесса. Иногда готового сервиса на месяц вместо разработки на полгода. Иногда пяти разговоров с клиентами, после которых становится ясно, что предложение непонятно.

Дешёвая проверка нужна не для моды и не ради отчёта о «тестировании гипотез». Она нужна, чтобы начать движение и не оплатить большую ошибку.

И одно правило: проверка — это использование, а не опрос. Когда человека просят оценить, он оценивает красоту и вкус. Правду говорит только использование.

Шаг 5. Сформулировать первый результат

Первый результат — не «готова система». Это слишком крупно.

Лучше так:

- заявки не теряются между каналами;
- клиент понимает услугу до звонка;

- менеджер видит статус без ручного опроса;
- отчёт собирается из одного источника;
- собственник может принять решение о следующем этапе.

Так маршрут выглядит, когда сработал. Владелица онлайн-школы хотела «личный кабинет как у больших». После пяти шагов первым результатом оказался не кабинет, а статус оплаты и расписание в один клик из письма. Сделали за три недели вместо полугода. Жалобы «где моё расписание» исчезли, администратор перестал отвечать на одни и те же вопросы, а кабинет вернулся в план — уже с ясным списком того, что в нём должно быть и зачем. Мечта не умерла. Она дождалась своей очереди.

Хорошая подготовка перед стартом отвечает не на вопрос «какие функции нужны?». Она отвечает на другой: **что делаем первым, чтобы получить пользу и не закопаться?**

7. Одна страница перед проектом

Если после книги у вас останется только один лист, пусть это будет он.

Заполните его до разговора с подрядчиком, командой или внешним инженером. Не идеально — честно.

1. Какой результат нужен?

Что должно стать быстрее, понятнее, дешевле, надёжнее или прибыльнее?

Плохой ответ: «сделать сайт». Лучше: «получать более подготовленные заявки на дорогую услугу».

2. Что сейчас мешает?

Где теряются деньги, время, данные, доверие или внимание?

Плохой ответ: «старый интерфейс». Лучше: «клиент не понимает, чем мы отличаемся, и уходит сравнивать по цене».

3. Как это работает сегодня?

Кто что делает руками, в таблицах, чатах, голосовых, голове одного сотрудника?

Не рисуйте идеальную схему. Опишите реальность.

4. Что мы хотим как образ?

Красивый сайт, кабинет, ИИ, автоматизация, приложение, портал — назовите желание прямо. Его не надо стыдиться.

5. Что нам нужно как первый результат?

Что из желания действительно должно попасть в первую версию, чтобы появился эффект?

6. Какую дешёвую проверку запускаем?

Прототип, посадочная страница, таблица, интервью, готовый сервис, ручной тест, разбор процесса, пилот на одном отделе.

7. Чего мы сознательно не делаем сейчас?

Это главный вопрос против расползания проекта.

Если вы не знаете, что не делаете, первая версия начнёт притворяться всем проектом сразу.

Кто что делает с этой страницей

Её не обязан заполнять собственник в одиночку.

Собственник решает — и это нельзя делегировать: какой результат нужен (вопрос 1), что оставляем как образ и что берём в первый результат (вопросы 4 и 5), чего сознательно не делаем (вопрос 7). Это ставки владельца, а не технические детали.

Помощник или координатор собирает: как всё работает сегодня (вопрос 3) — по разговорам с командой; что мешает (вопрос 2) — как список наблюдений, который владелец подтвердит; и черновик всей страницы целиком.

У команды спрашиваем: где процесс держится на ручных действиях, где чаще всего ошибаются, какие исключения случаются редко, но обязательны.

Подрядчику показываем заполненную страницу целиком — вместо «посчитайте нам систему». По его вопросам быстро видно, с кем вы имеете дело: кто уточняет результат, а кто сразу считает функции.

Руководитель отдела заполняет страницу так же, как собственник, с одним отличием: ответы на вопросы-ставки он несёт наверх как предложение, а не как принятое решение. Страница плюс открытые вопросы — уже черновик записки для генерального.

Итог — одна страница: семь ответов и список открытых вопросов. С ней можно идти к руководству защищать бюджет, к подрядчику за оценкой или к внешнему инженеру за проверкой направления.

Эта страница не заменяет разработку, дизайн или аналитику. Она делает другое: переводит разговор в действие и показывает, где вы знаете, а где угадываете.

Если после заполнения у вас появились три жёстких вопроса — лист сработал. Он не должен успокоить. Он должен вынести туман на стол.

8. Что должно быть на руках перед оплатой реализации

Перед тем как платить за реализацию, вам нужен не огромный документ, а ясность по нескольким вещам.

Какой результат нужен. Почему это важно сейчас. Как всё работает сегодня. Где теряются деньги, время или внимание. Что делаем первым. Что точно не делаем в первой версии. Как увидим пользу.

Этого часто достаточно, чтобы резко снизить риск.

Не потому что все детали известны. Они не будут известны. Любой живой проект откроет что-то по дороге.

Есть разница между нормальной рабочей неопределённостью и неясностью на старте. Первая управляется итерациями. Вторая превращает каждую итерацию в спор о том, что вообще имелось в виду.

Перед оплатой реализации должны быть зафиксированы хотя бы пять решений:

1. **Результат:** что должно заработать лучше.
2. **Действие:** что запускаем первым.
3. **Проверка:** как быстро увидим, что направление верное.
4. **Риск:** что может сломать проект раньше всего.
5. **Ответственный шаг:** кто что делает после встречи.

Обратите внимание: это не разделы ради разделов. Это решения перед стартом. Если их нет, документ может быть толстым, но проект всё равно пустой.

Если бюджет не ваш — вы руководитель отдела и защищаете его перед генеральным, — эти пять решений и есть ваша записка. Не «нужна система за столько-то», а: вот результат, вот первый шаг, вот проверка, вот риск, вот кто что делает. Такой разговор защищает не только бюджет, но и вас: решение принято не в одиночку и не на глазок.

Если после подготовки вы можете объяснить проект простыми словами, назвать первый полезный результат и сказать, чего пока не делаете, вы уже впереди большинства заказчиков.

Если не можете — рано платить за разработку.

9. Когда стоит звать внешнего человека

Иногда собственник сам может пройти этот маршрут. Особенно если проект небольшой, а проблема хорошо видна.

Но внешний инженерный взгляд полезен, когда внутри слишком много привычного.

Вы уже не замечаете ручные костыли. Команда считает странные обходы нормой. Желание «сделать красиво» заслоняет вопрос пользы. Разговоры идут по кругу. Подрядчики называют разные цены, а сравнить их невозможно, потому что каждый посчитал свой кусок тумана.

В такой ситуации внешний человек нужен не для терминологии и не для продажи разработки любой ценой.

Его работа — помочь сделать полезное действие точнее:

- где реальная боль;
- где лишнее;
- где риск;
- где быстрый выигрыш;
- где готовое решение лучше кастомной разработки;
- где без инженерной работы действительно не обойтись.

Хороший внешний человек не приносит «схему процесса» как самоценность. Он помогает сделать правильный первый шаг дешевле: до разработки, до договора на полгода, до спора с подрядчиком, до переделки.

Есть и обратная сторона: внешний человек в проекте есть, а запроса на результат нет.

Я делал систему управления большим мероприятием на несколько тысяч участников: расселить, встретить, накормить, сопровождать. Вокруг был огромный штат, который занимал себя модными решениями, не закрывающими суть. Это было терпимо: полно-

мочий резать лишнее у меня хватало. Настоящая проблема была глубже. Данные жили в разных отделах — у бухгалтерии счета и суммы, у оргкомитета фамилии, — а запроса на прозрачную общую картину в проекте на самом деле не было.

Я сделал свой этап, передал работающую систему расселения и вышел. После моего ухода эту часть переделали заново — по моей оценке, в разы дороже рыночной цены и без работавшей функциональности. Официально всё прошло успешно.

Из той истории у меня осталась заноза: моя часть была выполнена безупречно, но живого результата из неё не выросло — значит, я работал зря. С тех пор я меряю свою работу не сданным этапом, а тем, живёт ли проект.

Есть исполнители, которые зарабатывают на том, что проекты сдаются. Я хочу зарабатывать на том, что они потом живут.

В этом нет святости. Контракт мне выгоден, как и любому исполнителю, и деньги я считать умею. Просто мёртвый проект — плохой бизнес для обоих: клиент теряет бюджет, а я — репутацию и проекты, которые приводит только живой результат.

Отсюда правило про ответственность. Менеджер хочет построить мост дешевле — это нормально, экономить его работа. Но если мост рухнет, отвечать будет инженер. Поэтому инженер обязан отстаивать своё мнение, даже когда это неудобно и стоит ему контракта. Подрядчик, который со всем соглашается, — не опора. Опереться можно только на того, кто умеет сказать «нет» и готов выйти, если проект перестал идти к результату. Выход с зафиксированным этапом и переданной работающей частью — тоже честный результат. Дороже обходится тот, кто остаётся и молчит.

Это экономит не только деньги. Это экономит внимание собственника. А внимание обычно дороже, чем кажется.

Финал. Не платите за туман

Каждая непринятая развилка всё равно будет оплачена.

Вопрос только когда.

До старта — короткой работой: какой результат нужен, что делаем первым, кто делает, где границы, как проверяем пользу.

Или после старта — переделками, затянутыми созвонами, спорными переписками и усталой фразой «мы думали, это очевидно».

Не было очевидно.

Если вы планируете проект, начните не с вопроса «сколько стоит разработка?». Начните с вопроса «что мы на самом деле хотим изменить?».

Не принимайте красивую структуру за ясность. Не принимайте сгенерированный план за решение. Не принимайте список функций за движение к результату.

Сначала результат. Потом действие. Потом проверка. Потом деньги. Потом реализация.

Так спокойнее. И дешевле.

Следующий шаг

Возьмите одну страницу из главы 7 и заполните её сами или с командой. Если на каком-то вопросе вы зависли — это не провал. Это место, где проект мог съесть деньги позже.

Если нужен внешний инженерный взгляд — напишите мне. За одну-две встречи соберём карту первого результата: что делаем, что не делаем, как проверяем пользу и где разработка действительно нужна.

Не чтобы продать вам код любой ценой. А чтобы найти точку, где работа действительно создаёт ценность.

https://potapov.me/ru constantin@potapov.me Telegram: @potapov_me

Приложение 1. План первого результата перед проектом

Заполняется до разработки. Цель — не написать идеальное ТЗ и не разложить проект по красивым разделам, а запустить первый результат и не оплатить лишнее.

Сначала заполняется блок «Запуск результата». Остальные разделы — техническое приложение: используйте их только там, где ответ меняет действие, снижает риск или приближает результат.

0. Запуск результата

Заполните до деталей про роли, данные и интеграции. Черновик всего листа может собрать помощник; пометки показывают, чей ответ финальный.

Какой результат нужен бизнесу? *(решает собственник)*

> ...

Что сейчас мешает этому результату? *(собирает помощник по разговорам с командой, подтверждает собственник)*

> ...

Какой первый результат даст пользу? *(решает собственник)*

> ...

Какую дешёвую проверку запускаем? *(предлагает помощник или подрядчик, утверждает собственник)*

> ...

Что сознательно не делаем в первой версии? *(решает собственник)*

> ...

1. Цель проекта

Что должно измениться после запуска?

- ☐ больше заявок
- ☐ меньше ручной работы
- ☐ быстрее обработка
- ☐ меньше ошибок
- ☐ понятнее контроль
- ☐ новый канал продаж
- ☐ другое: ...

Главный измеримый результат:

> ...

2. Текущий процесс

Как это работает сейчас, шаг за шагом?

1. ...
2. ...
3. ...

Где сейчас больнее всего:

> ...

3. Пользователи и роли

- Клиент:
- Менеджер:
- Администратор:
- Руководитель:
- Бухгалтерия:
- Партнёр:
- Другое:

4. Основные сценарии

Какие 3–5 сценариев должны работать в первой версии?

1. ...
2. ...
3. ...

5. Данные и объекты

С чем работает система?

- ☐ заявки
- ☐ клиенты
- ☐ заказы
- ☐ оплаты
- ☐ документы
- ☐ товары/услуги
- ☐ отчёты
- ☐ другое: ...

6. Статусы

Какие состояния бывают у главных объектов?

Пример: новая заявка → в работе → ожидает ответа → оплачено → закрыто → отказ.

Ваш процесс:

> ...

7. Права доступа

Кто что может видеть и менять?

- Кто видит всё?
- Кто видит только свои данные?
- Кто может удалять?
- Кто может менять статус?

- Какие действия опасны без ограничения прав?

8. Интеграции

Какие внешние системы нужны?

- ☐ сайт
- ☐ CRM
- ☐ платёжная система
- ☐ Telegram
- ☐ email
- ☐ телефония
- ☐ 1С / бухгалтерия
- ☐ склад
- ☐ другое: ...

9. Исключения

Что бывает редко, но обязательно должно работать?

- ...

10. Что входит в первую версию

- ...

11. Что не входит в первую версию

- ...

12. Критерии приёмки

Как понять, что первая версия готова?

- ...

13. Открытые вопросы

- ...

Приложение 2. Вопросы перед проектом

Цель этих вопросов — не собрать красивую структуру требований, а запустить действие: что делаем, кто делает, зачем, как проверяем результат и что не оплачиваем сейчас.

У каждого раздела указан адресат: у кого спрашивать. Черновик ответов может собрать помощник или координатор; решения-ставки остаются за собственником.

Одна страница перед проектом — решает собственник, черновик собирает помощник

1. Какой результат нужен бизнесу?
2. Что сейчас мешает этому результату?
3. Как это работает сегодня — руками, в Excel, в чатах, в головах людей?
4. Где сейчас теряются деньги, время, данные, доверие или внимание?
5. Что мы хотим как образ — и что нам нужно как первый результат?
6. Какую дешёвую проверку запускаем до большой разработки?
7. Что мы сознательно не делаем в первой версии?

Дополнительные вопросы перед реализацией

Эти вопросы не нужно проходить сразу все. Это приложение на случай, когда решение двигаться дальше уже принято и нужно проверить, где проект может потерять деньги, сроки или управляемость.

Главные вопросы — обсуждают собственник и руководитель проекта

1. Что должно измениться в бизнесе после запуска?
2. Почему это важно сейчас?
3. Что будет, если ничего не делать ещё 3–6 месяцев?
4. Как процесс работает сегодня, шаг за шагом?
5. Где сейчас теряются деньги, время, заявки или контроль?
6. Какое решение нужно принять после этого разговора?
7. Кто участвует в процессе?
8. Кто принимает финальные решения?
9. Кто будет пользоваться системой каждый день?
10. Кто будет принимать работу у подрядчика?
11. Что будет считаться успешной первой версией?

Про пользователей и роли — спрашивать у команды

1. Какие роли нужны?
2. Кто что должен видеть?
3. Кто что может менять?
4. Какие действия должны быть запрещены для части пользователей?
5. Где ошибка в правах доступа может стоить денег или репутации?

Про данные — спрашивать у команды

1. Какие данные уже есть?
2. Где они сейчас хранятся?
3. Какие поля обязательны?
4. Какие данные часто вводят с ошибками?
5. Что нужно импортировать из старых систем или таблиц?

Про сценарии — спрашивать у команды, приоритеты подтверждает собственник

1. Какие 3 сценария самые частые?
2. Какой сценарий самый денежный?
3. Какой сценарий самый рискованный?
4. Какие действия нельзя отменить?
5. Какие исключения происходят редко, но обязательны?

Про интеграции — спрашивать у команды, проверять с подрядчиком

1. С какими системами нужно обмениваться данными?
2. Что делать, если интеграция недоступна?
3. Нужны ли уведомления? Кому, куда и когда?
4. Какие отчёты реально используются?
5. Какие отчёты нужны только «на всякий случай»?

Про первую версию — решает собственник

1. Что обязательно должно войти в первую версию?
2. Что можно отложить?
3. Какие функции кажутся желательными, но не критичными?
4. Что нельзя делать в первой версии, чтобы не раздуть бюджет?
5. Какие решения нужно проверить прототипом до разработки?

Финальный контроль — вместе с подрядчиком или внешним инженером

1. Какие вопросы ещё остались неясными?
2. Где наш ответ может изменить архитектуру?
3. Где наш ответ может изменить бюджет?
4. Где наш ответ может изменить сроки?
5. Что клиент считает очевидным, но подрядчик может не знать?