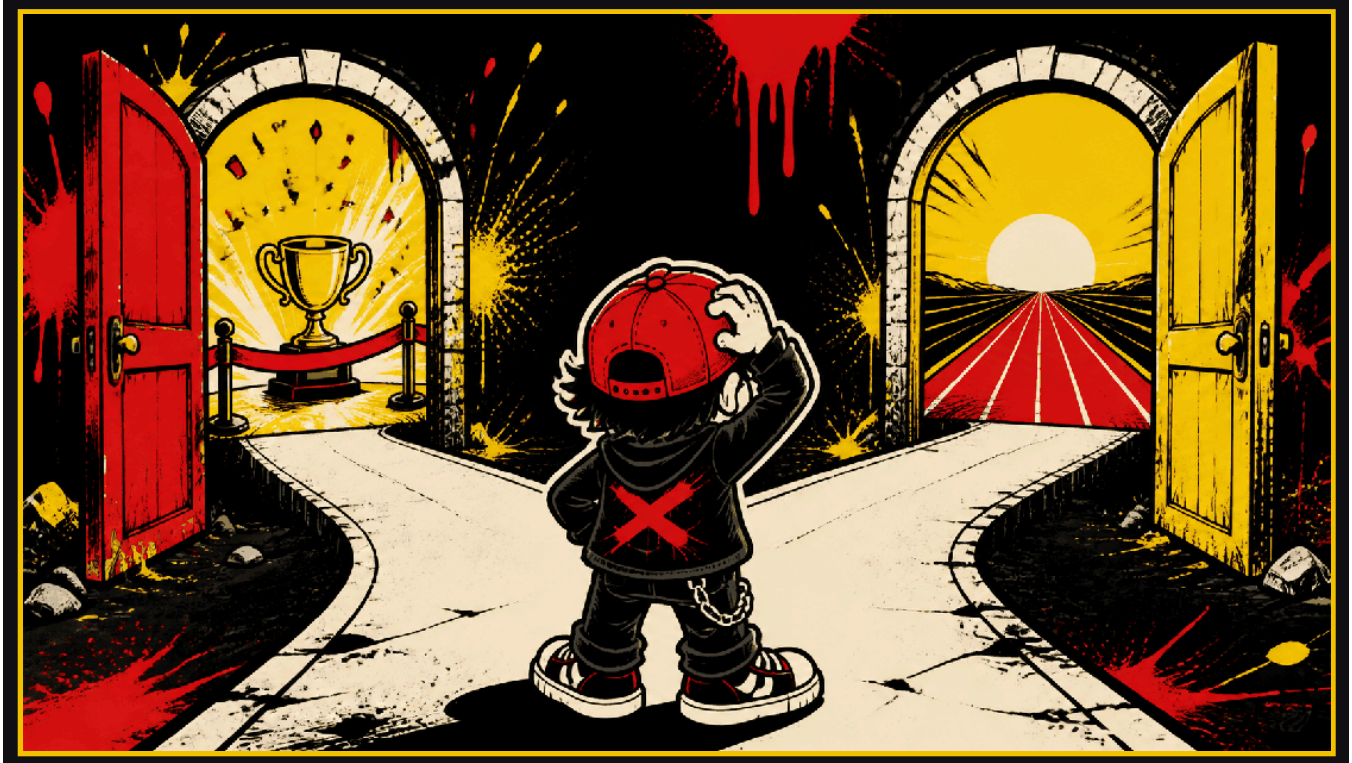


КАК ВЕСТИ ПРОЕКТЫ В НЕБОЛЬШОЙ КОМАНДЕ

Хендбук для тимлидов Ветменеджера · версия 1.1

Ветменеджер · 21.08.2026 · источник: workspace/vm/leads-course/handbook.md

Содержание

Зачем это 🤘

1. Что такое проект 🏁

2. Треугольник ограничений 🥊

3. Пять стадий ⚡

4. PMBOK за три минуты 🧠

5. Наш набор 🧰

6. Устав — «зачем и для кого» 📜

7. Scope — что делаем и, главное, чего не делаем 🪓

8. DoD — когда считаем, что готово 🏴

9. Сроки 🎰

10. Бэклог 🗃️

11. RAID — одна таблица вместо четырёх документов 🐉

12. Журнал решений 🕵️

13. Weekly Status 📡

14. Полномочия и эскалация 🚨

15. Закрытие и ретро ⚰️

16. Частые косяки 💀

17. Шпаргалка 🎯

Зачем это 🤘

Ты ведёшь проекты. Даже если тебя никто так не называл и в должностной инструкции этого слова нет.

Разделить линии поддержки. Свозить команду на конференцию. Запустить экспериментальную фичу. Собрать учебную базу для стажёров. Привести в порядок базу знаний. Это всё проекты. И каждый сейчас ведёт их по-своему, потому что способ изобретается заново — и шишки набиваются тоже заново.

Текст даёт две вещи.

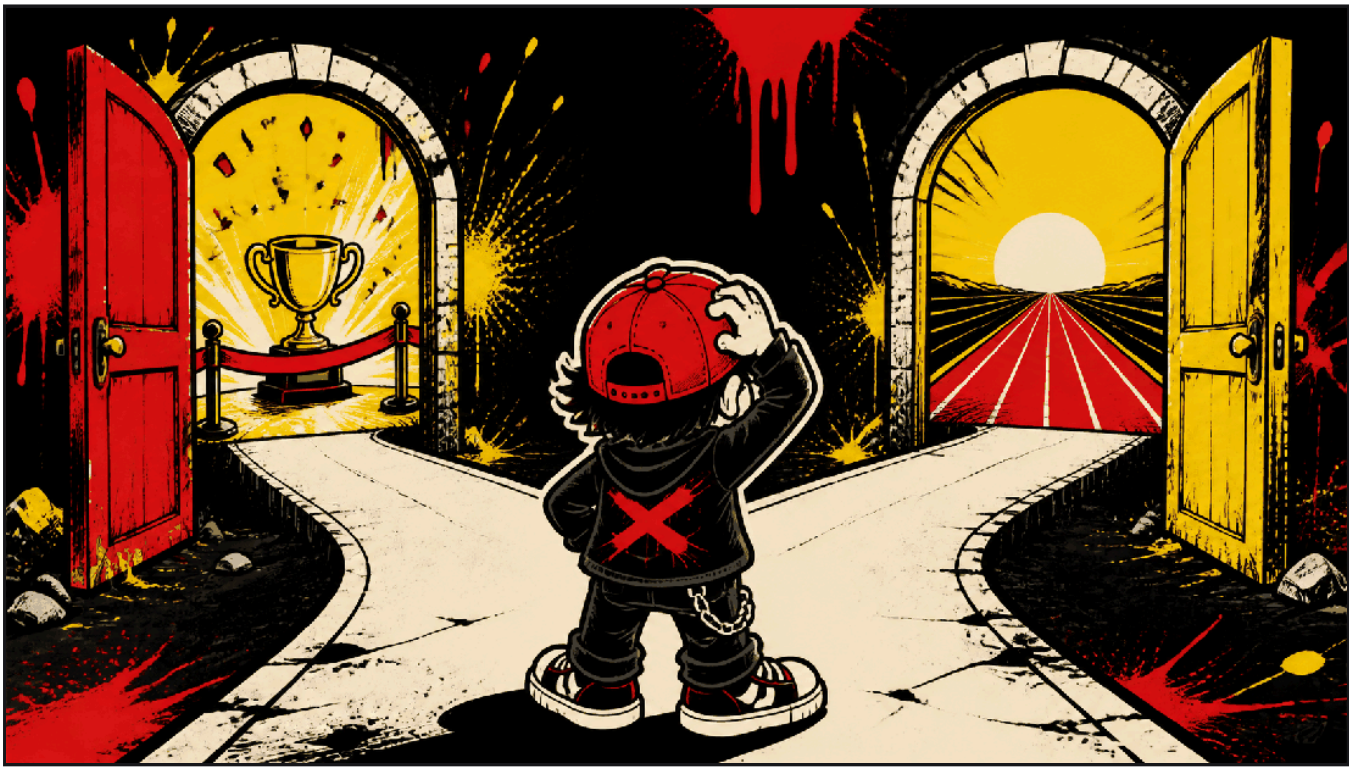
Общий язык. Чтобы «какой у нас scope?» и «покажи RAID» значили одно и то же в разработке, тестировании и поддержке. Тогда статус можно спросить одинаково у кого угодно, и человеку не надо гадать, в каком виде отвечать.

Рабочий минимум. Девять штук, каждая заполняется за полчаса и отвечает на живой вопрос. Не документооборот. Не отчётность ради отчётности. Набор, который помогает довезти результат.

Теории тут мало, и вся она в первых главах — чтобы ты понимал слова, когда их произносят вокруг. Дальше практика.

---

1. Что такое проект 🏁



Проект и операционка

Проект — работа с началом и концом, которая даёт результат, которого раньше не было. Доехали — проект закрыт.

> **Проект:** разделить линии поддержки на первую и вторую до 1 декабря. **Операционка:** каждый день разбирать входящие обращения.

Разница не в словах. У проекта есть цель, срок, ограниченные люди и момент, когда можно сказать «сделали». У операционки этого нет — она просто идёт и будет идти всегда. Ведут их по-разному: операционку выстраивают как процесс и меряют потоком, проект ведут к результату и меряют доездом.

Главный косяк — вести проект как операционку. Делать понемногу каждый день, без границ и вех. Через полгода он всё ещё «в процессе», а зачем затевали, уже никто не помнит.

Два вида проектов

С фиксированным объёмом. Понятно, что должно получиться: регламент написан, очереди настроены, первая линия обучена, поток разделён. Объём конечный — можно назвать дату.

Поток. Актуализация базы знаний, разбор техдолга, чистка справочников. Работа однородная, её просто много, и меряется она штуками. Выглядит как операционка, но конец есть — состояние, в котором мы хотим оказаться.

Разница рабочая, не теоретическая. У первых мы **обещаем дату**. У вторых — **фиксируем темп и каждую неделю пересчитываем прогноз**. Пытаться назвать дату для потока — это соврать себе и всем остальным. Подробности в главе про сроки.

Как выделить проект из своей работы

Возьми свою работу за квартал и разложи на две колонки.

В «проекты» уходит то, где на все четыре вопроса ответ «да»:

1. Есть результат, которого сейчас нет, а потом будет?
2. Понятно, когда это закончится?
3. Участвует больше одного человека?
4. Если не сделать — кому-то станет хуже, и ты можешь сказать, кому?

Хоть один «нет» — это либо операционка, либо просто задача. Задачу не надо обкладывать уставом и RAID. Взял и сделал.

И правило, которое экономит нервы: **если в проекте участвуют люди не из твоей команды — устав нужен точно.** Внутри своей команды можно договориться на словах. Через границу команды на словах не работает никогда.

---

2. Треугольник ограничений 🥊

У проекта всегда есть пять связанных ручек:

- **содержание** — что делаем;
- **срок** — когда;
- **ресурсы** — кем и за сколько;
- **качество** — насколько хорошо;
- **риски** — что может сломать план.

Они связаны. Дёрнул одну — поехали остальные.

Когда тебе говорят «сделайте в два раза быстрее» — это не задача, а начало разговора. Честный ответ такой:

> Ок. Тогда либо режем объём — вот что предлагаю выкинуть. Либо добавляем человека — нужен ещё один разработчик с 1 октября. Либо снижаем качество — выкатываем без нагрузочного и принимаем риск. Либо принимаем сдвиг даты. Что выбираем?

Это не торг и не сопротивление. Это единственный честный ответ — и самый управленческий навык из всего, что тут написано.

Лид, который отвечает на давление списком вариантов, ведёт проект. Лид, который отвечает «постараемся», просто переносит проблему на месяц вперёд. Через месяц она вернётся, только злее.

---

3. Пять стадий ⚡

Любой проект — от поездки на конференцию до релиза — проходит одно и то же.

1. Инициация — стоит ли вообще вписываться. Что за проблема, какая цель, кто заказчик, какой примерно масштаб, что тебе разрешено решать самому. На выходе — устав.

2. Планирование — как доедем. Что входит и что не входит, из каких кусков состоит, какие вехи, кто кого ждёт, сколько займёт, что может сломаться. На выходе — scope, roadmap, RAID.

3. Исполнение — команда делает работу. Твоя работа тут не «делать». Синхронизировать людей, убирать блокеры, добывать решения, следить за зависимостями, разговаривать со стейкхолдерами.

4. Контроль — идёт параллельно. Каждую неделю сверяешь: успеваем, не расползся ли объём, что с рисками, не потерял ли проект смысл. На выходе — недельный статус.

5. Завершение — это не «разработчики закончили». Принять результат, передать в эксплуатацию, закрыть договорённости, разобрать уроки, отпустить людей.

Пятую пропускают чаще всего. Проект сам не заканчивается: он либо закрывается явно, либо тихо превращается в операционку, которую никто не принимал и за которую теперь непонятно кто отвечает.

---

4. PMBOK за три минуты 🧠

PMBOK — свод знаний по управлению проектами, сейчас актуальна восьмая редакция. Это не методология и не инструкция «делай раз, делай два». Это сборник здравого смысла, практик и — что нам важнее всего — **общих слов**.

Наизусть его знать не надо. Надо понимать термины, потому что их используют все: в статьях, на конференциях, в разговоре с подрядчиком, на собеседовании.

Устроен он в три уровня.

Принципы — как себя вести: создавать ценность, брать ответственность, работать со стейкхолдерами, растить команду, управлять рисками, подстраивать подход под ситуацию, не терять качество.

Области управления — из чего состоит работа руководителя проекта. Их десять, таблица ниже.

Инструменты — устав, WBS, roadmap, реестр рисков, RACI, бэклог, Kanban, Гантт, отчёты, метрики, ретро. Инструмент берётся, когда отвечает на живой вопрос. Не потому, что он есть в списке.

Десять областей и что мы из них берём

Область	О чём	Что берём мы
Integration	собрать проект целиком, управлять изменениями	устав, журнал решений, закрытие
Scope	что делаем и чего не делаем	страница scope, декомпозиция, DoD
Schedule	сроки, вехи, зависимости	roadmap с вехами
Cost	деньги и оценки	у нас чаще человеко-недели, а не рубли
Quality	критерии качества и приёмка	Definition of Done команды
Resources	кто что делает	пять строк «роль — имя» в уставе
Communications	кому, что и как часто говорим	недельный статус, журнал решений
Risk	что может сломаться	буква R в RAID
Procurement	подрядчики и договоры	редко: конференции, дизайн, аутсорс
Stakeholders	кто влияет и чего хочет	4–6 строк в уставе

Что унести из этой главы: **PMBOK не требует производить документы. Он требует ответить на вопросы.** Ответил тремя строками в таблице — этого достаточно. «План управления качеством» на двенадцать страниц не нужен никому, включая PMBOK.

Плохой PMBOK — куча документов, которые никто не читает. Нормальный PMBOK — в любой момент понятно: что делаем, зачем, где мы сейчас, что мешает, кто принимает решение.

---

5. Наш набор 🧰

Девять штук. Больше в небольшой команде не нужно, меньше — начинает болеть.

Что	Когда заводится	Сколько времени
Устав	до старта	30–40 мин один раз
Scope	до старта работ	30 мин один раз
Roadmap с вехами	сразу после scope	20–30 мин, правится на статусах
Бэклог	с началом работ	живёт в трекере
RAID	до старта работ	15 мин + 5 мин в неделю
Журнал решений	с первого решения	2 минуты на запись
Weekly Status	каждую неделю	10 мин в неделю
DoD	один раз на команду, до первого проекта	15 мин, дальше не трогаем
Закрытие и ретро	в конце	час

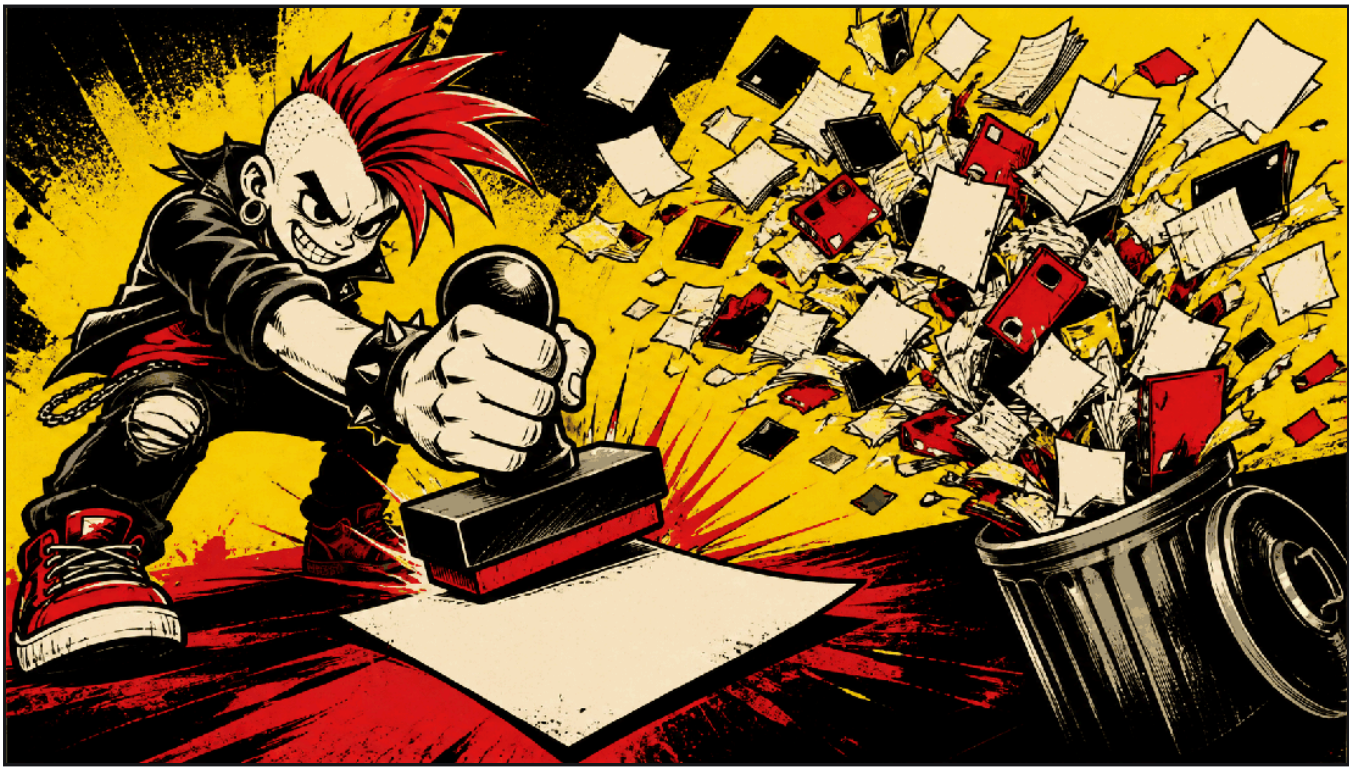
Цепочка целиком:

> **Устав → Scope → Roadmap → Бэклог → RAID → Журнал решений → Weekly Status → DoD → Закрытие/Ретро**

Итого на ведение проекта уходит примерно два часа на старте и минут двадцать в неделю. Это дешевле, чем один разбор полётов «почему мы это не сделали».

---

6. Устав — «зачем и для кого» 📜



Одна страница, которую ты пишешь до того, как кто-то начал работать. Отвечает на вопросы, которые потом всё равно всплывут, но уже посреди проекта и с нервами.

Что внутри:

- **Зачем.** Какую проблему решаем. Не «сделать фичу», а «клиники теряют записи и звонят нам в поддержку».
- **Результат.** Что появится, когда закончим.
- **Успех.** По чему поймём, что победили. Желательно проверяемое значение.
- **Заказчик.** Кто ставит задачу и кто принимает результат. Это может быть один человек, а может быть два разных — тогда пиши обоих.
- **Не делаем.** Две-три строки. Самая полезная часть устава, см. следующую главу.
- **Стейкхолдеры.** 4–6 строк: кто — насколько влияет — как и когда с ним общаемся.
- **Мои полномочия.** Что решаю сам, что согласую, что эскалирую.
- **Главная веха.**

Заводить отдельный «реестр стейкхолдеров» для команды из четырёх человек — перебор. Несколько строк в уставе закрывают ту же задачу.

Про полномочия отдельно. Это тот пункт, который лиды пропускают, а потом две недели ждут решения, которое могли принять сами. Или наоборот — принимают решение, которое было не их, и получают по шапке. Проговори это с заказчиком на старте: что ты двигаешь сам, что меняешь только с продактом, что уходит выше. Три строки, один разговор, ноль сюрпризов.

💀 **Косяк:** устав пишут после старта работ, «чтобы было». Тогда он бесполезен — все вопросы уже решены как попало. Устав ценен ровно тем, что заставляет ответить на них до начала.

---

7. Scope — что делаем и, главное, чего не делаем 🪓

Scope — граница проекта. Одна страница, два списка.

Делаем: 5–9 пунктов, каждый — законченный кусок, который можно показать. **Не делаем:** то, что логично напрашивается, но в этот раз не входит.

Второй список важнее первого. Все проекты умирают одинаково: не от того, что запланированное оказалось сложным, а от того, что по дороге прилипло ещё немного, потом ещё немного, и вот уже полгода прошло. Называется scope creep, лечится одной страницей на старте и одной фразой потом: «этого нет в scope, давай решим отдельно и запишем в журнал решений».

Декомпозиция

Она же WBS, если по-умному. Разбить результат на 5–9 кусков, каждый — ощутимая штука, которую можно предъявить.

Разделение линий поддержки разбирается так:

1. регламент маршрутизации;
2. настройка очередей в CarrotQuest;
3. обучение первой линии;
4. пилот на части потока;
5. полный переход.

Никакой нумерации 1.2.3.1 и уровней вложенности. Для команды из четырёх человек это плоский список или дерево на два уровня, не глубже.

Декомпозиция — это про **содержание**, не про сроки. Календарь из неё вырастает, но это следующий шаг и другой разговор.

Проверка, что разбил правильно: по каждому куску можно сказать, кто его делает и как поймём, что он готов. Не можешь — кусок слишком крупный или слишком размытый.

---

8. DoD — когда считаем, что готово 🏴

Definition of Done — короткий список того, что должно быть верно, чтобы работа считалась законченной. Заводится **один раз на команду**, а не на каждый проект.

Например, у разработки:

> код в мастере, тесты зелёные, ревью пройдено, документация обновлена, задача закрыта в трекере.

У поддержки или контента он будет свой — и это нормально.

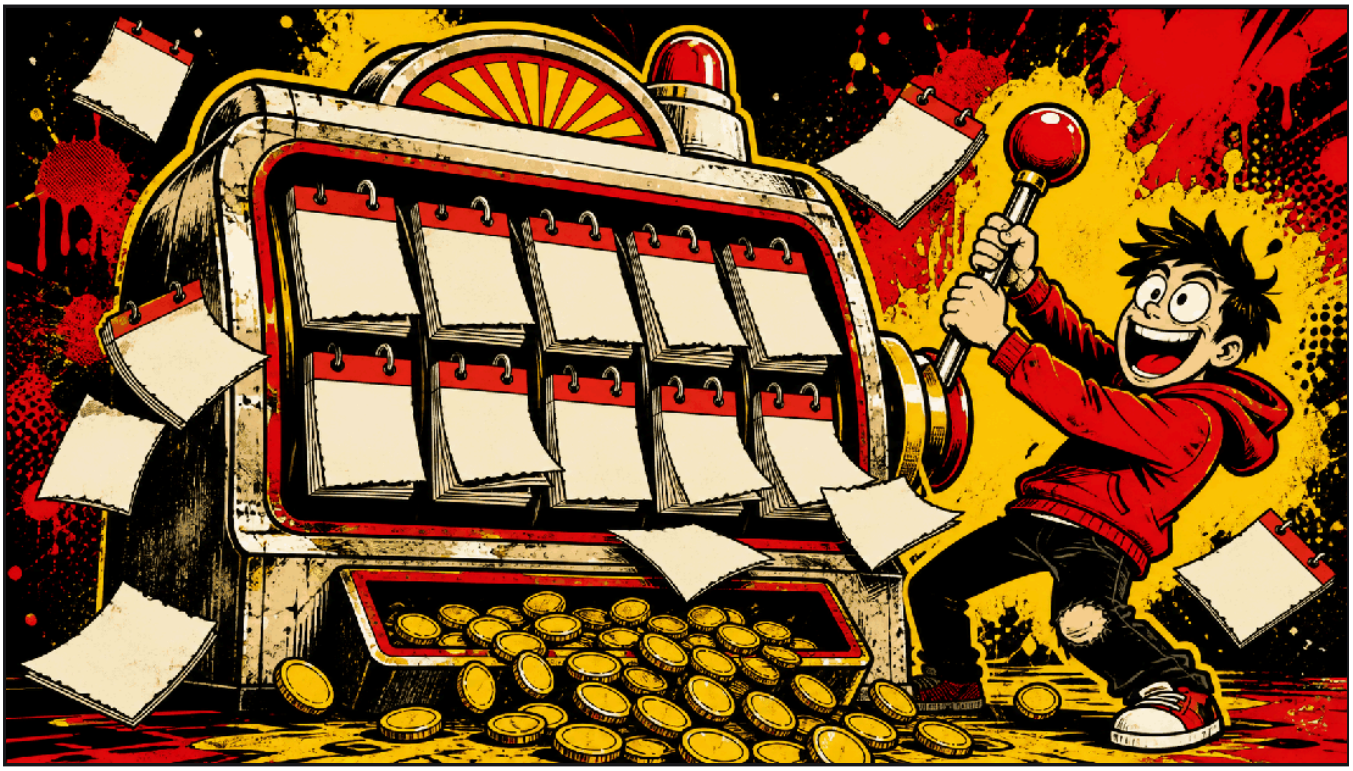
Отдельно от DoD живут **критерии приёмки** — это уже про конкретный результат: как заказчик поймёт, что получил то, что просил. Их пишешь в scope, по каждому крупному куску.

Разница простая: DoD — про то, как **мы** работаем. Критерии приёмки — про то, что **заказчик** получит. Первое одинаковое во всех проектах, второе каждый раз новое.

💀 **Косяк:** большой «План управления качеством». Не нужен. Нужен один список из пяти строк, который вся команда помнит наизусть.

---

9. Сроки 🎰



Самая больная тема, поэтому подробно. Шесть правил закрывают почти все реальные факапы.

1. Оценивает тот, кто делает

Ты не назначаешь срок сверху. Ты собираешь оценки и складываешь из них картинку. Это ещё и защита: срок, который человек назвал сам, он защищает. Срок, который ему спустили, он не защищает — он его переживает.

2. Человеко-дни — это не календарь

Главная ошибка вообще всех. Посчитали объём работы и назвали его датой.

Кусок на 10 человеко-дней, когда человек занят им один день в неделю, — это **десять недель**, а не две. У лидов операционка съедает половину времени, а то и больше, и про это стабильно забывают.

Считай в два шага: сколько работы → сколько её реально влезает в неделю.

3. Три точки вместо одной

Спрашивай не «сколько займёт», а «сколько если гладко / реально / если всё пойдёт не так».

Дальше можно посчитать: **(оптимистично + 4 × реально + пессимистично) / 6**. Это PERT из PMBOK, считается в уме.

Но ценность не в формуле. Ценность в третьем вопросе: «а если не так?» вытаскивает риски прямо на оценке, до того как они стали проблемами. Половина твоего RAID родится именно здесь.

4. Срок проекта — самая длинная цепочка, а не сумма

Если Flutter ждёт API, две недели работы Flutter не складываются с тремя неделями API — они идут следом. А то, что делается параллельно, вообще не складывается.

Критический путь по-пацански: нарисуй, кто кого ждёт, найди самую длинную цепочку. Вот это и есть срок проекта. Всё остальное имеет запас и не так страшно.

5. Буфер один и в конце

Не по 20% на каждую задачу — такие буферы съедаются молча, работа всегда занимает всё отведённое время. Один общий запас в конце проекта. Тогда его расход виден, и видно, когда он кончился.

6. Веха, а не дата

«Пилот на одной клинике» лучше, чем «готово 15 октября». По вехе видно, доехали или нет. По дате видно только то, что она прошла.

Roadmap — это 4–6 вех с датами и стрелками зависимостей. Не Гантт на сорок строк.

Длинные проекты, где даты не считаются

Актуализация базы знаний. Разбор техдолга. Чистка справочников.

Тут не надо мучиться и выдумывать дату. Режим другой:

- считаем **объём** — сколько всего штук;
- фиксируем **темп** — сколько делаем в неделю;
- считаем **прогноз** — осталось / темп;
- каждую неделю прогноз пересчитывается и показывается в статусе.

Дату не обещаем, прогноз показываем. Если темп не устраивает — это разговор про людей или про объём, то есть опять треугольник ограничений.

Вехи у таких проектов тоже есть, просто они не про даты: «половина базы актуальна», «все статьи по кассе переписаны».

---

10. Бэклог 🗃

Тут коротко, потому что он у всех уже есть. Задачи живут в Битриксе, а не в отдельном файле. Единственное, что от тебя нужно, — чтобы задачи бились с кусками из scope и чтобы по трекеру было видно, где проект стоит.

Если проект не разработческий (конференция, учебная база), бэклог всё равно нужен — просто это будет чек-лист, а не доска. Главное, чтобы он был в одном месте, а не в голове и в переписке.

---

11. RAID — одна таблица вместо четырёх документов 🐉

Самая полезная штука во всём наборе. Четыре буквы:

- **R — Risks.** Может случиться. «App Store задержит проверку».
- **A — Assumptions.** Мы на это рассчитываем, но не проверяли. «API партнёра будет готов 1 октября».
- **I — Issues.** Уже случилось. «QA заболел».
- **D — Dependencies.** Кто кого ждёт. «Flutter ждёт API».

Тип	Что	Влияние	Кто ведёт	Что делаем
Risk	App Store задержит проверку	high	лид	отправляем на неделю раньше
Assumption	API готов 01.10	high	бэкенд	проверить 15.09
Issue	QA заболел	high	лид	ищем замену
Dependency	Flutter ждёт API	high	разработка	работаем на моках

Риск и проблема — разные вещи. «Backend может задержаться» — риск. «Backend задержался на неделю» — проблема. Когда риск сбылся, он переезжает из R в I, и это нормальный ход событий, а не провал.

Про допущения. Самая недооценённая буква. Допущение — это то, на чём стоит весь план, но чего никто не проверял. Каждое допущение обязано иметь дату проверки. Непроверенное допущение — это риск, который ты ещё не заметил.

Не надо вести семьдесят строк. Пять-десять живых. Обновляется раз в неделю на пять минут, перед статусом.

---

12. Журнал решений 🕵️

Четыре колонки: **дата — что решили — кто решил — почему**. Плюс пятая, если решение двигает план: что от этого изменилось.

> 15.08 — отказались от SMS-авторизации. Решение: продакт. Причина: стоимость. Используем Telegram OTP.

Всё. Две минуты на запись.

Зачем: через два месяца — а на длинных проектах через полгода — никто не помнит, почему сделано именно так. И начинается «а какого хрена мы так сделали?», раскопки переписки и попытки переиграть то, что уже обсудили и закрыли.

Отдельная ценность — когда меняется scope. Любое «а давайте ещё вот это» либо отклоняется, либо принимается — и тогда записывается вместе с тем, что взамен выкинули или сдвинули. Тогда через месяц никто не удивляется, куда уехала дата.

Если из всего набора ты возьмёшь только одно — бери журнал решений. Дешевле всего, окупается лучше всего.

13. Weekly Status 📡

Десять минут в неделю. Формат — три строки:

> **База:** что обещали (релиз 1 декабря, scope такой-то). **Факт:** где мы сейчас (бэкенд отстаёт на неделю, пилот запущен). **Прогноз:** когда доедем при текущем раскладе (8 декабря).

Плюс к этому:

- что изменилось за неделю;
- что горит из RAID;
- где нужно решение и от кого.

Последний пункт — главный. Статус нужен не чтобы отчитаться, а чтобы получить то, что тебе нужно для движения. Если тебе ничего не нужно — так и напиши, это тоже информация.

Кому и как часто. Договорись на старте, кому и как часто: команде — дейли или чат, продакту и руководителю — недельный статус, критический риск — сразу, не дожидаясь недели. Пиши это в устав, в блок стейкхолдеров.

💀 **Косяк:** статус в формате «сделали то, сделали сё». Это отчёт о деятельности, а не статус проекта. Читающему нужно понять, доедем ли и не надо ли вмешаться, — а не сколько задач вы закрыли.

---

14. Полномочия и эскалация 🚨

Ты не должен решать всё сам. И не должен тащить наверх всё подряд.

Порог эскалации определяется на старте, в уставе:

> Бэкенд отстаёт на два дня → решаем внутри команды. Критический путь отстаёт на три недели, а дата фиксирована → идём к продакту и выше.

Хороший лид знает свой порог заранее и не изобретает его в момент, когда уже горит.

Правило: **эскалируй раньше, чем стало поздно, и приходи с вариантами.** Не «у нас проблема», а «у нас проблема, вижу три выхода, рекомендую второй, нужно твоё решение до пятницы». Это разница между «принёс проблему» и «принёс решение».

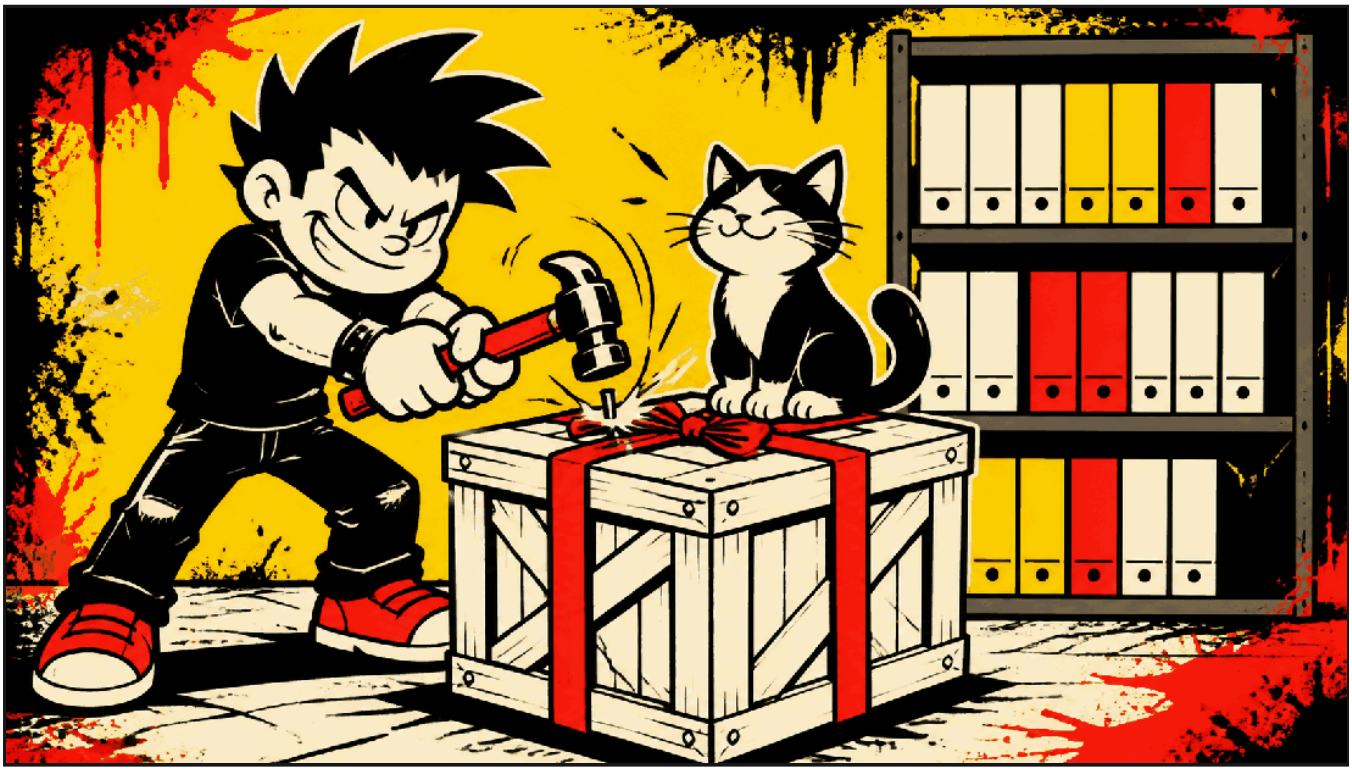
Отдельно проговори, кто вообще что имеет право решать:

- лид двигает задачи и сроки внутри вех;
- продакт меняет приоритеты и scope релиза;
- перенос фиксированной даты утверждает руководитель.

Три строки в уставе. Экономят недели.

---

15. Закрытие и ретро ⚰️



Проект надо закрыть явно. Не «разработчики закончили», а:

- **результат принят** — заказчик посмотрел и сказал «да, это оно»;
- **передан в операционку** — понятно, кто теперь это поддерживает и отвечает;
- **хвосты зафиксированы** — то, что не доделали, живёт в бэклоге, а не в голове;
- **договорённости закрыты** — подрядчики, доступы, счета;
- **команда отпущена** — люди знают, что проект кончился.

И ретро на час: что сработало, что нет, что в следующий раз делаем иначе. Записать и положить рядом с проектом. Через год это будет единственное, что от проекта останется полезного.

💀 **Косяк:** проект не закрывают, он просто затухает. Тогда никто не знает, кто теперь отвечает за результат, хвосты теряются, а уроки не извлекаются — и следующий проект наступает на те же грабли.

---

16. Частые косяки 💀

Собрано из того, что реально бывает.

🤷 **Проект без «зачем».** Начали делать, потому что попросили. Через месяц выясняется, что проблема была другая. Лечится первой строкой устава.

🐙 **Нет списка «не делаем».** Scope расползается по чуть-чуть, никто не замечает момента, когда стало поздно.

🎰 **Оценка в человеко-днях, обещание в календаре.** См. правило 2 в главе про сроки.

📜 **Устав после старта.** Написан для галочки, все вопросы уже решены как попало.

🧟 **RAID на семьдесят строк.** Ведётся ради ведения, никто не читает, обновляется раз в квартал. Пять живых строк лучше семидесяти мёртвых.

📡 **Статус как отчёт о деятельности.** «Сделали то, сделали сё» вместо «доедем или нет».

🕵️ **Решения нигде не записаны.** Через два месяца переигрываем то, что уже обсудили.

🚨 **Эскалация в последний момент.** Пришёл, когда спасать поздно, и без вариантов.

⚰️ **Проект не закрыт.** Затух, превратился в операционку, ответственного нет.

---

17. Шпаргалка 🎯

🚀 **На старте — два часа:**

1. Устав: зачем, результат, успех, заказчик, не делаем, стейкхолдеры, полномочия, веха.
2. Scope: делаем / не делаем, разбить на 5–9 кусков.
3. Roadmap: 4–6 вех, зависимости, оценки в три точки, буфер в конце.
4. RAID: 5–10 строк, у каждого допущения — дата проверки.

🔁 **Каждую неделю — двадцать минут:**

5. Обновить RAID.
6. Написать статус: база → факт → прогноз, что горит, где нужно решение.
7. Записать решения, если были.

🏁 **В конце — час:**

8. Принять результат, передать в операционку, зафиксировать хвосты.
9. Ретро: что сработало, что нет, что иначе.

👊 **Если запомнить только три вещи:**

- на «сделайте быстрее» отвечай списком вариантов, а не «постараемся»;
- записывай решения;
- закрывай проект явно.