Владимир Завертайлов

основатель и руководитель
scrum-студии «Сибирикс»

настольная книга *project*-менеджера

КАРТА КОМПЕТЕНЦИЙ РУКОВОДИТЕЛЯ ПРОЕКТОВ

13 ТАБЛИЦ С ОБУЧАЮЩИМИ МАТЕРИАЛАМИ

6 АВТОРСКИХ ЧЕК-ЛИСТОВ

что нужно знать, чтобы управлять **IT**, **digital** и другими проектами с учетом российских реалий

pro

Top expert. Практичные книги для работы над собой

Владимир Завертайлов

Настольная книга project-менеджера. Что нужно знать, чтобы управлять IT, digital и другими проектами с учетом российских реалий

«Эксмо»

2022

УДК 005:004
ББК 65.290-2

Завертайлов В.

Настольная книга project-менеджера. Что нужно знать, чтобы управлять IT, digital и другими проектами с учетом российских реалий / В. Завертайлов — «Эксмо», 2022 — (Top expert. Практичные книги для работы над собой)

ISBN 978-5-04-173940-9

Максимально полный путеводитель по профессии российского project-менеджера. В нем Владимир Завертайлов, основатель и руководитель scrum-студии «Сибирикс», которая входит в Топ-10 лучших веб-студий страны, рассказывает, как управлять собственным digital-производством и грамотно руководить проектом, заказывая digital-услуги на стороне. А еще – как расти в этой профессии, опираясь на знания о продуктовых техниках и понимая потребности бизнеса. Из книги вы узнаете: - чем kanban отличается от scrum и при чем тут agile; - как управлять неуправляемым – дизайнерами; - что лучше: собственная команда или аутсорс и как быть с техподдержкой; - откуда берутся оценки времени и как понять, что исполнитель их завышает; - что такое декомпозиция, зачем она нужна на проекте; - как планировать экономику проекта, рассчитывать операционные затраты и анализировать P&L продукта. В карточку подгружен издательский PDF макет.

УДК 005:004
ББК 65.290-2

ISBN 978-5-04-173940-9

Содержание

Владимир Завертайлов
Настольная книга project-менеджера. Что нужно знать, чтобы управлять IT, digital и другими проектами с учетом российских реалий

* * *

Друзья! Всем, кто приобрел эту книгу, мы дарим полгода бесплатной работы в лучшем персональном планировщике задач SingularityApp. Вы экономите 1000 рублей. Чтобы узнать, как получить подарок – переходите по ссылке https://singularity-app.com/pmbook/ или QR-коду

Введение

Что нужно, чтобы стать руководителем digital-проектов или руководителем продукта в IT? Уж точно не классическое вузовское образование (хотя без него будет трудно). На пути к должности руководителя вы потратите огромное количество времени на практику, не раз набьете шишки и столкнетесь с факапами, прольете немало пота, крови и слез, вложите в это много труда. Но самое главное – приготовьтесь каждый день ставить свою шкуру на кон. Потому что руководитель проектов и продуктов отвечает за свое детище, как никто в команде. И делает все то, что нельзя делегировать.

Вас к такому не готовили

Почти 20 лет я руковожу студией разработки и не один год провожу стажировки для студентов. Статистика печальна: востребованных специалистов для управления проектами и продуктами в вузах, увы, не готовят. Поэтому мне приходится отсматривать десятки претендентов на стажировку и отсеивать тех, кто хочет халявы в роли ничего не делающего надзирателя или легких денег (проджекты же много зарабатывают, правда?). Но те, кого я беру на стажировку – счастливчики. Потому что они моментально погружаются в рабочую атмосферу. Сразу вникают в процессы, самостоятельно делают аналитику, продумывают структуру проектов (пусть и не существующих в реальности), ведут их от начала и до конца, наступают на грабли, совершают ошибки. Сразу – действительно учатся. И даже если в дальнейшем наши пути расходятся, ребята уходят счастливыми и окрыленными.

Но что делать, если у вас за плечами нет такой стажировки, а вы хотите быть руководителем продукта или проекта? Или, возможно, уже им стали, но что-то постоянно не клеится: сроки срываются, команда косячит, и вам опять надо это разгребать в стопятьсотый раз? Знакомо? Знакомо. Нет, можно, конечно, уволиться и начать все сначала. Но собственные системные ошибки догонят где угодно – потому что от себя не убежишь.

На должность руководителей проектов я собеседовал множество раз, и это были самые разные люди. Одни работали в каких-нибудь госзакупках, другие – в прошлом были инженерами, но доросли до руководящих позиций, третьи – менеджерили команду в банке или вообще были маркетологами. Меня мало интересует опыт из резюме – он вряд ли расскажет о том, способен ли человек решать проблемы здесь и сейчас или нет. И уж точно не покажет, насколько крутой из него переговорщик и справится ли он с интеграцией на проекте. Меня больше интересуют навыки – те самые Soft Skills, о которых мы будем говорить в этой книге. И которых часто не хватает многим, и не только в IT-среде.

Кто виноват и что делать

Хорошая новость: вы не виноваты в том, что на старте в профессию не обладали огромным багажом знаний и что даже многолетний опыт не спас вас от ошибок. Так уж повелось, что в классических учебных заведениях необходимую информацию почти не дают, а на предыдущих местах работы редко удается прокачать все нужные скиллы.

Но есть и плохая новость: вы в ответе за то, что все так, как есть. Сейчас вы должны решить, что делать дальше: учиться или продолжать слепо ходить по старым дорожкам в надежде получить другой результат.

Если выбираете второй путь – ставьте книгу обратно на полку, диалога у нас не получится. Если все-таки первый – то придется потрудиться. Но оно того стоит.

В digital все крутится вокруг всего двух компетенций: технологии и коммуникации. Hard Skills & Soft Skills. Если проседают коммуникации, руководитель сможет завалить проект, сделанный сильной технической командой. И наоборот: позитивный и энергичный менеджер, который умеет договариваться – вытянет слабый проект и сделает так, что заказчик попросит еще. Технологии же нужны, чтобы общаться с командой и заказчиками на уровне эксперта.

Что вам точно понадобится?

▶ Честность. На уровне «пацан – сказал, пацан – сделал». Без этого с человеком, по-моему, вообще нельзя иметь дел.

▶ Интеллект. В первую очередь – системное мышление.

▶ Зрелость. Готовность брать на себя ответственность.

▶ Позитив и энтузиазм. Если менеджер по натуре депрессивен, и у него на лице написано: «Господи, когда же это все дерьмо закончится» – не стоит ожидать, что команда радостно подхватит знамя и с криками «Эге-гей! Дерьмо! Кончайся!» побежит делать проект.

▶ Внимание к мелочам. Большие проблемы вырастают из мелочей, которые проглядели на старте работ. Или заметили, но решили, что они никак не повлияют на будущее. Мелочи ничего не прощают большому.

▶ Готовность незамедлительно действовать.

▶ Требовательность. Умение пойти на конфликт, если что-то сделано плохо, отстаивать свои интересы и интересы дела.

Вообще, путь менеджера – это трудный путь. В нем много сложностей и много работы над собой, характером, привычками. А это больно. Приходится нести ответственность не только за себя, но и за других, и за весь проект. Отвечать за чужие косяки. Двойные неприятности.

Зато это очень интересный путь, интересная судьба. Практически бесконечные возможности как для горизонтального роста, так и для вертикального, если он вдруг вам понадобится. Хорошо платят. Вы приобретаете чертовски полезные навыки для повседневной жизни. И это определенная степень свободы, умение владеть собой. И еще – вы будете востребованы, и скучно точно не будет. Но это – если оно вам действительно надо. А раз уж вы держите в руках эту книгу, хочется надеяться, что это так.

Что вас ждет в этой книге

В 2017-м году мы в студии создали обучающий курс на Skillbox «Руководитель digital-проектов», который был одним из первых на рынке по этой теме. За 4 года существования его прошли сотни студентов. И каждый из них хотел бы иметь под рукой инструмент, на который всегда можно положиться в сложной и неоднозначной ситуации.

Цель – дать этот инструмент тем, кто только собирается в профессию руководителя проекта или руководителя продукта, и тем, кто уже в ней работает и развивается, но испытывает сложности. Можно сказать, это настольное практическое руководство для проджектов и продактов, которым приходится либо заказывать digital-услуги на стороне, либо управлять своим собственным digital-производством.

Почему книга сразу и для проджектов, и для продактов? Потому что в небольших проектах эти роли часто выполняет один человек. Project Manager (диджитал-продюсер, PM, менеджер, руководитель проектов – зовите, как хотите) – специалист, который своей шкурой отвечает за проект и несет ответственность за то, чтобы в процессе работы ни одно животное не пострадало. Графики, сроки, команда, бюджеты, релизы, качество – вот что в фокусе. Product Manager – это, скорее, про маркетинг, стратегию, приоритеты, трафик, сегменты, конкурентов.

Но работы полно и там, и там. Более того, хороший руководитель проекта должен владеть продуктовыми техниками, чтобы лучше понимать потребности бизнеса. А хороший руково-

дитель продукта должен вырасти из руководителя проектов, чтобы понимать возможности и ограничения своей команды и иметь адекватную картину мира. Именно поэтому в этой книге есть техники для обеих специальностей.

В ней также найдутся ответы на те непростые вопросы, с которыми сталкиваются начинающие и уже опытные менеджеры проектов и продуктов ежедневно:

▶ как общаться с подчиненными, чтобы не быть в их глазах мямлей или деспотом;

▶ как грамотно делегировать и что придется в себе искоренять, чтобы это уметь;

▶ почему Scrum – все еще передовой фреймворк для разработки проектов и продуктов в digital и как его внедрить в свою команду;

▶ как и с помощью каких инструментов проводить аналитику перед стартом проекта, чтобы в процессе разработки не случалось неприятных сюрпризов;

▶ что такое декомпозиция, зачем она нужна на проекте и как грамотно составлять сметы, чтобы потом не было мучительно больно и стыдно перед заказчиком;

▶ как управлять творческими сотрудниками и креативным процессом без магии, шантажа и подкупа;

▶ как строить команду, чтобы людям хотелось приходить на работу и расти профессионально;

▶ как распределять собственное время менеджера эффективно и при этом не выгореть и не уехать в бессрочный отпуск;

▶ что лучше: собственная команда или аутсорс и как быть с техподдержкой;

▶ как не плодить бюрократию, но при этом четко оформить процесс разработки документально;

▶ как интегрировать сайты и продукты с внешними сервисами и системами без моря слез;

▶ что делать, если все пошло не по плану и случился факап на проекте;

▶ и как, черт побери, понять все то, что ежедневно говорит тебе команда на своем айтишном языке.

Полистайте содержание. Посмотрите главы. Загляните в задания после каждого блока. Если есть сомнения – поставьте, где взяли. Не поможет. Особенно, если не планируете постоянно практиковаться. В книге, конечно, есть забавные истории, примеры, кейсы, чек-листы и шаблоны, которые можно брать и сразу применять. Но это – инструмент, а им нужно пользоваться!

Часть 1
Картина мира digital-проекта. Тонкости делегирования в IT

Возьмем такую задачку: **довести сайт до ума.**

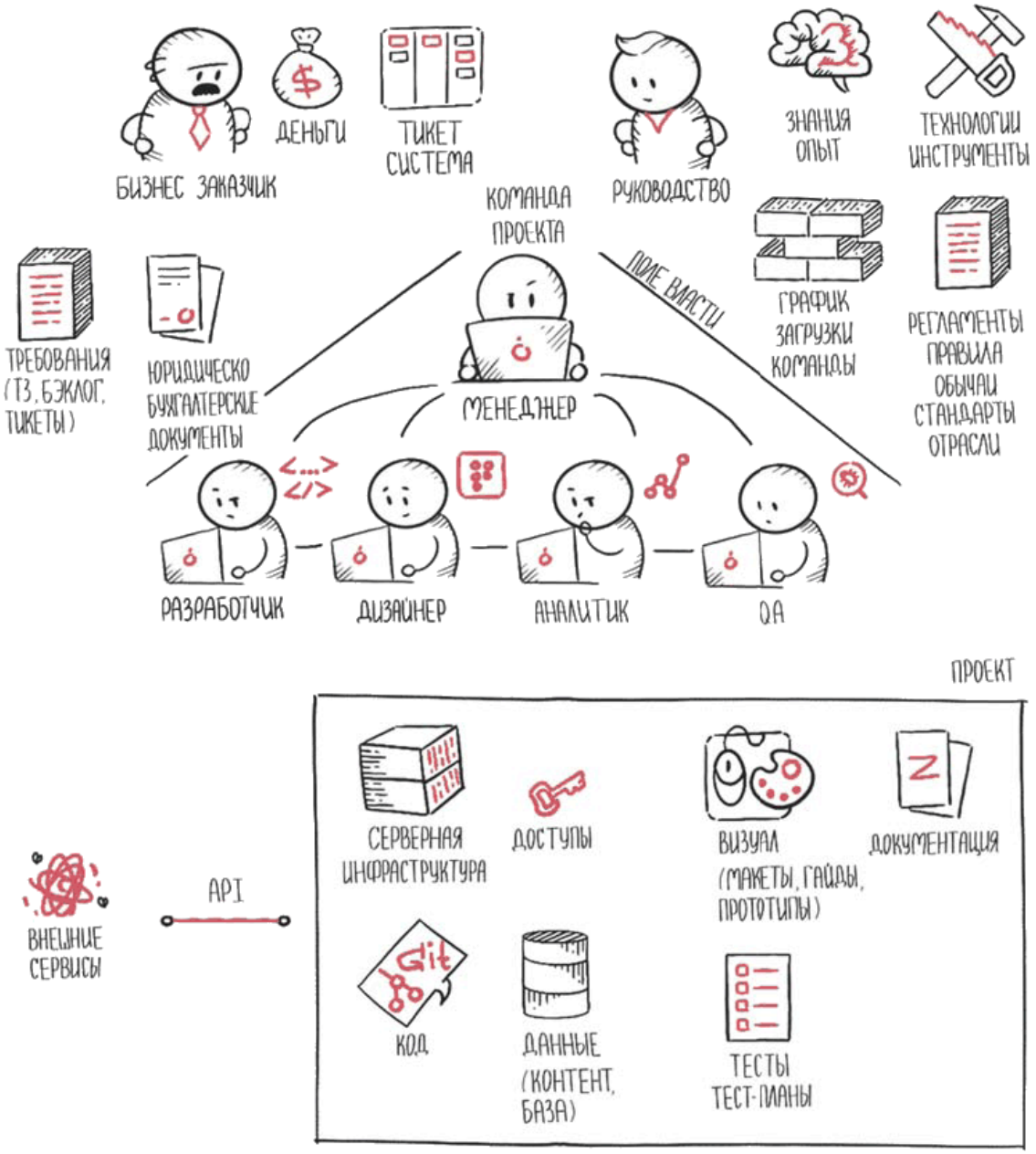
Нет, я не пошутил с формулировкой. Таких задач – доделать за предыдущими разработчиками – на фриланс-биржах пруд пруди. И несчастные сотрудники в них периодически вляпываются. Допустим, вы – менеджер, у вас микрокоманда: дизайнер, программист, аналитик, тестировщик. С которыми вы раньше не работали. И есть еще бизнес-заказчик, нешибко разбирающийся в digital-премудростях, но уже почему-то раздраженный. Он эту карусель оплачивает, если его устроит цена и качество. Понятно, что его в первую очередь интересуют деньги и сроки. Ваши действия?

Правильно: бежать!

А что нас смущает? Запредельная степень неопределенности? Тухлая перспектива? Неуверенность в команде? Непонятки с оплатой? Неадекватный заказчик? Поехали, потом заведешься? Гарантированный геморрой при негарантированном результате?

Давайте мысленно представим, как выглядит картина мира digital-проекта и сколько возни тут вырисовывается.

1.1. Картина мира digital-проекта



Итак, в нашей минимальной картине мира сразу получается куча объектов. И с каждым из них много возни и неопределенности. Объекты связаны, влияют друг на друга, за всеми приходится следить и что-то корректировать. Давайте разбираться, пока поверхностно, потом пойдем в дебри.

Заказчик. Бывают два типа заказчиков. Внутренний, когда разработка идет in-house. И внешний, когда проект разрабатывается на заказ.

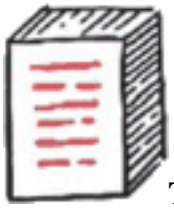
Заказчик может быть чертовски сложно устроен: не один человек, а группа (с противоречивыми интересами, требованиями к проекту и даже конфликтами). А еще бывают скрытые агенты влияния, например, у заказчика – жена, с которой он советуется, и ее мнение в итоге будет определять эстетическую сторону проекта. Но вы об этом не узнаете.

У заказчика определяющая роль. Он формирует концепцию проекта, приоритеты, финансирует весь карнавал. От него зависит все: что бы вы как менеджер ни делали, всегда остается шанс упереться в стенку на той стороне. Именно заказчик определяет степень успешности проекта. Подробнее о работе с заказчиком поговорим в главе 9.

Деньги. Энергия проекта, без них ничего не получится. Даже если это фановые, волонтерские проекты в свободное время или проекты за долю в будущих прибылях. Тогда деньги не присутствуют в явном виде – источником может стать сама команда. Классика провала: два друга пилили проект вместе по вечерам, а потом один женился и взял кредит. И все, приоритеты поменялись, разошлись дорожки…

Различают два ключевых формата работы. Time & Material – оплачивается время, затраченное командой, в таком случае задачи можно менять как угодно. И Fixed Price – цена и задачи оговариваются на берегу. У обоих вариантов есть как плюсы, так и минусы. Time & Material переносит риски на заказчика, но работу можно организовать быстрее и гибче. Fixed Price – заставляет разработчика зашивать риски и подстраховку в смету, лишает гибкости и снижает скорость выпуска функций из-за процедуры согласования бюджета, но яснее воспринимается клиентом. Существуют также гибридные модели.

Условия заказчика регламентируются тремя способами:



Требования. Постановки задач, технические задания (ТЗ), ***тикеты***[1], рисунки на салфетках, ***бэклоги***… Управление требованиями, выявление, уточнение, актуализация, расстановка приоритетов занимают массу времени. Рассмотрим подробно работу с требованиями и способы приоритизации в главе 4.

Юридические документы. Иными словами, бюрократия, более актуальная для заказной разработки. ***NDA***, договоры, приложения, акты и т. д.

[1] Расшифровку терминов, выделенных полужирным курсивом, можно найти в «пояснениях» (конец каждой главы).

Тикет-система. Хранит задачи для команды и статусы по ним. Может быть частью системы управления требованиями или существовать отдельно. Может быть только для внутренних нужд команды, а может пускать заказчика внутрь.

Команда проекта. В digital команда может сжиматься до одного человека, а-ля вебмастера, и включать в себя заказчика. Либо наоборот: разрастаться, выделяя отдельные функции в роли. Как строить и выращивать команды, мы разберем в главе 7. Командная работа подразумевает делегирование. Например, менеджер справится с задачами аналитика и тестировщика, но с ростом проекта целесообразно выделить это в отдельные роли. Или программист может администрировать серверы, если квалификация соответствует. Но как только серверного хозяйства становится много – приходится выносить это в отдельную роль – администратора. Для примера будет такая команда:

Менеджер. Ну тут все понятно, это – вы. Головой отвечаете за проект, работаете в области высоких энергий и мгновенной ответственности за базар. Про вас вся эта книга.

Разработчик (программист). Бородатый чувак, который пилит код. Пока без дебрей на чем, бэкенд-фронтенд. Универсальный боец.

Дизайнер. Отвечает за весь визуал и интерфейсы. Хотя пошла мода на специализации UI/UX-дизайнеров и т. д. Подробнее об организации дизайна поговорим в главе 6.

Аналитик. Конкретизирует требования. Подробнее об этом поговорим в главе 4, посвященной аналитике.

QA (quality assurance, тестировщик). Тестирует продукт.

Поле власти. Что-то типа электрического поля, к которому подключаются менеджерские инструменты типа «делегирования». Подробнее про власть разберем в главе 2.

Руководство. Ваш непосредственный руководитель и бизнес-заказчик могут быть разными людьми. С одной стороны, у руководства свои требования и интересы, с другой – у него есть ресурс, который недоступен вам, но полезен проекту. Власть, опыт, переговорные навыки, право закупать оборудование и ***софт***, менять процессы работы и т. д.

Кто для вас главнее: заказчик или руководитель, и что делать, если у них противоречивые требования? Например, заказчик попросил изменить процесс: нарисовать дизайн до аналитики. Можно ли? Да, если согласуете с руководителем. Приоритет отдавайте руководителю. При этом никто не против, чтобы клиент был счастлив.

Еще несколько компонентов digital команды:

Знания и опыт. Экспертиза команды. То, благодаря чему заказчик обратился к вам, а не к конкурентам. Знания и опыт индивидуальны у каждого члена команды.

Регламенты, правила, обычаи, стандарты отрасли. Документы или принципы, по которым организован рабочий процесс в команде. Формализуют знания и опыт команды. Такие регламенты собираются в корпоративные базы знаний и доступны всем участникам, но это не гарантирует 100 % осведомленность или применяемость.

Инструменты и технологии. Компьютеры и программное обеспечение, необходимые для разработки нового софта. Навыки владения такими инструментами принято называть Hard Skills.

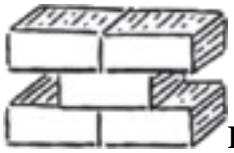


График загрузки команды. Чтобы спрогнозировать сроки готовности каждой функции, приходится планировать календарную загрузку каждого члена команды и учитывать зависимости задач. Если же вы работаете в мультипроектной среде – приходится учитывать загрузку команды на других проектах. Тут помогут сетевые графики, теория ограничений и метод освоенного объема из главы 5.

Проект. Это набор ***артефактов***, в основном – файлов:

▶ **Код,** как правило, в ***репозитории***.

▶ **Визуал** – макетов, прототипов, ***гайдлайна*** и т. д.

▶ **Серверная инфраструктура**, на которой ведется разработка, тестируется или работает сам проект.

▶ Разноуровневый **Доступ** и хранение паролей.

▶ Кроме кода и визуала в проекте обязательно есть **Данные**. Это либо база, либо контент, ради обработки которых проект и затевался.

▶ В развитых проектах код покрывают **Тестами** и пишут формальные ***Тест-планы*** выпуска продукта.

▶ Большинство проектов взаимодействуют с **Внешними сервисами**. Например, авторизуют пользователей через социальные сети или рассчитывают стоимость доставки товара. Для этого разработчики выполняют интеграцию через ***API***. Подробности в главе 10.

▶ Наконец, в зрелом проекте обязана быть **Документация**. Ее главная задача – отчуждение знаний о проекте из голов конкретных исполнителей. Документация снижает риски.

1.1.1. Прогулка по картине мира

Итак, мы героически вляпались в задачу «довести сайт до ума». Как бы я действовал (кое-где – чужими руками)?

> Жуть, этот план даже читать больно! А ведь вам завтра крутить окровавленную и облитую слезами ручку digital-разработки:)

▶ Поинтересовался, почему расстались с предыдущим разработчиком, на какой ноте, кто это был. Навел бы справки. По возможности переговорил с разработчиком, составил свое мнение.

▶ Получил от клиента предварительный список требований и задач. Рассказал всю процедуру и описал риски. Запланировал время на то, чтобы вникнуть в чужой код. Возможно, его придется полностью удалить, и все написать по новой. Первое время наша скорость будет ниже, чем у предыдущей команды, и у нас будет больше ошибок, потому что мы не знаем всех взаимосвязей. Также обсудил условия, при которых можно попробовать взяться за задачу.

▶ Грубо оценил стоимость проекта (вилочно, от-до), получил подтверждение бюджета у клиента.

▶ Согласовал работу по Time & Material. Работать с чужим проектом по Fixed Price – самоубийство. Выяснил, сколько примерно часов готов выкупать клиент в будущем и какие дальнейшие планы на сотрудничество. Разовый, короткий контракт, отношения на одну ночь меня бы не заинтересовали – в этом случае посоветовал бы закончить проект с предыдущей командой.

▶ Запросил доступы к коду.

▶ Провел ***код-ревью*** – процедуру анализа и оценки качества кода.

▶ Изучил текущую документацию. Если документации нет – это тоже показатель.

▶ Принял решение, можно ли работать с текущим кодом или нужно выбросить и все делать с нуля.

▶ Подписал контракт.

▶ Получил предоплату.

▶ Еще раз уточнил требования. Собрал их в бэклог. Отсортировал по приоритетам. Организовал работу ***спринтами***.

☐ Нарисовал прототипы. Сдал клиенту.

☐ Нарисовал дизайн, если задача это предполагает. Также сдал клиенту.

☐ Еще раз проговорил голосом результат с заказчиком, убедился, что мы все одинаково понимаем.

☐ Доформировал требования на уровне задач в тикет-системе с учетом изменений, которые появились в дизайне и прототипе. Обычно это мелочи, но иногда все разворачивается на 180°.

☐ Дал вычитать требования разработчикам.

☐ Проговорил задачи голосом с командой, ответил на вопросы. Получил оценки от команды, например, методом Planning Poker (подробнее в главе 3).

☐ При необходимости провел ***рисерч***. Это нужно для задач, с которыми команда никогда не сталкивалась.

☐ Запланировал разработку в календарном плане.

☐ Написал тест-кейсы или критерии приемки по каждой из задач.

☐ Запрограммировал. Следил за ходом работ, решал «затыки».

☐ По итогам разработки провел код-ревью нового кода.

☐ Протестировал, исправил ***баги***.

☐ Проверил производительность.

☐ Сдал заказчику проект на тестовом сервере, получил и обработал обратную связь. Мелкие недочеты исправил сразу, остальное перенес в будущие спринты.

☐ Провел ***ретроспективу*** с командой, посмотрел, что можно улучшить в проекте и процессах работы.

☐ Актуализировал документацию.

☐ Провел ***деплой*** (выкладку на боевые сервера).

☐ Обновил контент.

☐ Проверил работу на ***боевом сервере***.

☐ Подписал акты, получил постоплату.

▶ Выяснил, что еще хотел бы клиент, повторил бы цикл. Сам предложил улучшения проекта: по коду, функциям, дизайну, удобству и простоте использования.

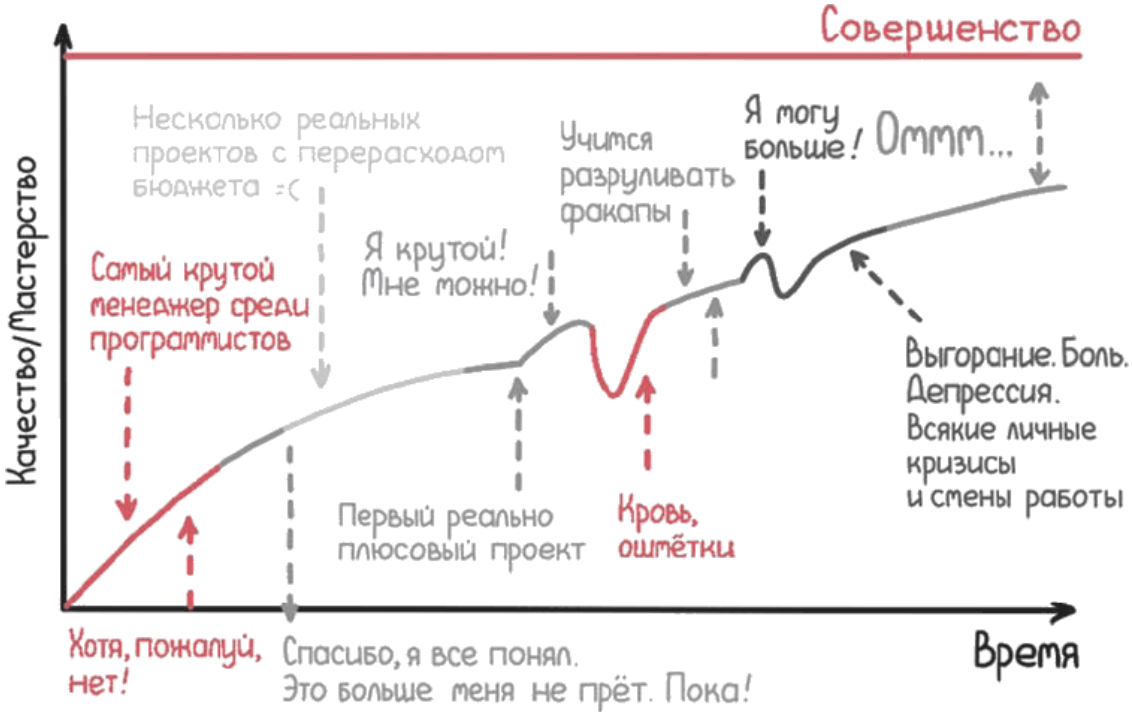
▶ Организовал работу по мелко-срочным отвлекающим тикетам (по более дорогой ставке), выполнение которых клиент не готов ждать.

В общем, у менеджера тут много дел. Причем, ценность большинства из них воспринимается клиентом весьма гадательно. И хорошо бы что-то делегировать.

> Час нормального проджекта должен стоить не менее часа нормального психолога.

1.1.2. Кривая роста мастерства

Как выглядит кривая роста компетенций и профессионализма руководителя? А это действительно кривая.



Кривая роста мастерства

1. Бурный рост

Для человека все ново, он заинтересован и очень быстро осваивает то, что называется «язык отрасли». Это как игра на гитаре, мальчики поймут: вам показали пять аккордов, вы

освоили их за три дня и, в принципе, большинство песен во дворе уже можете петь под гитару. Все девушки ваши.

Хотя уже и на этом этапе часть людей отвалится, потому что поймут – это не для них. Такой начальный профессиональный сплит-тест – «да-нет».

2. Страшно и ни черта не понятно

Дальше расти сложнее. Надо барре ставить, табулатуры разбирать, или (боже упаси) ноты учить. И получается, что усилий нужно прилагать все больше и больше, а видимых достижений – все меньше и меньше. Уменьшается и количество людей, которые могут оценить ваши успехи. Страшно и ни черта не понятно.

Здесь-то многие и сдаются. Особенно юные быстро обучаемые дарования. Интересней нахвататься по верхам, стать лучшим математиком среди гуманитариев или лучшим менеджером среди программистов. Лучшим летчиком среди парашютистов и так далее.

В психологии известен эффект Даннинга-Крюгера. Смысл его в том, что дилетант ведет себя, как правило, очень самоуверенно. Он чего-то нахватался, каких-то догм, и свою уверенность строит на этих догмах.

В свою очередь более профессиональный человек начинает терзаться сомнениями, поскольку уже отходит от догм, видит их ограничения, исключительные ситуации и нюансы. Начинает думать не так категорично. Поэтому со стороны порой кажется, что знания новичка, его повадки – более убедительны, чем знания профессионального специалиста, который говорит с большим количеством оговорок. Но со временем, с опытом, если человек прогрессирует – неуверенность уходит. Когда человек становится экспертом – он не так категоричен, как новичок, но уже точно знает, что сработает, а что – нет.

В росте компетенций менеджера такие перепады настроения – от ощущения всемогущества до ощущения собственного бессилия – абсолютно нормальная история. Более того, если вы таких перепадов не испытываете – значит, перестали расти.

Что делать? Просто продолжать, несмотря на страх и неясность. В какой-то момент картинка встанет на место. У кого-то этот путь занимает полгода, у кого-то – год. При этом нужно уделять больше времени практической работе на боевых проектах, а не теоретизированию.

3. Ясность мысли

Итак, после того, как с «детскими болезнями» разобрались, менеджер постепенно начинает набирать авторитет, опыт, компетенции. Вырабатывает, что называется, стиль.

Он хорошо знает правила компании, методологии, и мало-помалу начинает успешно вести проекты.

И вот тут, после первых нескольких успешных кейсов, его ждет ловушка. Дело в том, что многие вещи для менеджера уже привычны. Если раньше, например, он кропотливо работал по правилам, старательно фиксировал договоренности, мало импровизировал – то с какого-то момента он начинает чувствовать себя уверенным и крутым. И начинает эти правила нарушать.

4. Яма

Менеджер не справился с управлением, но вовремя ушел на больничный.

Стратегия полу-управленца

Менеджер почувствовал силу и начинает экспериментировать с правилами и подходами. И в какой-то момент у него начинают «сыпаться» проекты.

Например, сначала, работая с клиентами, вы все им подробно объясняли. Потому что вам самим нужно было проговорить, что и как будет. А потом вам это стало ясно-понятно. И вы перестали объяснять. И врезались в стену непонимания. Вам все очевидно, а клиенты стали как-то тупить (так это будет выглядеть для вас) и стали несговорчивыми. Причем, резко, все разом. Обычно критическая масса косяков копится, и все они вываливаются на разных проектах, но примерно в одно и то же время.

Как-то один из наших руководителей проектов пожаловался, что клиенты не понимают очевидные вещи: что такое верстка, чем она отличается от дизайна и программирования. Начались недопонимания – за что выставляется каждый конкретный счет. Приходится тратить время на урегулирование этих непониманий, а времени лишнего нет. Как выяснилось, руководитель проекта где-то поймал установку «раз уж мне очевидно – значит и клиенты должны знать, как все устроено». Однако это так не работает. Мы разобрали ситуацию. Поняли, что без резкого снижения нагрузки на менеджера нам не вырулить. Передали два проекта другим руководителям и составили «график некидалова» – недельные планы, в которых менеджер имел права заниматься не более чем двумя проектами в день. Но полноценно и сосредоточено, с созвонами, отчетами и временем на обучение клиента. Примерно через 6 недель ситуация стабилизировалась: менеджер стал меньше уставать, клиенты начали ставить хорошие оценки за этапы. Но если бы ситуацию пустили на самотек – менеджер вряд ли справился бы самостоятельно.

Это – яма. Из нее выкарабкиваются не все. Где-то семь-восемь из десяти. Тут некоторые, скажем так, кривые на душу люди, проявляют себя не с лучшей стороны, показывают свое «искусство вовремя уйти в сторонку». Причем, желательно «пока еще все хорошо». Кровь и ошметки приходится расчищать другим.

Яма неизбежна. Лучше всего, если в этот момент рядом будет опытный наставник, который вовремя протянет руку и из этой ямы вытащит.

Карабкаться сложно. Моя рекомендация: если вам невмоготу, если вы понимаете, что запутались – поговорите об этом с вашим начальником открыто, поищите вместе пути решения. В большинстве случаев вы сможете составить план и за пару месяцев выйти из кризисной ситуации, многому для себя научившись.

И даже если вы решите, что дальше с этой компанией вам не по пути – не бросайте за собой горящие города. Да, по закону вы можете махнуть рукой или шашкой: «А любись оно все конем», и через две недели ощутить вкус свободы. Но таких блуждающих менеджеров на рынке много. И кармически правильно будет привести дела в порядок, пускай это займет и два, и три месяца вашей плотной работы с вашим руководителем. Так правильно, так экологичнее. Вы большему научитесь и сможете сохранить лицо.

К счастью, большая часть людей яму преодолевает.

5. Мастерство

После кризиса уровень менеджера радикально меняется. У него появляется внутренняя уверенность. Он находит решения даже в сложных ситуациях. Такие менеджеры могут не только делать проекты, но и научить других. Это – опора компании. Факапы на этом уровне, безусловно, останутся, но справляться с ними будет уже легче.

На этом этапе руководитель проектов ищет задачи сложнее, интереснее. Он выступает на конференциях и тестирует новые технологии. Дорога в мир настоящих, жирных факапов!

Тут появляется риск перегореть – это когда после рабочего дня ничего не хочется, а сил хватает, только чтобы смотреть в стену и молча ненавидеть людей. Возникает желание уйти в монастырь, заняться духовными практиками или стать буддой. В общем, уйти в себя. Что делать – жизнь-боль, а в конце мы умираем.

На этой стадии очень важно научиться быстро расслабляться и быстро концентрироваться на задачах. Попробуйте спорт, хобби, медитации, путешествия. Кому что. Важно найти баланс между отдыхом и работой и перейти в режим гроссмейстера – когда задач решаете много, но в каждый конкретный момент сконцентрированы.

Самое интересное, что на этом развитие не заканчивается. Управленческое мастерство ничем не ограничено, нет предела совершенству, но к нему нужно стремиться.

1.2. Делегирование

Делегирование – это передача части полномочий или ответственности другим людям. Как правило – своим подчиненным.

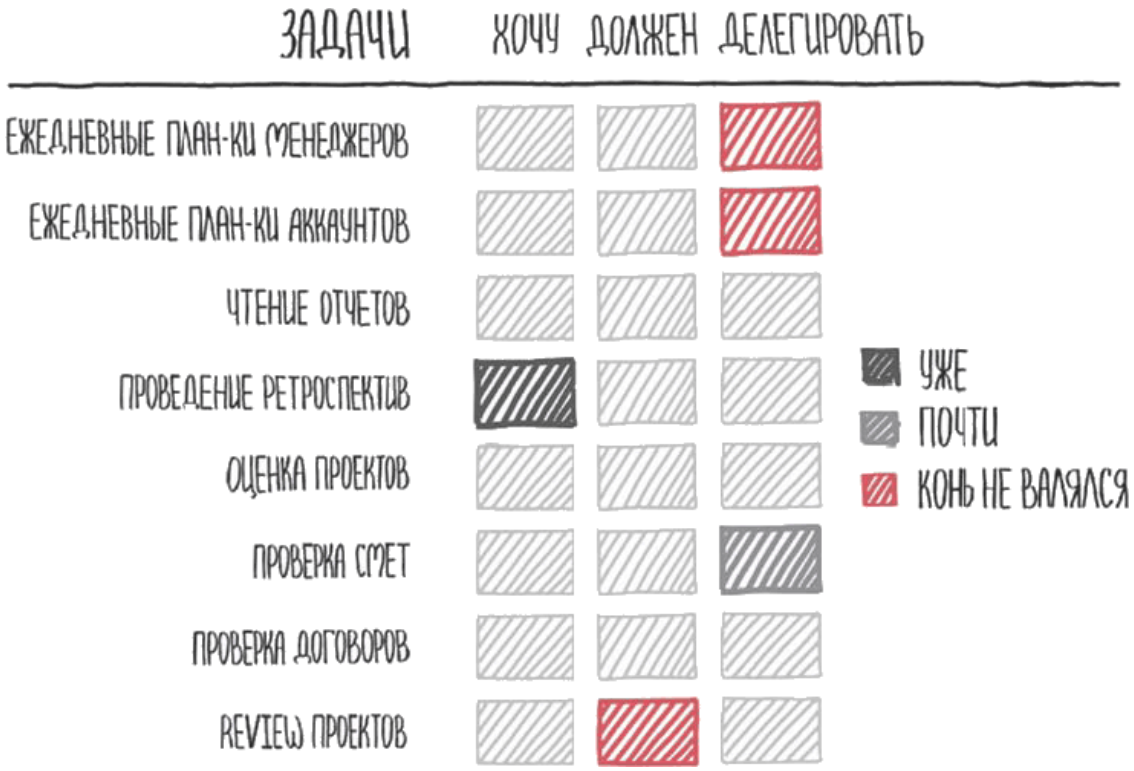
Делегирование невозможно без грамотного планирования. Банально, у вас не останется времени ни на постановку задач в нужной форме, ни на контроль. О планировании времени мы поговорим в главе 8.

Делегирование – ахиллесова пята российского менеджмента. Руководители делегируют плохо, боятся или ленятся это делать. Многие просто не умеют, поскольку делегированию, как и вождению автомобиля, нужно учиться. Менеджер, который не умеет делегировать, – бутылочное горлышко организации: он набирает на себя слишком много задач и стремится самостоятельно сделать их большую часть. Грамотное делегирование – это точка роста организации.

Вы как менеджер можете вообще ничего не делать своими руками и ничего не производить. Но при этом вы отвечаете за выработку управленческих решений, распределение задач, нагрузку своих подчиненных. Делегирование – это передача подчиненному задачи вместе с необходимыми для ее выполнения ресурсами. Например, полномочиями и ответственностью: за качество, сроки и другие явно или неявно согласованные параметры.

1.2.1. Что делегировать. Делегирующая сила

Я завел себе регулярную привычку – раз в полгода просматриваю список дел, которыми занимаюсь. Смотрю, сколько времени они у меня отнимают. Оцениваю, насколько они мне нравятся и насколько они стали рутинными. Смотрю, должен ли я их делать сам. Можно ли делегировать что-то из рутины, неприятных мне дел или тех задач, где лучше справится специалист. Планирую: когда, кому и что я могу делегировать и в каком виде.



Помогает табличка – что можно, а что нельзя делегировать:

Что можно делегировать	Что нельзя
Рутина (тестирование, перенос задач в тикет-систему)	Планирование, целеполагание, выработка стратегии, контроль результатов
Специализированная работа (программирование, дизайн)	Руководство сотрудниками, мотивация, поощрения и наказания
Подготовительная работа (подготовка отчетов, договоров)	Задачи с высокой степенью риска, неопределенности, новизны, необычности или задачи личного (строго доверительного) характера
	Горящие дела, без запасов времени на делегирование и проверку

Советую завести себе подобный ритуал или регулярную задачу в вашем таск-трекере (планировщике). И постепенно увеличивать вашу делегирующую силу.

1.2.2. Устно или письменно

И тот, и другой вариант уместен, но у обоих есть ограничения.

Основной плюс устного делегирования – это быстрее. Годится только для небольших, ясных и привычных заданий. Для команд, которые все понимают одинаково. Для сложных, непривычных заданий есть минусы:

▶ Сложно учесть все детали при постановке. Кто мало думал, тот много плакал.

▶ Подчиненному сложно запомнить множество деталей.

▶ Может аукнуться временем на переделку и повторным делегированием.

▶ Не оставляет артефактов. Появляется искушение при разборе полетов переиграть постановку («а я тебе сказал так-то», «а я тебе этого не говорил…»). Очень плохая практика, которая разлагает рабочую атмосферу. Да и в эту игру могут играть оба.

Письменное делегирование дает время обдумать детали. В итоге получается точнее. А размышления о предстоящих деталях минимизируют количество ошибок. Минусы:

▶ Требует времени на написание ТЗ.

▶ У подчиненного нет возможности уточнить задание в момент его получения.

▶ У вас нет возможности предварительно убедиться, что вас поняли правильно. Пропущен предварительный контроль.

Лучше работают гибридные варианты.

Текст + обсудить. Для сложных заданий – с чек-листами, условиями, пунктами и подпунктами – сначала пишем текст, потом даем прочитать и обсуждаем устно вопросы. Идеально для постановки задач программистам.

Если деталей в задаче слишком много, лучше разбить такие задания на несколько. Проще пропустить один пункт из 30, чем один пункт из пяти.

https://glvrd.ru/
Сервис «Главред»

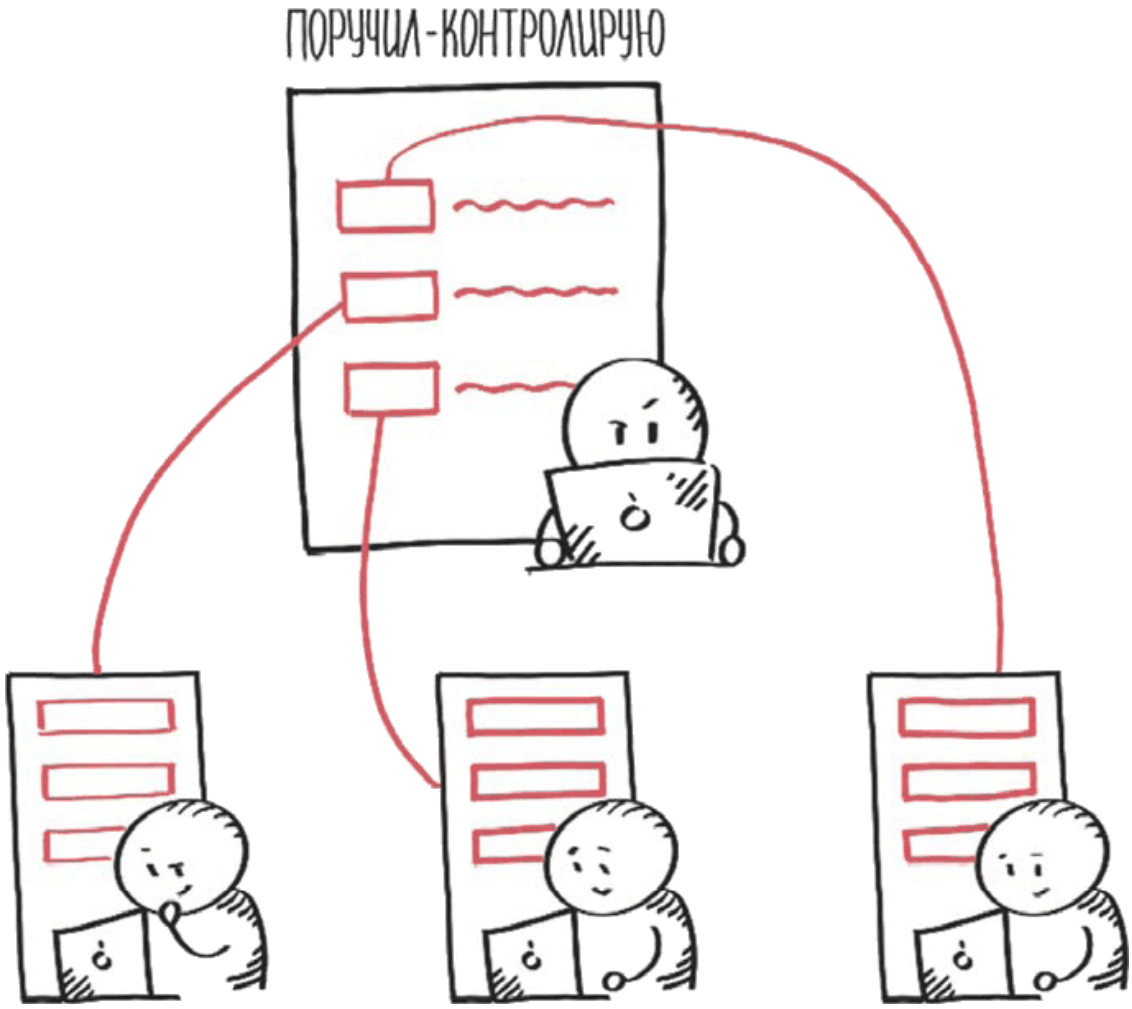
Выработайте ясный, четкий стиль. Помогает книга Ильяхова «Пиши, сокращай» (в конце главы есть список литературы). Можете даже первое время для тренировки прогонять постановки через сервис glvrd.ru – программа дает оценку качества стиля. Пользуйтесь сервисом, пока не почувствуете, что справляетесь без него. Постановки длиннее трех-четырех предложений навевают тоску. Не стоит писать длинные инструкции из-за паранойи и личных страхов.

Устно + проверка фиксации. Дать устное распоряжение или даже провести брейншторм, обсудить вопросы, спросить план действий и проверить, как задание было зафиксировано. Тут работает «эффект генерации» – мы лучше запоминаем то, что изложили сами. Можно зафиксировать итоги и ключевые точки самому, но это не так эффективно. Большинству клиентов удобно давать постановки в таком виде. Хорошо работает с дизайнерами, да и в принципе удобный способ.

1.2.3. Список «поручил-контролирую» и сроки о сроках

При устном делегировании вам нужно будет каким-то образом вспомнить о своем задании и своевременно его проверить. Я держу в личном таск-трекере список «поручил-контролирую», где кратко, пофамильно записано: кто что обещал и когда (или когда будет контрольная точка).

Список «поручил-контролирую» удобно вести в любом тудушнике или блокноте.


ПОРУЧИЛ-КОНТРОЛИРУЮ

Подобную же технику можно организовать в бумажном планировании – стопка стикеров, каждый из которых разбит на четыре сектора: что, кто, когда и отметка «сделал». В том смысле, что сделал вовремя, как и обещал. Раз в неделю отработанные стикеры выбрасываются. Или можно накопить статистику по людям: кто подводит вас чаще всего, а кто – четкий.

При этом я понимаю, что в режиме «здесь и сейчас» не всегда получится спрогнозировать дату выполнения задания. Но всегда есть возможность спрогнозировать срок получения срока. С этим списком проходят планерки. Ребята знают, что я вряд ли что-то забуду. Повышает качество фиксации заданий с их стороны.

Кроме того, мне нужно время на проверку задания. Поэтому, если договорились сделать задачу «сегодня», то по умолчанию я ожидаю ее до 17:00 (а не в 23:59).

Рекомендую завести себе такой список, задавать вопрос про сроки о сроках, а также определиться, что значит «будет готово сегодня».

1.2.4. Где делегировать, а где – нет. Туалетное делегирование

Туалетное делегирование – это когда сотрудника в неформальной обстановке (в курилке, туалете, на кухне или прогулке с собакой после работы) коллеги нагружают задачами. Задачами правильными, нужными. Но когда человек к этому морально не готов и сконцентрирован на том, чтобы НЕ работать. Из-за этого он может забыть даже факт того, что его о чем-то просили. И еще виноватым останется. Коллеги же, которые задачу выдали, чувствуют, что свой долг выполнили.

Таким образом, у туалетного делегирования три признака:

1. Неформально. Задача выдается в неформальной обстановке. Курилка. Кухня. Коридор. Любое нерабочее пространство.

2. Корректно. Задача выдается по адресу и корректно. Если не исполнили – понятно, кто не исполнил. И понятно, что именно.

3. Безвозвратно. Поставивший задачу сотрудник считает, что выполнил свою миссию и повторять информацию в рабочем пространстве уже не будет.

Туалетное делегирование встречается в любых коллективах, от «Газпрома» до семьи (мама попросила помыть посуду, когда сын был в разгаре онлайн-боя – как же это запомнить?). И всегда заканчивается обидами и расшатыванием авторитета озадаченного.

Так что практиковать такое делегирование – дело тухлое. Часть задач постоянно будет теряться. Уважение к человеку, которого нагружают задачами – тоже. Поэтому, если у себя в компании вы заметили, что подчиненные делятся задачами в нерабочее время или в нерабочем месте, боритесь с этим.

Туалетное делегирование – очень тонкая вещь. Подчиненный может подойти с ВОПРОСОМ (дескать, как мне быть с компанией А) – руководитель отмахнется фразой «надо подумать». И тут подчиненный получит моральное право считать, что ситуацию с компанией А теперь разруливает его руководитель. Итог: через пару дней подчиненный с чистой совестью подойдет и спросит: «Глебываныч, как там с моим ВОПРОСОМ дела?» Спасибо, что не: «Как там с моим ПОРУЧЕНИЕМ дела». Это самозахват полномочий. За такое надо бить, и бить жестко!

В отличие от ***пересаживания обезьян***, туалетное делегирование работает в две стороны. Делегировать можно не только руководителю, но и сам начальник может бросить мимоходом: «У меня идея! Круто бы было, чтобы у нас на проекте была такая-то фича!» Подчиненный кивнет, скажет: «Да, реально прикольно». Итог: начальник считает, что задача поставлена. Подчиненный – что просто поговорили. Через некоторое время обоих ждет большой облом.

Решение

Избегайте делегирования в нерабочих местах. Иначе подчиненный не сможет зафиксировать постановку и взять на себя обязанность. Лучше всего подходят запланированные встречи, один на один (например, регулярные планерки раз в неделю-две или ***стендапы***). Да, вы имеете право дернуть подчиненного в любой момент, но это непрофессионально и говорит о вашем слабом планировании. А подчиненного выбивает из потока – о важности работы в потоке подробнее поговорим в главе 2. Важно оставлять время на разбор вопросов и задавать конкретные контрольные вопросы о деталях. Так вы поймете, что подчиненный все правильно понял и составил адекватный план действий. Следите, не пытаются ли делегировать на бегу вам. Введите в коллективе термин «туалетное делегирование» и пресекайте это явление.

1.2.5. Саботаж принятия задачи. Вопросы ради вопросов. Вяленький алгоритм хранителей космоса

Есть, говорят, такая секта – «Хранители Космоса». Предсказав однажды конец света, они побросали семьи, квартиры, работы и что там у них еще было. И стали ждать часа Х, неистово молясь.

Настал час Х. Земля не взорвалась. Пришельцы никого не забрали. Небеса не разверзлись.

Вы думаете, они сильно расстроились и разошлись по домам? Хренушки! Хранители космоса объявили, что ТОЛЬКО БЛАГОДАРЯ ИХ МОЛИТВАМ мир получил отсрочку! Хитро, не поспоришь.

Алгоритм такой:

1. Начиная новое дело, объявить о возможных негативных последствиях. Что уже само по себе КАК БЫ дает право не верить в результат и действовать деструктивно.

2. Вяленько что-то делать. Право действовать «вяленько», не напрягаясь, человек как бы отвоевал себе своим скепсисом и неверием в результат.

3. В случае наступления негативных последствий – высунуться и сказать: «А Я ЖЕ ГОВОРИЛ!!!» (Если, конечно, будет кому «а-я-же-говорить»: в случае с Хранителями Космоса и спрашивать, и отвечать было бы некому.)

4. В случае, если негативных последствий не наступило, говорить, что это ТОЛЬКО БЛАГОДАРЯ МОЕЙ САМООТВЕРЖЕННОЙ РАБОТЕ.

Алгоритм часто применяют неосознанно. Он вшит где-то на уровне BIOS-а в мозгу (см. А. Курпатов, «Красная Таблетка»). В digital этим особенно грешат программисты. Иногда пропуская 4-й пункт, ибо он как бы подразумевается.

Противоядие:

1. Знать, что Хранители Космоса не угомонились, и что их алгоритм, зараза, действует.
2. Поймать человека на применении этого алгоритма. И рассказать, что именно он делает. При необходимости – повторить. При острой необходимости и частых «А-Я-ЖЕ-ГОВОРИЛ!» поинтересоваться, не сектант ли он.

Подобная же схема работает, когда подчиненный задает много вопросов ради вопросов, уводя разговор в очень частные случаи, запутывая и как бы намекая, что начальство не продумало все до конца. Таких хитрецов надо заставлять формулировать вопросы письменно, заранее. И отказываться обсуждать с ними вопросы, которых не было в списке.

Тех, кто регулярно включает дурачка («А почему вы не сказали, что нельзя сушить кота в микроволновке?»), неплохо бы спросить, что он этим хочет вам доказать? Вы же ведь можете и поверить…

1.3. Уровни делегирования

1.3.1. Делегирование на уровне идеи

Подойдет для делегирования подчиненному с примерно таким же, как у вас, уровнем компетентности и ответственности, со схожей мотивацией. Вы ему доверяете, он ведет дела в том же стиле, что и вы. Таким людям можно не разжевывать задачу на атомы – хватит и постановки на уровне идеи, они сами ее «докрутят» и сделают все в лучшем виде.

Руководитель: Давай организуем семинар для клиентов?
Сотрудник: Давай, а на какую тему?
Р: Не знаю, что-нибудь про интернет-магазины.
С: Ок, в июне есть время. Давай я придумаю тему, составлю план подготовки, можем обсудить.
Р: Да сам как-нибудь самоорганизуйся:)
С: Ок.

1.3.2. Делегирование на уровне тезисов

Такой способ делегирования тоже подойдет для человека из предыдущего пункта. Единственное отличие – уверенности, что тот поймет все правильно, у вас чуть меньше. Таким людям, помимо идеи, нужно задать направление.

Р: Я хочу провести семинар, как в прошлом году, на 300–400 человек. Придумай тему, напиши план подготовки и какие ресурсы тебе понадобятся для этого.
С: Ок.

1.3.3. Делегирование на уровне задач

Так делегируют сотрудникам с высоким уровнем квалификации, если они уже решали задачи подобного уровня. И у которых представление о выполнении задачи сходится с вашим.

Р: Нужно организовать семинар, как в прошлом году, на 300–400 человек. Тема – интернет-магазины.
С: Что требуется от меня?
Р: Подбери зал на 300 человек в пределах 1-го кольца и не дороже 100 000 рублей.
С: Хорошо, я составлю список подходящих, через два дня покажу – обсудим.

1.3.4. Делегирование на уровне инструкций

В остальных случаях придется разжевывать. Особенно, если дело незнакомое, рисковое и ответственное, где в результате можно либо многое приобрести, либо многое проиграть.

Совет

Там, где уровень риска высокий, степень ***декомпозиции*** и детализации задач тоже должны быть высокими.

Р: Мы организовываем семинар. Вот список клиентов, вот ***скрипт***. Тебе нужно будет обзвонить клиентов по списку, используя этот скрипт. Сейчас ты заучиваешь его, подходишь

через 15 минут ко мне и воспроизводишь его слово в слово. Если все ок – начинаешь обзвон, вот телефон. Тех, до кого удалось дозвониться, помечаешь зеленым. Как понял?

С: Взять скрипт, взять список, скрипт выучить, подойти и через 15 минут сдать его тебе, обзвонить. Потом пометить тех, кому удалось дозвониться, зеленым.

Р: Тебе всего хватает для этого?

С: Да, все есть.

Р: А зеленый маркер?:)

Со стороны может показаться, что делегирование на таком уровне – это маразм. Дается много вводных, которые сложно запомнить и удержать в голове, поэтому тикет-системы и письменные постановки здесь спасают.

Но при этом вы не застрахованы от неверного выбора уровня делегирования, и это может произойти как устно, так и письменно. Если вы делегируете на слишком низком уровне и слишком детально описываете задачу более компетентному сотруднику, это может его обидеть, разозлить или привести к саботажу. Чем больше в задаче будет нюансов, пунктов и детальных описаний, тем выше риск потерять какой-то из этих пунктов в процессе работы.

1.4. Smart-делегирование

Популярный ***фреймворк*** постановки задач. SMART – аббревиатура, где каждая буква обозначает:

- ▶ Specific – конкретный;
- ▶ Measurable – измеримый;
- ▶ Achievable – достижимый;
- ▶ Relevant – значимый;
- ▶ Time-bound – ограниченный во времени.

Конкретность – хотим, чтобы сотрудник выкопал яму. Измеримость – яма должна быть метр на метр, в сквере. Достижимость – за счет каких ресурсов можно достичь результата: лопата лежит в сарае, для копания можно привлечь разнорабочих, бюджет 2000 рублей. Значимость – подчиненным важно объяснить, зачем мы достигаем этого результата, и как этот результат связан с целями компании: яма нужна, чтобы посадить дерево, это нужно для поддержания имиджа экологичной компании. Ограниченность во времени – указать конкретные сроки: через 3 дня, потому что идет сезон посадки деревьев.

Цели, которые вы ставите, должны быть трудными, но выполнимыми.

SMART – это популярный метод постановки задач, но это не панацея. Кому-то нужно более детально разжевывать, кому-то – менее. Метод хорошо вписывается в классический цикл Деминга «планируй-делай-проверяй-воздействуй». Только «делать» будет кто-то другой:

Планирование

Предполагает, что вы оставляете в графике время на постановку задач и доведение их до подчиненных (обсуждения, брейнштормы), а также время на контроль. Если вы не запланируете на эти процедуры достаточного количества времени, не удивляйтесь, что делегирование идет как-то криво, а задачи выполняются халтурно.

К планированию относится и выбор исполнителя: важно учитывать загрузку сотрудника, рабочий график, уровень знаний и умений. На этом этапе вы также определяете параметры задания в зависимости от уровня исполнителя: постановка на уровне целей, тезисов, задач или жесткое «упал-отжался». И вот здесь, если постановка должна быть более детальной, пригодится SMART-модель. Возможно, параметры задачи придется разобрать по приоритетам: на обязательные и желательные.

Количество времени на делегирование напрямую зависит от исполнителя: опытный сотрудник, который поймет на уровне идей, или

новичок, которому придется давать детальные указания. Поэтому выгодно повышать уровень компетентности сотрудников – это снижает управленческие издержки. Но если люди жалуются, что требования – невменяемы, стоит к ним прислушаться и изменить уровень детализации.

При планировании вы определяете методы и ресурсы и проговариваете цели, реальные и достижимые. И, конечно, формируете сроки, которые тесно взаимосвязаны с методами и ресурсами. Например, яму можно выкопать экскаватором – это быстрее, но стоить будет дороже. И не забывайте о разнице между трудоемкостью и финальными сроками выполнения задачи. Сотрудник может сказать, что на задачу уйдет пять часов, но при этом у него не получится заниматься ей весь день из-за вороха других задач – из-за этого результат вы увидите через пару-тройку дней.

Контроль

Контрольных точек может быть несколько – единого правила, определяющего их количество, нет. Зависит от параметров задачи, уровня сложности, уровня компетенции подчиненных. Бывает, что может потребоваться итеративная работа по контролю (свят-свят). Бывает, что задача новая, неизвестная – и тогда приходится уточнять по ходу работы и сроки, и другие параметры. После прохождения этой стадии сроки уже фиксируются железобетонно.

Санкционирование

По итогам контроля нужно проводить управляющие воздействия: поощрять или наказывать, давать обратную связь либо менять параметры задания. Например, сменить исполнителя, поскольку тот делает задачу критически медленнее плановой скорости.

1.5. Постановка задач на рисерч

К счастью, мы работаем не в конвейерной сфере, и у наших разработчиков и дизайнеров есть поле для творчества. К сожалению, это означает, что не все задачи мы можем декомпозировать и оценить, даже примерно. Появляются новые, незнакомые технологии или темы, с которыми мы не сталкивались. В этом случае разумно выделить время на исследование. Результатом такого исследования должен стать план действий и оценка необходимых ресурсов.

Исследование требует ресурсов, иногда – серьезных, и если задача новая и неизвестная, то процесс оценки стоит разбить на стадии:

1) Постановка задачи, анализ и предварительная оценка

Например, задача – анимированная 3D-модель, а вы такого раньше не делали. Предварительно вы закладываете 8–16 часов на всю эту задачу.

2) Фаза рисерча

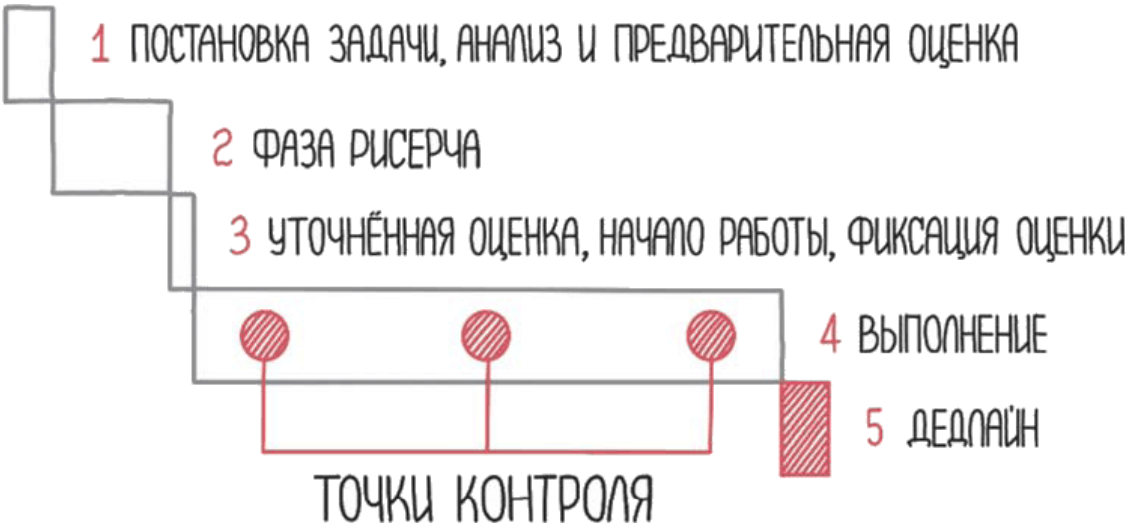
Мы берем себе час-два на «поиграться» с какой-то технологией или новым ***стэком***, провести нужные эксперименты. В примере про 3D-модель нам нужно будет посмотреть, как «запекаются» текстуры, посмотреть фреймворки для 3D-анимации, изучить софт, посмотреть примеры. В итоге оценка может увеличиться в 4 раза, или вы поймете, что только на стадию рисерча вам нужны исходные 8–16 часов. Бывает. Возможно, в этот момент вы упростите задачу, обойдетесь без анимации. Или поменяете исполнителя, найдете опытного. Или добавите ресурсов. Я называю это «контролируемый факап». Провал возможен, но глубина провала известна заранее.

3) Уточненная оценка, начало работы, фиксация оценки

Когда вы поняли, с чем имеете дело, оценки фиксируются железобетонно.

4) Выполнение задания. Согласованные контрольные точки

Дальнейшие изменения оценки возможны только как форс-мажор или как косяк. И как только появился какой-то затык, нужно немедленно о нем сообщить. В длительных и сложных задачах также согласовываются контрольные точки.



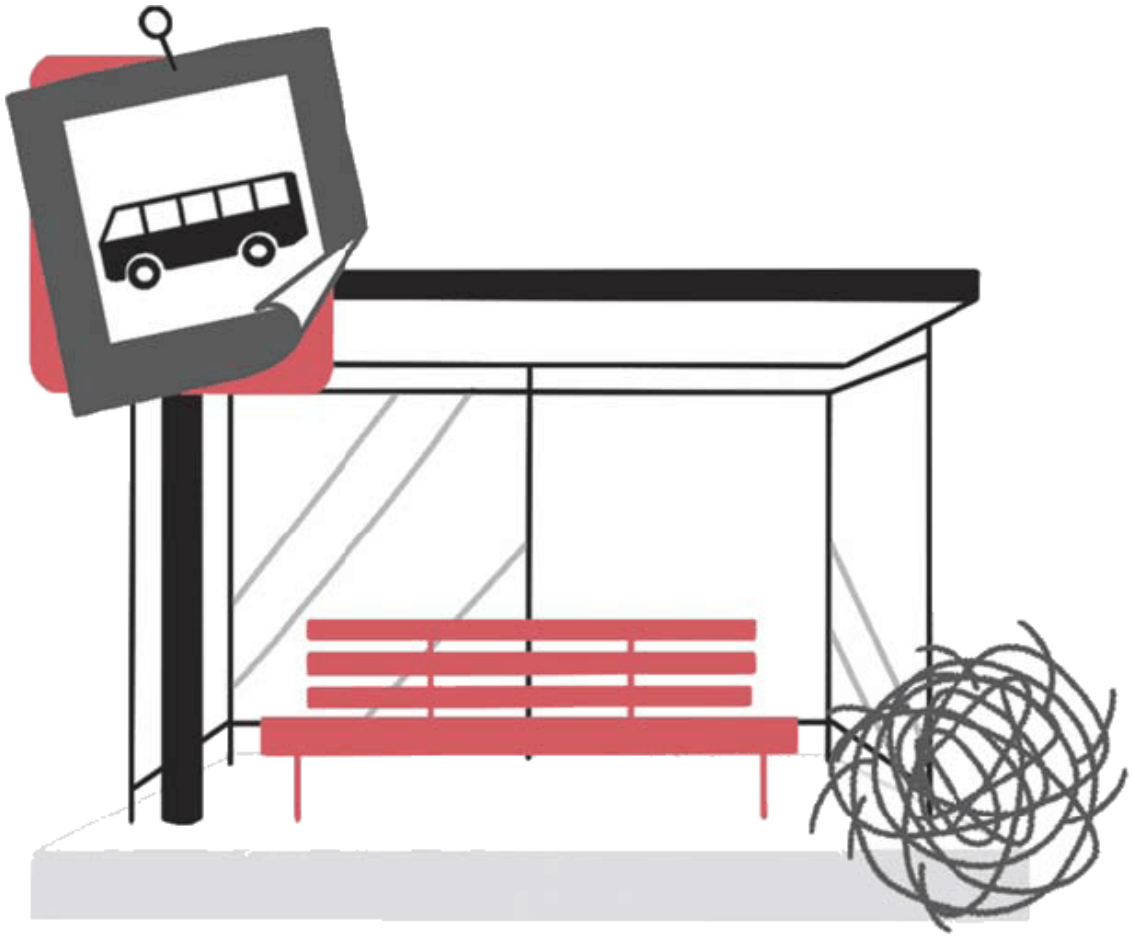
На первом и втором этапе оценки и прочие параметры можно менять. Начиная с третьего – уже нет.

Если у вас нет навыков в сфере, в которой нужно делать декомпозицию, ее нужно поручить сделать подчиненному с соответствующими навыками. Результат получить в письменном виде. И если что – проверить сторонним специалистом, мнению которого вы доверяете.

1.6. Фальшивые автобусные остановки. Тонкое искусство контрольных точек

Есть такое явление – фальшивые автобусные остановки. На них висит расписание, но автобус к ним никогда не приезжает. Остановки устанавливаются на территории или рядом с домами престарелых или другими клиниками, где содержатся люди, страдающие старческим слабоумием. Пациент, решив сбежать из стационара, часто остается на фиктивной остановке ждать автобус в уверенности, что он сможет оттуда уехать. Здесь его может обнаружить или догнать медицинский работник.

Это яркий пример, как стереотипы и привычки можно применять для управления и выстраивания контрольных точек. Старческое слабоумие – момент далеко не ключевой.



Пример из практики: один мой знакомый очень любит внедрять KPI. Из пяти метрик одну делает фальшивой. В реальной работе получить ее нельзя. Хитрецы же могут накрутить любую херабору.

Другой пример: как-то я подготовил чек-лист, первым пунктом которого было: «Внимательно прочитать документ до конца». Дальше было примерно 30 пунктов. Последним был «Выполнить пункты 1 и с 5 по 12, остальные проигнорировать». Жестко, да?

Нет четкого алгоритма расстановки контрольных точек. Их не должно быть очень много. Это лишняя нагрузка на вас и раздражитель для исполнителей. Попробуйте мысленно представить, как выглядит провал по задаче. Теперь мысленно представьте, что было до этого про-

вала. Попробуйте найти такую точку, где провал уже возможен, но ситуацию можно исправить. Постарайтесь использовать «предсказывающие» контрольные точки.

Например, можно отслеживать, что исполнитель **закончил** задачу (и тогда о провале вы узнаете без запасов времени), а можно – что **приступил**. В этом случае хотя бы синдром студента – прокрастинация до последнего – вашему исполнителю не грозит. Договориться о контроле лучше заранее, во время делегирования. Избыточный или внезапный контроль бесит умных, талантливых людей.

> Используйте предсказывающие контрольные точки. Оговорите их заранее.

1.6.1. Контрольные точки методом поводка

Посмотрите, как собаководы выгуливают своих питомцев. Ведет себя хорошо – дают полную свободу. Отпускают с поводка. Ведет хуже – поводок натягивается. И чем хуже – тем короче поводок. А могут и за ошейник взять.

Вот так и с контрольными точками. Справился ваш подопечный – молодец! Даем больше свободы. Большие сложные задачи, меньше контроля. Не справился – значит, пока рановато такие задачи решать. Уменьшаем поводок. Задачи мельче и проще, больше контрольных точек.

1.7. Ошибки делегирования

Но если все ясно, то почему руководители не делегируют или делегируют через ж… плохо?

1.7.1. Неумение делегировать

Бывает, когда руководитель – адреналиновый наркоман, которому нравится работа в аврале. Когда все срочно, все важно и все горит. Проблема в том, что, как и с любой зависимостью, уровень адреналина нужно постоянно повышать, иначе становится пресно и неинтересно.

И тогда человек накидывает на себя все больше задач, а сотрудников погружает во все больший форс-мажор – чтобы только опять все «горело». Как следствие, в какой-то момент менеджер превращается в супермена, который прилетает и решает мирские проблемы. Собственно, ради этого ощущения «на острие атаки» весь аврал и затевается. Другой бонус – алиби очень занятого человека и повод для оправданий в духе «я впахивал и сделал все, что мог». Иными словами, «я в домике».

Вторая причина так-себе-делегирования – ЧСВ. Чувство собственной важности. Это когда руководитель намеренно ставит задачу так, чтобы подчиненный сделал ее плохо или не сделал вообще. В итоге руководитель приходит к сотруднику, отодвигает его в сторонку и начинает делать сам с гордым выражением лица «смотри, как надо». К сожалению, в российских реалиях это встречается чаще, чем хотелось бы.

Другой вариант – руководителю страшно хочется что-то сделать самому. Ну вот прямо руки чешутся. Стоит вспомнить, что многие руководители вырастают из хороших специалистов. А это люди, которые годами нарабатывали свою квалификацию и которым сложно взять и вот так просто расстаться с этими навыками. Время от времени это нормально, если руководитель добивается результата своими руками, и при этом большая часть работы делается руками других людей через делегирование. Но важно отдавать себе отчет: для чего вы делаете эту работу. Просто чтобы поддержать свою квалификацию или потому что вы уклоняетесь от руководства?

Понятно, что не бывает без форс-мажоров, когда нужно все бросать и лично подключаться к команде, чтобы задача была решена вовремя, и по ней все было хорошо. Единственная проблема – большую часть форс-мажоров создаем мы сами, своими руками: например, неправильным планированием своего времени и времени подчиненного. А потом врем себе, что виноваты звезды.

1.7.2. Боязнь идти на конфликт

Подробнее об этом поговорим в главе 2, про власть и мозгоклюйство. Одна из частых причин избегания делегирования – неуверенность в моральном праве давать распоряжения. Вчера вы сидели за одним столом, ходили вместе на обед, а сегодня – вы уже руководитель, а он – подчиненный. И с этим теперь надо как-то жить.

Другая причина – нежелание столкнуться с сопротивлением. Действительно бывает, что подчиненный либо морально сильнее вас, либо превосходит по уровню эмоционального интеллекта. Поэтому, если появляются какие-то проблемы по задаче, возникает желание сделать ее самостоятельно – лишь бы не провоцировать конфликт, навязывая кому-нибудь свою волю.

Часто молодые руководители проектов опасаются испортить отношения с подчиненными – эдакое желание быть хорошим и добрым для всех. Со временем проходит, но не у всех. И, конечно, менеджеры часто стремятся избежать публичных ошибок. Когда вы отдаете в распо-

ряжение сотруднику какую-либо задачу, вы можете ошибаться в ее постановке или в технических нюансах. Либо ваше распоряжение может привести к негативным последствиям. Когда происходит ошибка или такие последствия наступают, и вам на них указывают, это всегда удар и по власти, и по чувству собственной значимости. Самое противное – потом кто-то может долго припоминать эту ситуацию.

1.7.3. Неуверенность в своих подчиненных

Для неуверенности могут быть разные причины. Подчиненные могут быть слишком загружены – и тогда из-за неуверенности в том, что задача втиснется в чей-то график, руководитель ее просто не ставит.

Также вы можете быть не уверены в квалификации подчиненных. Но по-хорошему грамотный руководитель должен заниматься постоянным мониторингом квалификации сотрудников и наращивать ее до необходимого уровня. Другая крайность – неуверенность в мотивации подчиненных: а хватит ли им мотивации выполнить поставленную задачу?

Еще один вариант – сомнения, что сотрудники верно понимают поставленную задачу и сделают ее именно так, как вы того хотите. Бывает, что вы даете распоряжение в общих словах. Например, «поедь к Иван Иванычу, возьми такой-то документ и привези его мне». И вот, приходит подчиненный и говорит: «Не получилось». Он не съездил, а позвонил. Иван Иваныч был на совещании и не перезвонил. А вы-то знаете, что Иван Иванычу звонить бесполезно – он такой человек, к которому нужно именно приехать. И здесь не остается никаких рычагов, кроме режима бармалея: «Делай так, а то уволю!» Просто потому что на рассказ о всех тонкостях взаимоотношений с Иван Иванычем у вас уйдет полдня.

1.7.4. Нет достаточных управленческих качеств

Если руководитель не умеет планировать свое рабочее время и время своих подчиненных, не оставляет места для постановки задач и для контроля, то ничего удивительного, что делегирование начнет сыпаться. Для контроля за выполнением работ действительно нужно закладывать время, а после контроля – давать обратную связь и следить, чтобы эти изменения были внесены. Это тоже не все умеют.

Часто бывает, что руководитель и подчиненные очень по-разному понимают круг своих задач и ответственности. Это также может стать причиной конфликтов и неудачного делегирования.

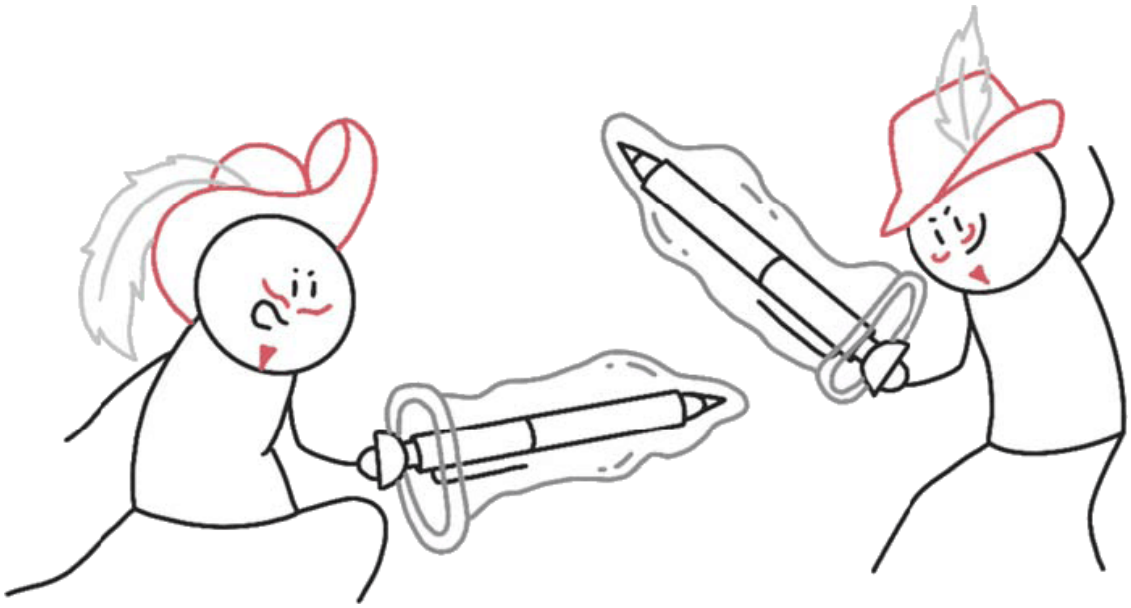
Иногда руководителям не хватает умения замечать командные достижения. Например, вы попросили свою команду самоорганизоваться и проверить проект. Сами убежали заниматься великими делами – скажем, спасать другой проект. А потом вы предъявляете претензии к команде: почему на проекте есть баги, почему не все проверено. И вот вы уже снова в роли спасателя проекта: героически до часу ночи тестируете, а потом с гордой миной заявляете, что только благодаря вам все работает, как надо. А команда – так, мимо проходила.

Сама процедура делегирования зависит не только от того, как поставлена задача, но и от уровня подчиненного. Отправляя человека по пути, думай, кто по этому пути пойдет.

1.8. Правила письменной контрацепции. Как ставить задачи, не убив друг друга

Программисты – как джинны. Осторожнее в своих желаниях, они могут исполниться.

Как письменно формулировать свои требования к программистам? С одной стороны, навык несложный, но именно из-за деталей чаще всего можно получить на проекте долгоиграющий геморрой, а потом ходить и удивляться, почему все складывается наперекосяк.



Итак, солнечное утро. Вы сидите за чашкой кофе. Лениво просматриваете свой проект. И находите в нем какой-нибудь баг. Рука тянется открыть отдельное окно браузера, открыть таск-менеджер, например Jira. Авторизоваться там, найти нужный проект, поставить задачу с описанием бага в грамотных формулировках, приложить скриншоты, описать, как этот баг воспроизвести, назначить ответственных, запланировать себе контроль и так далее. Вот если так – то вы святой человек. Ведь интуитивно в этот момент хочется сказать: «Опять эти упыри накосячили, я что им – ***бета-тестер*** что ли?!» Ну, и устроить этим уродам веселую жизнь. И не дай бог они начнут сопротивляться.

Почему так? Почему даже мелкий баг на проекте может приводить в бешенство и заказчиков, и менеджеров проектов? Причина тому – транзакционные издержки.

Мы интуитивно чувствуем, что на решение мелкой проблемы придется затратить довольно много энергии. На мелкие огрехи часто проще махнуть рукой, чем доделывать те последние 10 % работы, которые занимают оставшиеся 50 % времени. Возможность быстро поставить задачу, прилагая минимум усилий мозга и телодвижений, напрямую влияет на качество проекта и добрые отношения внутри рабочей группы. Хотя и двояким образом.

Минимум транзакционных издержек – это когда программист сидит рядом, под боком, и вы силой голоса призываете его, тычете пальцем в монитор и говорите: «Опять ничерта не работает, пойди разберись!» Нормально ли это? Отчасти.

Совет

Большая часть крупных релизов перед дедлайном и перед выпуском проходят через фазу стресса. И здесь возможно все – вплоть до трехэтажных матов и жестких конфликтов. В целом, это даже полезно. Главное, чтобы такое

не вошло в традицию. День-два в таком ритме можно пережить, дальше – нет. Все должно войти в русло, задачи должны поступать из одного источника, а не сыпаться на бедного-несчастного разработчика с пометкой «СРОЧНО», «КОГДА УЖЕ БУДЕТ» и «НЕМЕДЛЕННО».

Рекомендую посмотреть старое интервью со Стивом Джобсом про драки внутри команды и как они влияют на качество продукта – я его периодически показываю своим ребятам, когда начинаются конфликты.

Вернемся к разработчику, который услышал резко высказанное замечание «баг на проекте» или «ничего не работает». У него всего два варианта действий: либо уйти в оборону, либо – в нападение. Но что адекватного можно ответить на формулировку «Ничего не работает»?! У тебя компьютер что ли не включается?!

Часто из-за нехватки времени на постановку, из-за транзакционных издержек и из-за нежелания думать менеджеры и клиенты склонны ставить неконкретные задачи. На что программисты абсолютно резонно попросят больше конкретики – мол, пиши техзадание. А могут и психануть.

Когда атмосфера накаляется, диалог может складываться как-то так:

– У вас баг на проекте!

– А где именно?!

– ВОТ ТУТ, ЕПТ!!!

Плюс скриншот.

Когда пишешь разработчику матом, либо просто «баг» или даже «косяк» – это воспринимается как личное оскорбление (чуть позже мы посмотрим на когнитивные искажения). Это серьезный плевок в душу разработчика. Пишите простыми, понятными формулировками. Не машите кулаками там, где не должно быть драки. А с надписями КАПСОМ будьте совсем осторожны – это воспринимается так, будто вы на человека ОРЕТЕ. Конечно, только если это не ваш управленческий ход, чтобы загнать кого-то под тумбочку. Только ведь он потом оттуда вылезет! В общем, давайте не капсить. В мире и так слишком много этого дерьмеца.

1.8.1. Меньше насилия, детка!

Вообще, речь на письме выглядит значительно жестче, чем то же самое, произнесенное устно. В подтверждение – известный анекдот про фразы «Папа! Вышли денег!» и «Папа, вышли денег». Если вы имели опыт переписки с клиентами из Европы или США, то наверняка заметили, что те очень часто используют вежливые слова – «пожалуйста», «спасибо» – и часто благодарят. Слова не влияют на суть, но могут сгладить стиль общения. В русских реалиях такое встречается редко. Причина – якобы, вежливая речь может быть воспринята как слабость. На мой взгляд, эта идея глупая, и письменную речь нужно обязательно смягчать.

На тон сообщения влияет буквально все – особенно смайлики и знаки препинания в конце предложений. У Пелевина есть интересная фраза: «Смайлик – это визуальный дезодорант». Поэтому стоит приучить себя к крайне вежливому стилю, не лениться писать «пожалуйста» и «спасибо», ставить смайлики на письме, чтобы сгладить жесткость и сформировать позитивное отношение к себе в коллективе. В ***баг-листах*** или задачах это может быть не всегда уместно, но в переписке, в постановке задач, в комментариях – точно не навредит. Посмотрите, как по-разному будут читаться, вроде бы, хвалебные слова, в зависимости от знака в конце предложения:

Молодец…

Молодец.

Молодец!!!
Молодец:(
Молодец:)
ХОРОШ!

Оформляя баг-листы или письма, всегда задумывайтесь хоть на полсекунды, как это прочитают и с какой интонацией. У нас в студии терпимо относятся, если разработчик использует ненормативную лексику в баг-листах и в комментариях. Для менеджеров и тестировщиков это – табу, для разработчика – иногда можно, если он этим не злоупотребляет, и если эти комментарии не увидят клиенты. А вот за неуважительные высказывания, устные или письменные, по отношению к клиенту уже надо наказывать или, как минимум, – разбираться, разговаривать про добро и зло. Для меня это индикатор «все ли нормально на проекте?!». Потому что, если появляется ругань в сторону клиента или проекта, дальше по инерции ситуация становится все менее и менее управляемой.

Совет

Не допускайте КАПСА, мата и ругани в письме и старайтесь отучать от этого коллег. И не стесняйтесь использовать смайлы, чтобы смягчить жесткость, присущую письменной речи.

Окей, вы такой молодец, написали разработчику вежливое обращение со смайликом и скинули ссылку. Насколько это конкретно? Приложили скриншот? Уже лучше. Можно стрелок к нему еще нарисовать. Уже почти хорошо. Только все равно не спасает. Многие грешат надписями на скриншотах. Но часто конкретики с ними все равно нет.

Я не преувеличиваю ни на грамм. Ставится задача с формулировкой «Косяк на сайте». КАПСОМ. Ставится ссылка на скриншот, где тоже что-то невменяемо написано капсом. Не удивляйтесь, если в комментариях к такой задаче вам также начнут отвечать капсом, нервно и грубо. Тикет-система тут будет не виновата – так накуролесить можно и в Jira, и в Trello, и в чем угодно еще.

Надписи на скриншотах не возбраняются, но они должны быть конкретными.

1.8.2. Найди меня потом, попробуй!

Вместо «Вот тут проблема» можно написать: «В тексте ссылки на отзыв – абракадабра». Гордый собой менеджер или заказчик считает, что на этом его миссия по формулированию проблемы – заметьте, четкому формулированию – закончена. Мол, вот баг, вот скрин, описание – смотри на скрине. Но так не пойдет: если таких багов двадцать (или двести), баг-лист превращается в мешанину, и как что искать в таком случае, неясно.

Сейчас за кадром оставим вопрос, почему у вас заказчик нашел баги, и их так много, но акцент не на этом. Это могут быть и формулировки заданий – разницы нет.

Если в таком виде с вами работает ваш клиент, и у вас не получается договориться о нормальных формулировках (ему тоже не до формулирования и четкости и проще вам заплатить побольше денег, чтобы вы сами все фиксировали с его слов), тогда проблем нет. Фиксируйте все сами, сами переводите с клиентского на программистский. Берите за это деньги. Это воспринимается абсолютно нормально в 80 % случаев и ценится (если вы не сильно косячите). Вы экономите клиенту время – и это тоже часть менеджерской работы. **К исполнителям формулировки должны попадать такие, чтобы задачу можно было найти поиском.** Очевидная вещь, но приходится и про такое рассказывать.

Формулировки стоит писать так, чтобы по ним было легко найти, где случился и как проявился баг. Подробно описывайте, что заставило баг вылезти. И поменьше оценочных формулировок: что это за «абракадабра в тексте ссылки»? В том же Битриксе, например, по умол-

чанию не генерируются ссылки с человеко-понятными названиями, любой программист вам скажет про это. И даже не попытается понять вашу проблему. Мол, в коробочной версии это работает именно так, чего вы от меня хотите?! Это нормальное поведение программистов – и нельзя их за него осуждать.

1.8.3. Проблема или требование?

Бывают формулировки, из которых непонятно: это проблема или требование? Пример – «ссылка фиолетовая». И что? Она должна такой быть или ошибка в том, что она фиолетовая, а должна быть серой? Дело в том, что непонятно, что именно вы хотите, чтобы с этим фактом сделали и как именно его поправили.

1.8.4. Мне кажется…

Другая частая ошибка менеджеров – формулировки задач в виде пожеланий. С оговорками «мне кажется», «я думаю», «а как на ваш вкус» и так далее. Опять же, если такие формулировки приходят от клиента – нормально, если менеджер их уточняет и конкретизирует. Слово «кажется» – это из словаря терминов-паразитов. В баг-листах, в формулировках задач прилагательные и местоимения – это слова, которые можно трактовать двояко. «Каждый», «любой», «все», «больше», «меньше» и так далее. Это признак нечеткой постановки. Как минимум – перепроверьте, можно ли такие слова перевести в четкие цифры или конкретизировать (не всегда, но можно).

1.8.5. Поплачь о нем… В другом месте!

Еще одна дичь – эмоциональность (или даже скрытая пассивная агрессия) в постановках, чатах или рабочих артефактах. Ниже – пара примеров из практики: правильно-неправильно и пара примеров на «подумать», как могло бы быть правильно, а главное – чтобы поставить себя на место того человека, который получил задачу или обратную связь о своей работе в таком виде:

Неправильно	Правильно
Имею подозрения, что плавающая шапка не по макету, если таковой имеется. Если не имеется, то я утверждаю, что надо переделывать.	Плавающая шапка. Сверить с макетом.
А шо с заголовками?	Сделать заголовок по макету.
Карточки товаров не по макету. Почему?	Сделать карточки товаров по макету.
Идей нет. Предлагайте. ФуллХД выглядит стремно. Нарушена высота по сравнению с макетом.	<ваш вариант и чувства в этот момент?>
Мне плохо ((((Можно я не буду тестировать дальше адаптив, а просто скажу, что ВЕЗДЕ НАДО ПЕРЕДЕЛЫВАТЬ? У меня на нервной почве живот заболел уже. Я же ненавижу баги. Переживаю всем сердцем (((((	<ваш вариант и чувства в этот момент?>

Представьте, что программист переживает расставание с подружкой, пришел с плохим настроением. Открывает текстовый редактор. Создает файл VsePloho.php. В нем – ***класс*** PolniyPipec. А в нем метод KakViVseDostali (vi, vse). На другой день настроение наладилось, помирились, новый файл уже SuperPuperMegaKorzina.php и все в таком роде. И это все идет на ревью службе безопасности. Думаю, там разговор будет короткий.

1.8.6. Принцип пирамиды

Организуйте постановки по принципу «пирамиды». Заголовок задачи должен быть такой, чтобы была понятна суть и не приходилось много читать. Детали помещайте в описание.

1.8.7. Приоритеты

В баг-листах, помимо описания проблемы, ее места на сайте и способов воспроизвести баг, также указывается **приоритет**. В разных компаниях системы приоритетов обычно складываются исторически. У нас она – вот такая:

▶ 0 – критические баги (падает сервер или не работает оплата), нулевой приоритет ставим в случае, если тестировщик из-за этого бага не может дальше продолжать проверку проекта.

▶ 1 – либо что-то забыли реализовать, либо что-то критичное по юзабилити.

▶ 2, 3 – менее критичные вещи.

▶ 4 – ошибки в текстах или текстовые правки.
▶ 8 – это «хотелки», некритичные баги.

Замечу, что это именно наши приоритеты по разработке. Например, у рейтингового агентства «Тэглайн» приоритеты другие: для них ошибка в продающем тексте может быть более приоритетна, чем криво работающая форма подачи какой-нибудь анкеты. Ну, ради бога.

Мы намеренно пропускаем 5, 6, 7 – если такие приоритеты понадобятся на конкретном проекте, мы их просто придумаем, не ломая основную систему.

Соответственно, если поставить нашей задаче приоритет, четко показать программисту, что это – «хотелка», он, скорее всего, это сделает, не задавая лишних вопросов. Кстати, приоритеты – очень полезная вещь, поскольку мы можем управлять качеством продукта. Например, закрываем все под 4-й приоритет включительно и запускаемся. Почему бы и нет, если вдруг скорость запуска нам приоритетнее полировки проекта.

1.8.8. Перфекционисты должны гореть в аду. Но это не точно

Есть клиенты-перфекционисты, которые считают, что нельзя выпускать проект с незакрытыми багами. Увы, в наше время это утопия. Любой софт выпускается с какой-то долей некритичных ошибок. С какой именно – это вопрос дискуссионный. Бывает, что пользователи используются как бета-тестеры. Скорость на запуске (или Time To Market, TTM – время от начала разработки идеи до ее конечной реализации) важнее. Понять это можно, посмотрев на обновления приложений, которые прилетают к вам в телефон. Большая часть обновлений в описании содержит две строчки: «Увеличена стабильность, улучшена производительность». На русский язык переводится как: «Мы поправили пару багов». Даже у Apple ошибки, которые известны годами, – это норма.

Исключения – софт, отвечающий за критические элементы инфраструктуры: ядерные реакторы, системы управления двигателями и так далее. Но и там не все гладко, можете почитать про компанию «Тойота», у которой 11 тысяч глобальных переменных и 82 тысячи нарушений в коде. В общем, ошибки в коде будут всегда, как только проект перерастает уровень песочницы.

1.8.9. Горшочек, не вари

Было ли у вас такое, что вы моете посуду, а вам все время подкидывают новую? Опа, а тут еще и сковородка! Это откровенно бесит. Я считаю, что программисты должны видеть конечный набор багов и задач. Поэтому, если тикетов много, стоит организовать работу мелкими итерациями (спринтами). И даже если параллельно с исправлением багов идет процесс тестирования, то новая пачка багов должна попасть к программистам, только когда они отработают и закроют уже выданный им объем. Это не всегда возможно и не везде применимо, но стоит к этому стремиться, потому что так в итоге спокойнее.

1.8.10. Подведем итоги. Правила грамотной постановки задач разработчикам

1. Маты и КАПС. Вам – нельзя. Разработчикам – иногда можно.

2. Место. Указывайте конкретное место, где возникла ошибка.

3. Поископригодность. Пишите так, чтобы запись легко искалась по ключевых словам.

4. Пирамида. Используйте принцип пирамиды. Важное – в начало. Детали – в кончало.

5. Скриншоты – важное дополнение. Подкрепите текстовое описание скриншотом.

6. Решение, а не мнение. Формулируйте не только проблему, но и ожидаемое решение – четко и однозначно, а не в виде абстрактных пожеланий и мнений.

7. Не впадайте в истерику. Пишите по делу, без эмоций.

8. Приоритизируйте. Расставляйте приоритеты и организуйте по большим баг-листам работу микро-спринтами.

9. Закончите уже! Не подкидывайте посуду!

Вот тогда вы как менеджер будете сильно нравиться программистам, и в мире будет больше добра, радости и взаимопонимания. Но это не точно:)

1.9. Семь простых истин из авиации о делегировании и контроле

В авиацию я попал не через авиацию, а почти случайно. Просто повезло столкнуться с правильными людьми. С тех пор прошло несколько лет. Кроме самолета освоил вертолет, посчастливилось полетать по Алтаю, освоить строгий спортивный самолет Су-29, 3-ю лигу по пилотажу, поучаствовать в паре авиашоу. Летаю меньше, чем хотелось бы, и до сих пор чувствую себя дилетантом. Постоянно узнаю новое. Этот опыт меняет картину мира в управленческой работе над digital-проектами.

1. Все, что можно – проверяй сам

Как правило, самолеты должны обслуживать специально обученные техники. Их у нас мало.

Да, есть специальные чек-листы проверки самолета перед вылетом. Обойди самолет, все проверь, все пошатай, бла-бла-бла… Но есть и фактор разгильдяйства. Ребята рассказывали про забытые техником в самолете пассатижи. Рули переклинило. В тот раз обошлось, летчик чудом посадил машину. В авиации, если ты хочешь жить и летать долго – ты просто обязан проверять все сам.

Делаешь проект, управляешь проектом – да, у тебя есть команда, которая вроде как все смотрит, видит, знает, тестирует. Но глуп и беспечен тот руководитель, который только и делает, что верит всему на слово. Не проверит проект лично, на себе. Не сложит своего мнения о происходящем.

2. Зрелищно – не всегда круто

Из опыта участия и наблюдения за авиашоу – публику больше всего заводят проходы на предельно-малых высотах с грохотом и дымами (восторг!). Технически сложные фигуры и обратный (красный) пилотаж, когда от перегрузок лезут глаза на лоб – любят, но не ценят пропорционально тем усилиям, с которыми даются эти фигуры. Например, штопорные бочки – крайне противные фигуры, съедающие все силы. Но они не настолько вау-эффектные. Профессионал оценит. Публика – нет.

Я видел в проектах, когда море сил тратится на иллюстрации (в том числе – 3D), а по итогу – ну да, картинка и картинка. И наоборот, когда стоковое видео или фото в банальном слайдере вызывает реакцию: «Вау! Какой дизайн!»

3. Мелочь ничего не прощает большому

В авиации мелочей не бывает. Никаких. Нигде. Любая мелочь может тебя угробить.

Незакрытые лючки в полете или незакрытый фонарь, недотянутая гайка, и дальше уже ситуация развивается быстро, и, как правило, хреново.

Как-то после зимы я выполнял тренировочный полет. Ремни на пассажирском месте были благоразумно пристегнуты. Все пять раз проверено. Выполняю фигуру «вертикаль» (самолет летит вертикально вверх, теряет скорость, вверху переворачивается и летит вертикально вниз, быстро набирая огромную скорость). На выходе из вертикали ручка управления вдруг становится невероятно тугой, самолет еле слушается.

Ориентируюсь: поджопник с пассажирского кресла заблокировал дублирующую ручку управления. Как позже оказалось – за зиму высох клей. Благо, в RV-7 пассажир и пилот сидят бок о бок, ситуацию удалось мгновенно исправить.

Ты можешь сделать круто на своем проекте глобальные вещи. Но мелочи – это то, что может погубить твой проект. Обращай внимание на каждую деталь, какой бы мелкой и незначительной она ни казалась. Мелочи – это то, что может стать большой проблемой, если вовремя на них не отреагировать.

4. Тщательно выбирай, кого брать на борт

Я очень избирательно отношусь к пассажирам-покатушникам на борту и очень тщательно их инструктирую. Вряд ли возьму незнакомого или показавшегося хоть в чем-то неадекватным человека.

Был случай, когда на взлете испугавшийся (или от эйфории) пассажир попытался перехватить у меня управление. Благо в RV-7 он сидел рядом, и можно было успокоить пассажира. В самолетах типа Extra, Як-52 или Су-29, где кабины разделены и посадка друг-за-другом – такой возможности нет.

Ребята рассказывали про ситуации, когда испуганная девушка-пассажир крепко зажимала сильными ногами ручку управления самолетом и на призывы: «Расслабься. Расслабь ноги. Ноги расслабь! РАЗДВИНЬ НОГИ $%^^…!!!» – реагировала слабо.

Кто у тебя будет на проекте, и как он себя поведет – от этого во многом зависит будущее твоего проекта. В идеале, у тебя должна быть возможность менять членов команды, особенно если ты головой отвечаешь за результат.

5. Вся ответственность, один черт, на тебе

Да, есть куча всяких контролирующих органов, транспортной прокуратуры, проверок, центров сертификации, подач планов, диспетчеров, техников, метеорологов, и всякого такого. Но как только ты – КВС (командир воздушного судна) и принимаешь решение о взлете – ВСЯ (ВСЯ!) ответственность на тебе. Отмазки не канают. Никакие.

Ты управляешь проектом (или компанией) и позволяешь себе оправдания, типа «ой, я тут всего лишь не посмотрел/не подумал», или любишь переваливать ответственность на внешние обстоятельства, сложного заказчика или рукожопых исполнителей? Сорян, тогда рано тебе еще этим заниматься.

Брать на себя всю ответственность – это, кстати, прикольное чувство, попробуй.

6. Летать больше – безопаснее, чем летать мало

Чем больше ты летаешь – тем больше будет разных ситуаций.

Вот недавно знакомые летчики в районе Телецкого озера попали в облако саранчи. Одно зеленое насекомое попало в ПВД – приемник воздушного давления, который обеспечивает определение скорости относительно воздуха. Итог – отказал указатель скорости.

Скорость – наша жизнь. В горах – особенно интересно. Но ничего, сориентировались. Дошли до ближайшего аэродрома, выдерживая скорость по GPS, а изменение маршрута объяснили «санитарной необходимостью» (хорошая формулировка, емкая!).

Лучше летать много, чем 1–2 раза в год. Пожалуй, пару раз в год – это самое опасное. Либо не летай, либо летай постоянно. Да, вероятность попасть в веселую ситуацию – выше, но и опыт здесь – бесценный помощник.

То же и про управление. Занимаясь чем-то профессионально – нельзя делать это время от времени.

7. Ищи наставников и оставляй следы

Так уж получилось, что Су-29, который пригнали из-за границы, два года пылился в ангаре. Самолет сложный, суровый. Летать на нем стало некому, однако с учителями везло. Познакомился с Михаилом Переверзевым, мастером спорта России международного класса, многократным призером чемпионатов России, Европы и мира… Короче, крутым парнем.

Сколько было скепсиса у него в глазах, когда я обратился к нему с просьбой о тренировках: «На таких самолетах только мастера спорта раньше летать могли, и то не все… Это не учебная машина. Эх, времена!» Тем не менее, желание вновь полетать на пилотажнике пересилило, и Михаил согласился. Каково же было наше с ним удивление, когда на третьем часу тренировок мы поняли, что я готов к самостоятельному вылету. На четвертом часу полетел сам, в сложных метеоусловиях. Самолет очень живой, и я даже на 10 % не освоил его потенциал.

Помогли увидеть ошибки недавно установленные дымы. Когда за тобой дымный след – сразу видно косяки. В кабине газовая камера, жарко, пот разъедает глаза, а солнышко слепит, но ничего: назвался пилотом – терпи и тренируйся.

Учить и тренировать вас никому неинтересно. И уж точно – никто не обязан вами заниматься. Это время, нервы, и в конце концов – риск. Если вдруг вас учат чему-то на работе и в рабочее время (управлению проектами, дизайну, программированию, soft skills), и вам это идет на пользу – относитесь к этому с благодарностью. А не как к само собой разумеющейся херне. В жизни очень сложно найти хороших учителей и уж тем более заинтересовать/убедить их заниматься вами. Большая удача.

1.10. Для тренировки

Итак, мы посмотрели картину мира digital-проекта, убедились, что для нас там полно работы и посмотрели, как эту работу делегировать (не поубивав друг друга и с возрастающими шансами на успех).

1. Какие еще объекты вы бы добавили в картину мира digital-проекта? Что вы с ними делаете? Как они связаны друг с другом?

2. Понаблюдайте в течение недели на какие операции и сколько вы тратите времени и энергии. Нравится ли вам это делать, или это рутина? Вы обязаны ее делать, или можно ее кому-то делегировать? Составьте план делегирования рутины. Приводите его в действие.

3. В течение недели попробуйте делать постановки четко по SMART, а письменные постановки с использованием принципа «пирамиды». Проверяйте их через сервис glvrd.ru.

Литература

https://www.youtube.com/watch?v=bo4i525EszY

Андрей Курпатов, «Красная Таблетка».
Максим Ильяхов, «Пиши, сокращай».
Максим Ильяхов, «Новые правила деловой переписки».
Старое интервью Стива Джобса о командной работе (ссылка спрятана в QR-коде слева).

Пояснения

Тикет-система – система онлайн-постановки и контроля задач.

Тикеты – собственно, задачи.

Бэклог – инструмент Скрама, по-простому – это список всех хотелок проекта, отсортированный в порядке приоритетов.

NDA – Non-disclosure agreement, соглашение о неразглашении: юридический договор, между сторонами с ограничением доступа третьим лицам к каким-либо данным и/или результату работ.

Софт – программное обеспечение, или ПО.

Репозиторий – место, где хранится код и вся информация о его изменениях. Репозитории могут находиться на компьютере разработчика, на компьютерах его коллег и на удаленном сервере.

Гайдлайн – руководство по правильному использованию визуальных элементов на сайте.

Тест-план – документ, где описывается весь объем работ по тестированию проекта: как, что и когда проверять. В идеале, тест-план пишет опытный тестировщик, а выполнить тест-план способна обезьянка с мышкой.

API – описание способов, которыми один сервис или программа может взаимодействовать с другим сервисом, программой или сайтом.

Спринт – временной интервал, обычно в 2–4 недели, в течение которого scrum-команда выполняет заданный объем работы.

Рисерч – (en: research) исследование: поиск и сбор необходимой для выполнения задачи информации, проба новой технологии и так далее.

Ретроспектива – собрание команды проекта, чтобы проанализировать прошедший спринт и создать план улучшений для следующего спринта.

Деплой – развертывание кода сайта на сервере.

Боевой сервер – обычно весь программный код пишется на тестовом сервере в студии, а после переносится на боевой, рабочий, сервер заказчика.

Артефакт – результат какого-то процесса: в разработке – файлы исходного кода или файлы данных, в управлении – поставленные в тикет-системе задачи.

Пересаживание обезьян – когда подчиненные свешивают своих обезьян, свои проблемы на своего босса, а тот покорно их принимает, потому что «без вас никак не разобраться» или «вам нужно подписать», или «давайте подумаем, что можно сделать» (см. «Одноминутный менеджер и обезьяны», Бланшар К., Онкен-мл. У., Берроуз Х.).

Стендап – короткая планерка менеджера с командой проекта на 10–15 минут, где обсуждаются три главных вопроса: что было сделано вчера, что будет делаться сегодня и какие есть проблемы.

Декомпозиция – разбивка большого и неповоротливого проекта по аккуратным и понятным блокам. Об этом подробнее поговорим в главе 5, посвященной декомпозированию.

Скрипт – это последовательность действий, описанных с помощью скриптового языка программирования.

Фреймворк или, по-русски, каркас – программное обеспечение, которое облегчает разработку. С ним код пишется быстрее, а в дальнейшем его проще поддерживать.

Стэк – в разработке: набор инструментов для создания проектов, который включает языки программирования, фреймворки, системы управления базами данных, системы управления контентом (CMS) и прочие используемые технологии.

Бета-тестер – почти настоящий пользователь, который интенсивно использует почти готовую версию продукта/сайта/приложения, чтобы выявить максимальное число ошибок для их последующего устранения перед окончательным выходом (релизом).

Баг – ошибка на сайте, в сервисе или в приложении.

Баг-лист – список всех найденных тестировщиком багов, который используется разработчиками для того, чтобы эти баги пофиксить (отловить и починить).

Класс – шаблон кода, по которому создается какой-то объект.

Код-ревью – процесс ревизии кода одного программиста – другим, более опытным. Цель: улучшить качество кода и обеспечить соблюдение принятых в команде стандартов.

Часть 2
Требовательность: как развивать ее у себя и своих сотрудников

2.1. Сколько программистов нужно, чтобы выкопать яму?

Давайте решим такую задачу (гротескная, но так будет веселее).

Допустим, у вас есть сотрудник, и вам по каким-то причинам нужно, чтобы он выкопал яму. Все необходимые параметры, в рамках разумного, придумайте самостоятельно.

Пока не читайте дальше, потратьте минутку для того, чтобы сформулировать (лучше вслух), как вы будете такую задачу ставить. Я подожду.

Сформулировали? Если вы использовали, например, смарт-подход и умеете развешивать контрольные точки, у вас могло получиться что-то типа: «Вася, нужно выкопать яму 30×30×30 см для захоронения офисного котика. А то он сдох, и нам тут скоро всем будет очень душно. Вон в том углу забора. Сегодня до обеда. Лопату возьми в сарае, вот тебе ключ. Справишься? А есть шанс, что не справишься?»

Теперь усугубим. Допустим, вы руководитель диджитал-проектов, а Вася – программист. Вы произносите похожую мантру (мол, выкопай яму, очень надо). Он смотрит на вас честными глазами. И говорит: «Не-хо-чу!» Ваши действия?

Первая естественная реакция: «Ну что за бред! Почему я должен уговаривать программиста заниматься непрограммистскими делами?» Потом идут попытки:

▶ надавить на Васю трудовыми обязанностями (облом, Вася знает, что его дело – код писать, а не ямы копать);

▶ уболтать на личном уровне: либо по-дружески, либо девушкиными ходами, строя глазки (облом, Вася – интроверт, идеалист и женатик);

▶ подкупить премией или даже напугать штрафами (не прошло, у программистов все нормально с зарплатами, а с угрозами штрафов идите вы знаете куда? – правильно, в трудовую инспекцию);

▶ или сослаться, что это дебильное задание дало вышестоящее руководство (совсем гнилой ход, особенно с точки зрения морали – фу так делать! – и Вася вас раскусит).

Самые чуткие, внимательные и креативные руководители проектов (а таких попадается 5–7 %) отреагируют на Васино «Не-хо-чу!». Попробуют выяснить, чего хочет Вася, что ему интересно, и связать свое задание с этим интересом. Такая гибкость ума и чуткость воистину достойна уважения.

Так о чем эта задача? И какое отношение она имеет к реальной жизни? Ведь бред же сивой кобылы – просить программистов ямы копать. Тем более, не имея на это внутреннего морального права.

Теперь представьте, что вы приходите к программисту, вам срочно надо кнопку на сайте перекрасить. Из красной в зеленую. Потому что клиент рвет и мечет (мантры, вроде «URGENT! IMPORTANT! ВЫ НЕ КЛИЕНТООРИЕНТИРОВАННЫЕ» и т. д.). Никак свою задачу не мотивируя. Клиент имеет, наверное, право ничего не объяснять за свои деньги, но это не точно.

А программист Вася вместо того, чтобы за две минуты поправить ***CSS-файлик***, да еще и варианты вам показать с экрана в ***Developer Tools***, начинает артачиться: «Не-хо-чу! Дизайн – это дело не программистское! Вот принесете мне отрисованный макет, утвержденный клиен-

том, я и поправлю. А то уже пять раз за день туда-сюда-обратно. Достали, менеджеры, блин, работать спокойно не дают».

И в чем-то есть его правда. Но для вас он ведет себя как говнюк. А отфутболить неопытного руководителя проектов фразами «задача не моя» или «предоставьте мне то, это, и еще вон-то, поменяйте кресло, поставьте новый комп и монитор побольше, сделайте нормальные постановки, и тогда я, может быть, соизволю что-то там запрограммировать» – Вася может практически по любому поводу, по любой задаче. С этим все сталкиваются. Мне особенно везло на специалистов 1С на стороне клиентов, на которых смотрят как на священную корову. А если он, ко всему, по совместительству родственник директора, заставить сделать его что-то полезное для проекта, да еще и в срок – тот еще квест.

Ключевых вопроса тут два:

1. Где та грань, когда вы требуете от программиста какого-то бреда (вроде выкапывания ямы), а где он просто говнится, чтобы вы отстали?

2. На что опираться руководителю проекта, чтобы добиться нужного ему результата?

Начнем со второго вопроса. Требовательности.

2.2. Требовательность

Требовательность – базовое качество руководителя, в том числе – и руководителя digital-проектов. Это умение настоять на своем, отстоять интересы компании, команды, проекта и свои, если на эти интересы покушается вселенная.

В идеале управление digital-проектом сводится к расстановке приоритетов: это делаем в первую очередь, это – во вторую. Команда весело разбирает тикеты и дружно бежит их выполнять. Желательно автономно: сами подумали, сами быстро и качественно сделали. Вас же на всякую фигню не отвлекают. А только радуют результатами.

Размечтался. Гораздо чаще можно встретить сопротивление требованиям, затупы, а иногда – скрытый или даже явный саботаж.

Требовательность плотно связана с понятиями «власть» и «эксплуатация». Про источники власти много любопытного у Макиавелли, Ницше и других великих умов мира сего. Власть божественная или государственная, бла-бла-бла… – нам так круто не надо. Для управления digital-проектами интересна власть, которая устанавливается внутри команды проекта и разделяет людей на тех, кто командует, и тех, кто подчиняется.

Требовательность – токсична, и в больших количествах – яд (вплоть до психосоматики). Давят сроки, обстоятельства, клиенты, объем параллельных проектов. И малейшее сопротивление требованиям или уточняющий вопрос команды исполнителей вызывает у перегруженного руководителя раздражение. Оставаться при этом еще и в позитивном настроении – сложно.

Я знаю целый класс руководителей уровня «директор по маркетингу», бегающих с конференции на конференцию, с выставки на выставку (в промежутках – в отпуск), забегающих в офис поработать на пару дней и тут же сливающихся на следующую выставку – лишь бы не заниматься управлением.

Требовать мягко и тактично они не умеют. На внятное распоряжение способны только в состоянии аффекта, если их сильно разозлить. Надо ли говорить, что такое поведение деструктивно не только для самого руководителя, но и для его рабочей группы?

Приходится учиться требовать. А в книгах по менеджменту тема табуирована. Принято писать о мотивации, командном духе, самоорганизации и бирюзовых компаниях. Вести дискуссии о том, нужен ли вообще менеджер на проекте. Искать всевозможные поводы как можно дальше дистанцироваться от этого геморроя. При этом забывается, что в западных компаниях субординация, бизнес-процессы, требовательность и исполнительность нормально работают по умолчанию. Никто в голос не ругается, не орет, но если пригласили на разговор за закрытыми дверями – все прекрасно понимают, что дело дрянь. Мотивационные плюшки и самоорганизация – это уже тюнинг.

Забавно выглядит, когда руководитель, начитавшись книг по мотивации и самоорганизации или вернувшись с семинара, вдохновенно решает убрать вот эту самую токсичность управления от себя подальше. Прибегает в компанию и с криками «Ага! Мы с завтрашнего дня – бирюзовые!» пытается перевалить «головняк» на команду. А потом расстраивается, что не работает. И почему-то я не удивлен. Делегировать сначала научитесь!

Уверен, если бы в таск-трекерах была кнопка «Долбануть током» – она бы пользовалась популярностью. Пока такой кнопки нет, ею должны уметь становиться вы сами.

Самоорганизующимся командам нужен самоорганизатор. Уметь планировать, требовать, настаивать, при этом соизмеряя необходимую и достаточную силу. Хотя говорить про требовательность не принято: сочтут упырем, эксплуататором и фашистом.

В долгосрочной перспективе для блага проекта, организации и ее обитателей сильная власть и требовательность предпочтительнее хаоса и анархии. Известный афоризм Гарри

Трумэна, 34-го президента США: «Любое действительно эффективное управление на поверку оказывается диктатурой». При этом никто не против, чтобы все были довольны, счастливы и мотивированы. Как такового противоречия здесь нет.

Самоорганизующимся командам нужен самоорганизатор.

2.3. Механизм власти

Винтовка рождает власть.
Мао Цзэдун

Механизм власти до ужаса прост: выполняй мою волю, а иначе наступят некоторые (по умолчанию – хреновые) последствия. В основе лежит **воля** и **угроза** (явная или неявная). С одной стороны, чертовски обидно (от отвращения до бунта, когда понимаешь, как и когда этот механизм применяют лично к тебе). С другой – факт, что все хорошее в этом мире сделано с использованием механизма власти. С третьей – никуда не денешься.

Наша задача – научиться применять этот механизм профессионально.

2.3.1. Поле власти

У Александра Фридмана вычитал интересную аналогию с полем власти: это вроде электричества, к которому подключаются управленческие инструменты типа делегирования, контроля, мотивации и так далее. Если поле власти сконфигурировано правильно – инструменты работают адекватно. Аналогично с электричеством: параметры тока в розетке верные – устройства работают нормально. Если нет – устройства глючат или даже сгорают.

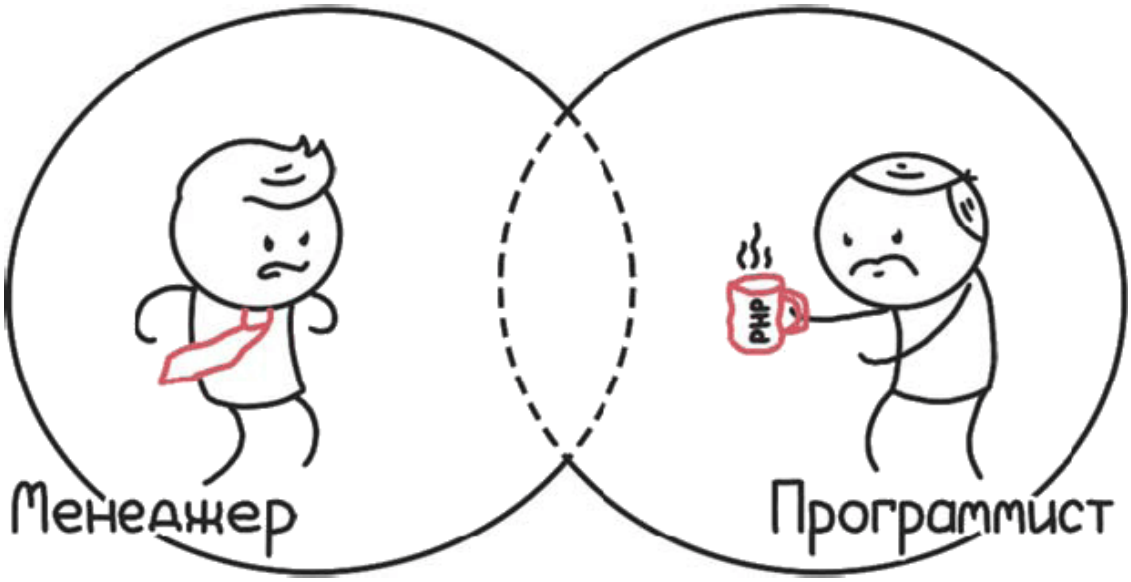
Всю власть в компании возьмем за 100 %, поделенных между сотрудниками. Причем, зачастую это распределение образовывается стихийно. По историческим причинам. И в результате взаимодействия сотрудников власть перераспределяется (у одного уменьшается, у другого увеличивается). Это архиважный тезис – **вы что-то с кем-то делаете (или не делаете), а в результате у вас и у него меняется реальная власть.**

Рекурсия: источник власти – в ее грамотном применении. Поэтому молодым руководителям проектов важно сразу начать применять ту небольшую власть, которая у них есть, для тех дел, для которых она годится.

2.3.2. Самозахват

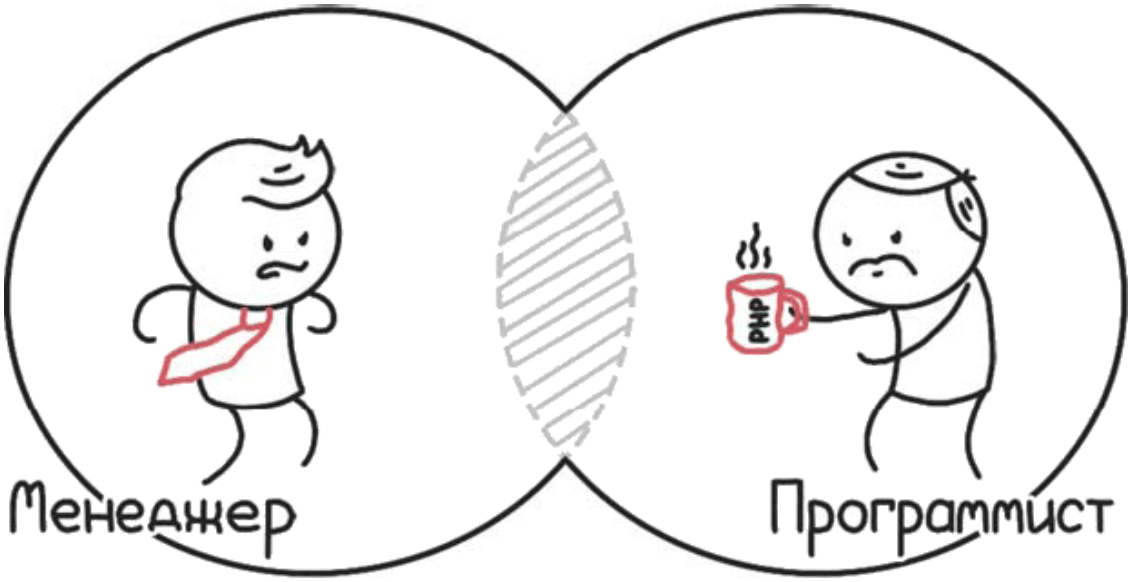
Давайте посмотрим на поле власти на простой модели.

Вот компания состоит всего из 2-х человек – менеджера и программиста – и между ними как-то распределилась власть. У кого власти должно быть больше?



Менеджеры тут же скажут: «У менеджера!», а программисты на это ничего не скажут, но про себя хмыкнут: «Ну-ну, давай, покажи, что ты там из себя за менеджер». И можете не удивляться, если программисты считают менеджеров обслуживающим персоналом. Причем, назойливым, глупым, мешающим работать и думать об ***архитектуре***, ставящим неадекватные задачи и вечно лезущим со всякой несущественной фигней.

Таким образом, в поле власти может возникнуть красная зона, где люди по-разному понимают круг своих обязанностей, свою зону ответственности и свои права.

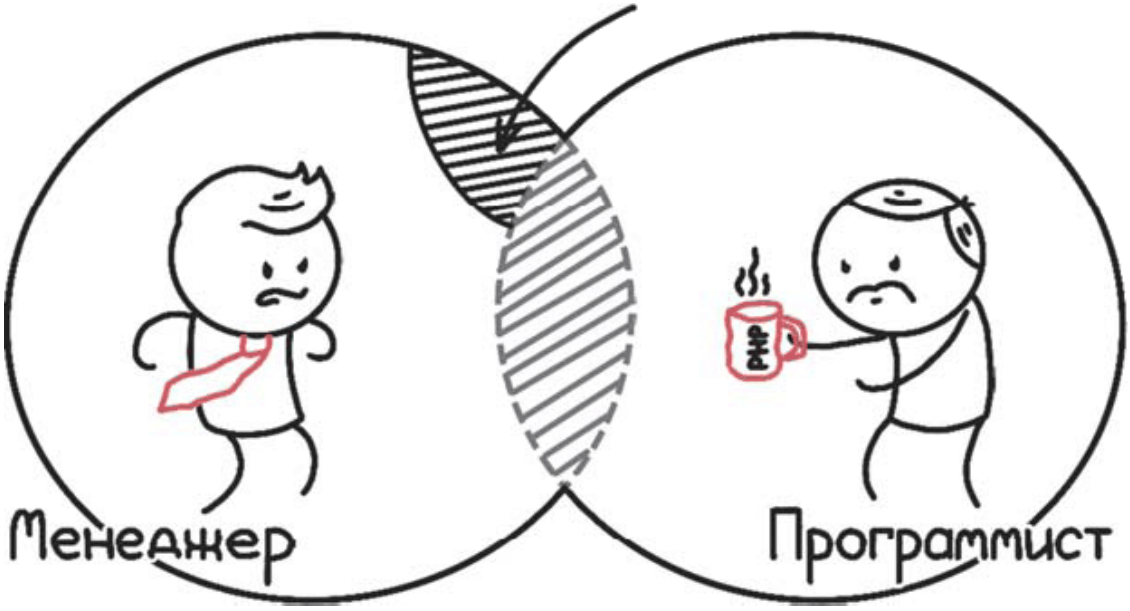


Плохо, если сотрудники в компании не знают, как именно власть распределена; плохо, если знают по-разному. Такая мышиная возня отнимает у компании много энергии. В таких случаях вопрос решается повышением компетенций руководителей (нужны энергичные живчики), введением хороших корпоративных правил и наказаний (вплоть до замены нелояльных сотрудников). То есть, долго и дорого.

Вариантов потерять власть – много. Например, самозахват.

Допустим, у вас есть правило: «Получил задачу – оцени». Программист может сначала, как бы случайно, не оценивать задачи (типа, забыл). Если на это не отреагируют (ну сделал,

и ладно) – не оценивать специально. Если попросят оценки – сказать: «Не знаю, не могу». А потом и вовсе поскандалить и отказаться оценивать задачи: «Вы, менеджеры, ни черта не понимаете в природе программирования, никаких оценок там дать нельзя…» Если менеджер не отреагирует – сформируется право обычая «Васе можно». И все: власть менеджера уменьшилась, а программист отвоевал себе свободу не думать перед началом работы и не отвечать за сроки.



Другой пример: одна моя сотрудница отвечала за организацию пятничного дизайн-баттла. Каждую пятницу ей нужно было придумать какую-то тему, собрать вечером дизайнеров на 1 час, в рабочее время. Дизайнеры ровно час рисовали по макету. Дальше нужно было организовать публичную презентацию (каждый дизайнер защищал свою работу), и можно было проголосовать за какой-то из макетов, выбрав победителя.

С одной стороны, это прокачивало навыки презентации, ускоряло дизайнеров и просто было весело. С другой, за долгое время проведения таких баттлов пошла профанация: часто работы не имели художественной ценности, а выигрывала самая укуренная. И вот в одну из пятниц я узнаю, что баттла не будет. «Как так?» – спрашиваю я. Ответ был таким: «А они отказались сегодня. Говорят, каждую неделю – слишком часто, давайте через неделю хотя бы». Тут много чего было нарушено, но острее всего – самозахват: дизайнеры захватили право решать, будет баттл или нет, а также, чем они будут заниматься в рабочее время по пятницам.

За самозахват бьют, и больно.

Чаще всего менеджеров продавливают, лишая прав:

▶ получать оценку оставшейся работы – от этого и программисты, и дизайнеры будут увиливать под любым соусом: мол, я не гадалка и не знаю, сколько времени это займет;

▶ выбирать инструменты и методы работы – проще говоря, технологии;

▶ выбирать, использовать ли готовые модули или писать с нуля;

▶ оценивать выполненную работу.

2.3.3. Как бы шутка

Допустим, я программист и сделал какую-то противозаконную гадость. Например, отказался оценивать задачи. Наотрез, категорически, навсегда. И менеджер-слабак это покорно принял.

Но, скорее всего, это не за один день произошло. До этого я готовил почву: всерьез обсуждал абсурдность тратить время на оценку задач. И никто меня в этом не остановил.

А до этого я прощупал почву – пошутил про глупость оценки задач, все посмеялись, и никто меня не остановил. Слова, сказанные как бы «в шутку» нужны, чтобы прощупать границу допустимого. Чуть-чуть наступаем на чужую территорию, кончиком носочка, но готовы в любой момент ногу одернуть.

А до шутки я серьезно думал, что хорошо бы было избавиться от оценок, и никто меня в этих мыслях не остановил.

А еще что-то было до того, как я про это подумал.

И чем дальше заходило дело, тем больше усилий нужно было, чтобы меня остановить.

Самозахват (да и любое нарушение) спускать с рук нельзя, но отвечать нужно соразмерно. Шутливый самозахват лучше остановить шуткой. Твердой по содержанию, но мягкой по форме. Например, программист как бы в шутку произносит: «А сдается мне, что оценивать задачи – это лишняя трата времени». И смотрит на вас добрыми глазами. Можно так же в тон, шутливо ответить: «А сдается мне, что не оценивать задачи – это попытка увильнуть от ответственности». Верните ему это же без агрессии, но с улыбкой.

2.3.4. Воля к власти

Кстати, вряд ли программисты сидят и серьезно думают «как бы так у менеджера власть оттяпать» – им это неинтересно (и даже брезгливо). Но вот как сделать, чтобы:

▶ можно было играться с прикольными и новыми технологиями (о которых другие программисты отзываются позитивно – так действует механизм причастности к успешной группе);

▶ работать по тем правилам, которые удобны;

▶ не отвечать за сроки и ошибки (намеренно использовал слово «ошибки», хотя оно не очень корректно – за ошибки наказывать нельзя).

И несильно себя напрягая? Как-то само собой происходит, что менеджеру навязывается воля программиста, а скрытой или явной угрозой будет его ворчание, токсичное поведение или программисто-зависимость. Борьба за власть выглядит как рациональная борьба за свободу.

Вторая частая причина потери власти – отказ от ее применения. В силу лени или страха. Применение власти требует поставить свою шкуру на кон (см. Н. Талеб, «Рискуя собственной шкурой. Скрытая асимметрия повседневной жизни»). Выбирая комфорт, мы выбираем отказ от власти. Или нам не нравится применять власть, потому что мы это делаем плохо. Не умеем, а прокачивать себя – ленимся. В итоге полно менеджеров, которые ни черта не решают. Робкие, пугливые, рукожопые, что-то вечно «передающие» и «выясняющие».

Становиться руководителем пора только тогда, когда у вас появились важные личные цели, а добиться своими руками вы либо не можете, либо это слишком долго. Если со всем этим «управлением» разбираться не хочется, а хочется сидеть в своей норке и ни за что не отвечать – в менеджеры вам рановато. Выбор – либо командовать, либо подчиняться. Третьего не дано.

Кроме самозахвата, лени и страха, менеджер может потерять власть и своими неаккуратными действиями. Есть власть реальная, а есть – декларируемая или номинальная. И если менеджер очень сильно увлекается декларируемой властью, пишет какие-то грозные указивки, письма, задачи и требования, которые никто в итоге не исполняет, – его власть уменьшается.

Например, менеджер может потребовать у программиста каких-то новых отчетов, которых раньше не было. С точки зрения программиста – менеджер залез на его территорию. Что произойдет, если программист эти отчеты писать не будет, а менеджер при этом не будет настаивать? Очевидно, что реальная власть менеджера в этот момент уменьшится, а авторитет программиста в коллективе может увеличиться: какой он крутой, отстоял свои права.

Совет

Если вы только начинаете отстаивать свою власть и авторитет в коллективе – для начала давайте сотрудникам небольшие и простые задания с большим запасом времени. Но добивайтесь 100 %-го выполнения. Не 98 %, а именно 100 %. И если все идет хорошо – постепенно увеличивайте нагрузку и сложность задач.

2.3.5. Функции власти

Цикл Деминга «планируй-делай-проверяй-воздействуй» (PDCA, Plan-Do-Check-Act) – классическая диаграмма с точки зрения организации грамотной работы, о который мы уже упоминали в главе 1. Но если дополнить цикл с точки зрения функций власти, получится следующее:

- ▶ планируй – формируй принципы работы и картины мира подчиненных;
- ▶ делай – доводи правила до сотрудников;
- ▶ проверяй – обеспечивай соблюдение правил;
- ▶ воздействуй – награждай и наказывай.

Правда, есть этому циклу и классическая альтернатива, которая встречается в организациях довольно часто.

Функции власти

Планируй

Планирование не терпит суеты, тут нужно действовать постепенно, настойчиво и последовательно. Мы должны продумать в своей голове порядок, по которому другие люди будут делать наш проект или вообще работать. То есть создать принципы, парадигмы, картину мира, правила и задачи, приоритеты.

Делай

Доводите правила до подчиненных: где-то это уместно письменно, декларативно, где-то – устно, в виде обсуждений.

Декларативный способ – это как публикация закона: документ вышел, остается только его прочитать и исполнять. Метод резко снижает затраты на доведение правил. Минусы – не согласные с этими правилами подчиненные могут подвергать письменные документы неправильной трактовке, оглуплять и дискредитировать. Искажать не суть, а дух. Как-то так его прочитать вслух с хитрющей интонацией, что всем кругом станет смешно. Начните с описания проблемы, из-за которой возникло конкретное правило, объясните причины и приведите примеры. Еще один минус деклараций – для честных подчиненных, в случае возникновения вопросов, их будет некому задать.

Альтернатива – беседовать и доводить правила до каждого сотрудника лично. Затратно по ресурсам, хотя, в случае наставничества дает обучаемому больше суперсилы. Как вариант, используйте видео-формат. У меня большая часть регламентов текстовые, но снабжены видео. Так сложнее исказить дух или трактовать по-своему. Но, если возникают вопросы и нет куратора, то и этого формата может быть недостаточно.

Еще один вариант – доводить регламенты на общих собраниях. Техника хороша тем, что можно и ответить на вопросы, и посмотреть на реакцию сотрудников. В ***Scrum*** это уместно делать на ретроспективах – специальных собраниях, где обсуждаются проблемы проекта, ищутся решения и вырабатываются новые правила.

Минусы: вы работаете с умными людьми, и, когда будете доводить до них регламенты, – встретитесь с сопротивлением, подколками, хитрыми вопросами. Особенно если новые регламенты меняют распределение власти. Готовьтесь к работе с возражениями заранее.

Доведение регламента до пользователей мало чем отличается от процесса продажи. Поэтому умение грамотно продать свое видение будущего, правил работы, регламентов, требований или выполняемых задач – очень важный навык для менеджера и для руководителя проектов.

Проверяй

Наблюдать за ходом работ. Здесь мне нравится понятие «гемба» из японского менеджмента – это то место, где производится ценность. С ним также связан принцип «Тойоты» «Иди и смотри» или, в зависимости от перевода, «Иди сначала в гемба и осмотрись!» Вылезь из своего долбанного онлайна, дойди ножками, и посмотри, как на самом деле люди работают.

Если что-то несложно проверить – проверяйте сами, смотрите сами, вникайте и разбирайтесь сами. Покликайте мышкой проект, а не слепо верьте тестировщикам, что там багов нет. Проверьте со своего телефона. Поиграйте с проектом. Ну и смотрите, как люди делают работу на своих рабочих местах. Наблюдайте сами, как люди работают, а не просто слушайте то, что они об этом говорят.

Воздействуй

По итогам контроля производим корректирующие воздействия. Раздаем поощрения и наказания.

К сожалению, без инвестиции вашей личной энергии, без регулярного воздействия и выполнения этих операций, дело протухает. Кто-то должен крутить это колесо. Вялый, угрю-

мый, депрессивный руководитель убивает команду. И наоборот, позитивный и энергичный способен творить чудеса.

Сама система управления должна периодически пересматриваться: нет ли перегибов, тухляка, можно ли что-то улучшить. Так рождаются самообучающиеся, самоорганизующиеся команды, где роль менеджера – задавать ритм и производить энергию.

2.3.6. Что будет, если нет?

Среди менеджеров-интровертов редко, но встречается паттерн поведения, когда они все общение по проекту переводят в почту или в тикет-систему, а лично с исполнителями никогда не общаются. Причина? Поскольку требовать трудно и токсично, можно столкнуться с сопротивлением – придется отстаивать свою позицию здесь и сейчас, но не у каждого на это хватает «пороху».

Поэтому менеджер открывает свою любимую тикет-систему, пишет там задачки и ждет, когда они будут сделаны. Вообще, тикет-системы – это отличный инструмент, но вот чисто письменный подход работает на очень небольших проектах с низкой долей неопределенности. Чуть проект сложнее, чуть выше неопределенность – без личных обсуждений, без брейнштормов, без споров уже не обойтись.

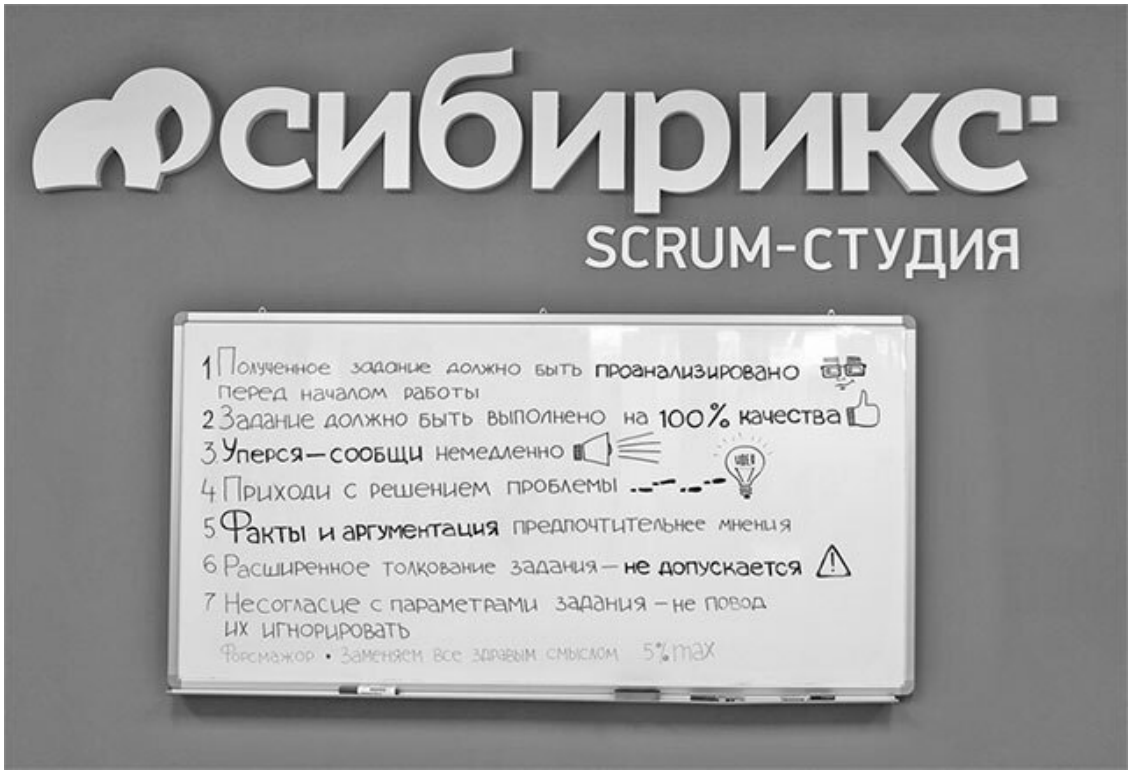
Совет

Отдавая распоряжение на проекте, особенно если оно касается каких-то непопулярных мер, – я рекомендую вспомнить замечательную песенку Семена Слепакова «Что будет, если нет?»

Что будет, если ваше распоряжение или обещание, или угроза не будут выполнены? Как вы будете действовать в этой ситуации? Важно заранее продумать меры, на которые вы лично готовы пойти, чтобы добиться нужного вам результата. Если не уверены, что распоряжение будет выполнено – не надо такие распоряжения отдавать. Вы просто дискредитируете себя, свою власть и свои полномочия.

2.3.7. Центрирующие парадигмы Фридмана

Центрирующие парадигмы – набор принципов, который был предложен Александром Фридманом для снижения управленческих издержек, формирования адекватного поля власти и выработки одинаковых правил игры. У нас они написаны на доске, на самом видном месте. Настоятельно рекомендую прочитать оригинал.



2.3.8. Источники власти: на что опираться в самом начале

Итак. Развитие требовательности – это не самая простая штука на земле. И начинать нужно с личного уровня. С требовательности к самому себе. Учиться не врать себе. Смотреть, где ты ленишься, не дорабатываешь. И заставлять себя доделывать или делать необходимые вещи. Это база.

Дальше нужно постепенно освоить источники власти:

▶ Полномочия или роль. Это то, что вам досталось, когда вас назначили менеджером.

▶ Право наказывать или награждать. Если вам оно дано; если нет – нужно его получить у своего руководства. Алгоритм можно найти в книге Сивожелезова «Сложные переговоры с подчиненными» – держите ее под рукой.

▶ Информация. Если вы знаете больше о проекте, волей-неволей люди к вам прислушиваются, у них формируются ожидания, что вы знаете, как надо действовать, и это дает вам дополнительную власть.

▶ Квалификация. Опыт. И результативность. Если вы крутой в каком-то деле – это тоже наделяет вас определенной властью. Авторитетом. Люди будут вас уважать, так как знают, что с вами можно добиться успеха и нужного им результата.

▶ Регламенты, стандарты отрасли, центрирующие парадигмы и другие нормирующие документы.

▶ Обычаи компании, неписаные правила.

▶ Ну и иррациональные вещи, такие как харизма, возраст, тембр голоса. То, что подсознательно воспринимается как «О, это наверное начальник. Крупная шишка». То есть, многие люди охотнее начнут подчиняться здоровому, брутальному, бородатому мужику, который говорит басом и со всех чего-то требует, чем молодому человеку с писклявым голосом, который прячет глаза за дипломом MBA.

Дальше можно найти того, кто явно слабее вас, и попрактиковаться на нем. Чем-то озадачить, посмотреть, как реагирует. Сможете ли вы справляться с его бунтами и закидонами. Если есть под рукой дети – попробуйте заставить их что-нибудь сделать. Например, съесть кашу, вместо того чтобы играть в айпад. Уверяю, это бесценный опыт, после которого вам будет практически все понятно про управление людьми.

Дальше. Если вы вступили в роль менеджера, и ваш руководитель обладает авторитетом, властью – можете смело поначалу опираться на его авторитет. То есть, поначалу отдавать распоряжения, действуя как бы от его имени: «Иван Иванович попросил, чтобы ты сделал кнопку красной. К какому часу это будет готово, скажи пожалуйста мне, чтобы я его сориентировал?»

Как только все ваши подчиненные к этому привыкнут – все, отдавайте распоряжения без ссылки на вашего руководителя. С опорой на вашу должность. Вы руководитель проектов, поэтому и требуете.

Далее – опора на привычку подчиняться.

И дальше уже выход на иррациональный уровень. Харизма, требовательный вид, голос, взгляд (жесткий или мягкий), интонация. Вы чего-то требуете, потому что так надо. Никому ничего не объясняя.

Вот примерно такие ступеньки. Вообще, развитие требовательности можно вынести для себя в такой личный отдельный проект. Где-то на полгода-год.

2.3.9. Ступеньки требовательности и мастерства

Придется ли вам всегда проявлять агрессию? А вот это не факт. Давайте посмотрим, как это развивается.

Уровень «Слабак»

Допустим, вы не готовы идти на конфликт. Окей. Конфликтов будет мало. Но результат будет за ваш счет. Ваших выходных или овертаймов, например.

Уровень «Злоблин»

Допустим, вы заводитесь с полоборота. Чуть что – сразу готовы идти на конфликт. Тоже не очень хорошо, окружающие будут от вас шарахаться.

Уровень «Боец»

Готов конфликтовать ради дела. Если надо подраться – подерусь. В общем, с этого уровня уже можно становиться менеджером. Хотя не все окружающие будут вам сильно рады. При этом руководство тут осуществляется лично. Лично отдаете распоряжения и лично, фактом своего существования, обеспечиваете, чтобы эти распоряжения выполнялись. Уже неплохо, но трудозатратно.

Уровень «Дипломат»

Умеет не только конфликтовать, но и договариваться. Конфликты редкие, но и готовность идти на них – низкая. Тоже неплохо, но дипломат при прямом столкновении с бойцом, скорее всего, проиграет. Поэтому предпочитает управлять через бюрократию, регламенты. А это не всегда продуктивно.

Уровень «Нормальный руководитель»

Он, в случае чего, – готов идти на конфликт. Поставить негодяев на место. Обеспечить плохим людям проблемы и веселую жизнь. Все про это прекрасно знают. Поэтому на конфликт не нарываются. И конфликты становятся все реже и реже.

Уровень «Хороший руководитель»

Вот это очень интересно. Хочу – иду на конфликт, хочу – не иду. Могу и так, и так и сам выбираю, когда и что применить. При этом все подчиненные довольны и счастливы – ну, потому что адекватное руководство, хорошая управляемость и высокая результативность.

Ну а дальше выход на **уровень иррационального**. Когда все проходит вроде как само собой.

Заметьте, с опытом необходимость идти на конфликт сначала возрастает, а затем спадает.

Через это, наверное, все руководители проходят. И пока они учатся конфликтовать, обязательно будут жертвы. Будут обиженные. Просто по неопытности. Кого-то слишком сильно отменеджерили – и все, теперь он ваш заклятый враг. Экологичнее тренироваться на тех, кого не жалко. Ну и ни в коем случае не на домашних, родных и близких. Ваша требовательность – это для работы. Домашние тут как бы ни при чем.

2.4. Нормальная эксплуатация и мозгоклюйство

Итак. Слабые руководители не готовы проявлять требовательность. Испытывают дискомфорт или моральные муки из-за того, что им приходится заставлять коллег дело делать. Выражаться это может по-разному: от состояния «верните меня обратно с управленческой должности на линейную» до высказывания: «А я не хочу быть объектом эксплуатации и не хочу эксплуатировать!»

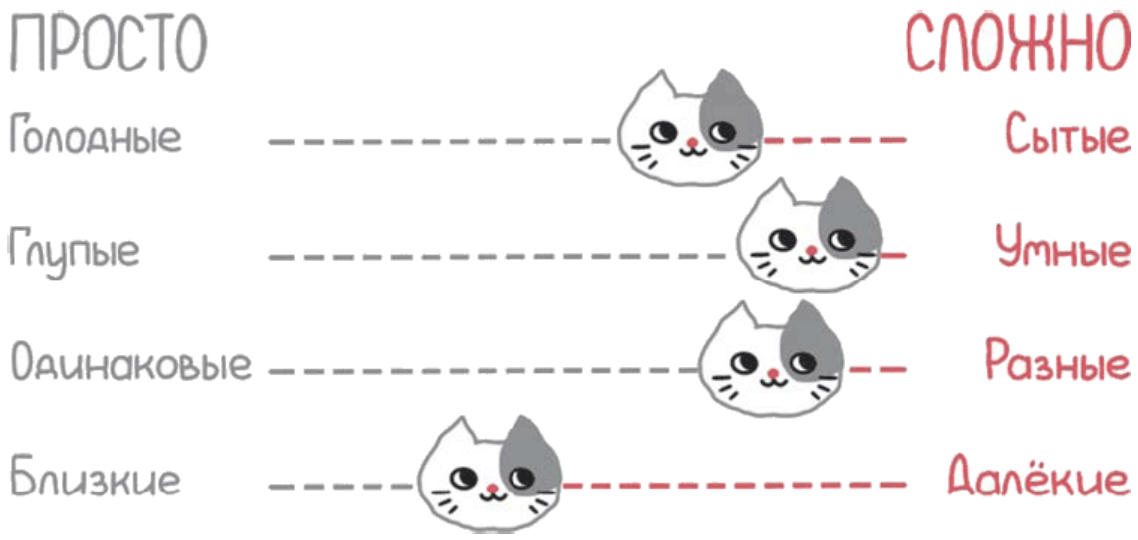
Гадость в том, что вас никто не спрашивает – хотите вы или нет. Вы либо командуете, либо подчиняетесь. Других вариантов не существует.

Мы как руководители заинтересованы, чтобы сотрудник оставался с нами как можно дольше и работал как можно продуктивнее.

Нужно еще и понимать, кем вы управляете. В диджитале это – айтишники: дизайнеры и программисты. А они, как правило, люди сытые. Сытыми управлять сложнее, чем голодными – приходится дополнительно напрягаться, ведь рублем их вряд ли замотивируешь (чем мотивировать – так сразу и не скажешь). Это люди умные, часто – с системным мышлением, и их задурить не получится. А вот с их стороны вы постоянно будете испытывать подколки и попытки проверить вашу власть на прочность. Кроме того, каждый из них – личность, каждый требует индивидуального подхода и по-разному реагирует на одни и те же слова и задачи.

Руководителям digital-проектов вообще выпала доля иметь дело с наиболее трудно управляемой категорией людей: сытыми, умными, творческими индивидуалистами, у которых свое на уме и к которым порой нет физического доступа (удаленка рулит?). Но даже если вы рядом, посмотреть глаза в глаза и поговорить один на один зачастую трудно. Просто потому, что интроверты прячут взгляд и не любят подобного рода разговоров. Короче – сложно.

Задача менеджера – использовать потенциал, интеллект, время и ресурсы этих людей во благо своих проектов, и делать это грамотно. Грамотный руководитель должен понимать интересы сотрудника, его характер, мотивацию, знать его тактико-технические характеристики (хотя бы на уровне ***карт компетенций***), уважать его свободу воли. В общем, знать и учитывать особенности при постановке задач и ведении проектов.



Нормальная эксплуатация

- ▶ своевременное планирование;
- ▶ грамотная настройка: постановка задач и целей;
- ▶ мониторинг и контроль;

- обратная связь;
- четкие границы допустимого (зеленая, желтая и красная зоны);
- системное сервисное обслуживание.

Диапазон допустимых режимов эксплуатации часто зависит от целей организации. Есть разница, когда компания работает ради денег здесь и сейчас, или когда она планирует зарабатывать и в будущем. Есть разница в ценностях компании: важно ли, чтобы сотрудники были счастливы, или этот параметр желателен (иногда и вовсе необязателен). Есть разница и в выпускаемом продукте – он рядовой (как туалетная бумага «Обычная») или им хочется гордиться?

Это те параметры, которые в западной литературе задаются в документах с громкими названиями «миссия» или «ценности компании». Сама идея таких документов – неплохая, но глупые шарлатаны ее окончательно испортили.

2.4.1. Мозгоклюйство

Наверное, каждый хоть раз ощущал на себе разницу, когда к нему отнеслись требовательно, но справедливо, а когда занимались мозгоклюйством? К сожалению, эту разницу не так-то просто формализовать. Если клиент говорит: «Сделайте мне сайт, как у Apple», как понять, это нормальное требование или мозгоклюйство? Вполне возможно, что нормальное, ведь допустимо, если мы в самом начале плохо представляем конечный результат. Ненормально, если мы не готовы тратить и выделять **достаточное количество ресурсов** исполнителю на поиск решения, зато готовы выделять бесконечное число яда для критики результата и личности исполнителя.

Причем, ресурсы – это не только деньги. Это и время, знания, навыки. Это конкретизация задачи, то есть готовность того человека, который ставил задачу, погрузиться в проект, потратить свое время на брейнштормы, обсуждения требований и обратную связь. Когда мы требуем от исполнителя «сделай то, не знаю что» или хотим, смутно понимая результат и не обсуждая конкретику, без экспериментов и права на ошибки, чтобы тот сам догадался, каким должен быть результат и как его достичь, – это уже не требовательность, а мозгоклюйство.

> Мозгоклюйство – это требовательность без предоставления необходимых конкретному исполнителю ресурсов.

2.4.2. Работа в потоке

Естественное состояние человека – расслабленное. Так задумано природой. Экономим энергию или, как говорит Макс Дорофеев, – мыслетопливо. И, если нас никто не напрягает, ничего не требует – мы на расслабоне: сделать что-нибудь не против, но так, чтобы это было интересненько и не слишком напряжно. Мороженку скушать, киношку посмотреть, новую программку потыкать.

Мозг легко заинтересовывается любой фигней, даже работой. Вы когда-нибудь читали за завтраком этикетки, например, от сока? Зачем? Непонятно, но очень интересно. А если еще и на казахском…

Поэтому задача менеджера – создать такую ситуацию или атмосферу, при которой человеку будет проще начать выполнять дело, чем разруливать последствия – «почему не начал?».

Но опять же. Мы управляем умными, творческими людьми. И минут через 15–30 работы они входят в состояние потока. Это когда ты сконцентрирован, мир вокруг как бы перестает существовать, тебя прет и хочется работать, работать и работать. Это супер-продуктивное состояние, результативность выше в десятки раз.

Помните, когда вы до ночи засиживались за делом или за игрой? Время течет незаметно, а все, что вы делаете, – доставляет удовольствие. Усталость не чувствуется, а если чувствуется – это «добрая» усталость, как после пробежки. Вот в этом блаженном состоянии программисты и дизайнеры кайфуют от работы.

Чтобы попасть в поток, нужно минут 15 или больше сосредоточенно заниматься делом (позже рассмотрим технику Помодоро). Из потока легко вылететь – достаточно, чтобы дернули по мелочи. Сосед задал дурацкий вопрос, например. Вы чертыхаетесь, полчаса продуктивной работы летит в корзину, а с ними и хорошее настроение.

Посчитайте, сколько раз вас нужно дернуть за день, чтобы вы задолбались, стали злым и раздражительным, но ничего толком не сделали? Вот поэтому я запрещаю дергать ребят без экстренных причин (по экстренным все же можно, только для начала мысленно сравните выгоду от «дернуть» и полчаса времени в утиль, а также готовьтесь обосновать экстренность) и прошу вырубить всплывающие уведомлялки чатиков.

Менеджеры в таком состоянии пребывают редко – нас слишком часто пытаются поюзать (коллеги, клиенты, чатики, почта). Учитесь закрываться. А для этого нужны такие правила игры, когда менеджер тоже может работать в потоке, и дергать его запрещено. Клиентам это, кстати, очень не нравится: нас приучили к мгновенным ответам. Нам кажется, что то, на чем мы сфокусированы в данный момент, и есть «самое важное», и если на наше «самое важное» не реагируют – мы либо теряем его из фокуса, переключаясь на другое «самое важное», либо раздражаемся. Стоит ли говорить, что через пару часов «самое важное» зачастую превращается в пустяк? Предварительный вывод: чатики – зло для продуктивности (про голосовые сообщения вообще молчу).

Чатики – зло для продуктивности.

Ну да ладно. Допустим, вы попали в состояние потока. Знаете свою цель, вам – хорошо, и хочется работать еще и еще.

И здесь появляется риск ухода в перфекционизм. Человек может начать полировать код. Вылизывать пиксели. Долго размышлять об архитектуре. Или по три дня к ряду подбирать наилучший оттенок фиолетового. Тут экономика не сходится.

Менеджерская задача – держать человека в этой зеленой зоне графика, где ему интереснее сделать, чем не делать, и при этом вовремя забирать результат. То есть, обеспечить и интерес сотрудника к делу, и рентабельность его работы.

Для этого мы должны грамотно планировать его задачи и следить – интересно ли сотруднику, или он заскучал. Давать обратную связь, отдых, обучение. Словом, сервисное обслуживание. Можно ли тут сэкономить? Угу. Кратковременно добьетесь всплеска результативности, но в итоге получите текучку талантливых кадров, жуткую нагрузку на кадровый отдел и хреновую репутацию. А вместе с этим в долгосрочной перспективе – кратное увеличение себестоимости проектов.

2.4.3. Форсаж

Можно ли использовать людей в режиме форсажа? Ответ – да, можно. К форсажу можно прибегать, если этого требует дело, и если результат главнее последствий и выгорания людей. При этом вы должны понимать, что усталость, ошибки, демотивация возрастают. Но для понятной и великой цели, которую ЧЕТКО объяснили, и указали сроки, когда форсаж закончится, форсаж допустим и иногда даже полезен. Этакая встряска для организма.

В digital-компаниях рекомендую пару раз в год проводить ***хакатоны*** или делать внутренние фановые проекты. Но допустимость форсажа – это не повод устраивать в компании вечную «сессию». Никто не призывает эксплуатировать людей на износ. Но, опять же, всякое бывает.

2.5. Когнитивные искажения у дизайнеров и программистов

В человеческой психике есть баги, которые мешают нам объективно воспринимать реальность. Они – причина прокрастинации, желания отморозить уши назло маме и того, что временами люди не понимают друг друга, хотя, вроде бы, не дураки и общаются на одном языке. В психологии причины таких проблем называются когнитивными искажениями.

Как и баги в коде, когнитивные искажения можно исправить. Но для этого нужно понимать, где их искать и какими они бывают.

На странице Википедии в списке когнитивных искажений целых 200 пунктов, и 199 вы, скорее всего, найдете у себя. Рассмотрим те, которые часто встречаются у работников digital-сферы и у людей интеллектуальных и творческих профессий. Зная, как работают когнитивные искажения, вы научитесь понимать, почему подчиненные капризничают, и сможете регулировать их поведение.

2.5.1. Слишком оптимистичные оценки работы

Или, как вариация – пессимистичные оценки, только чтобы от вас отстали. Искажение встречается часто и лечится повторяющимися вопросами. Если человек отвечает не на тот вопрос, что вы задавали, нормальная практика – повторять вопрос несколько раз, пока вы не добьетесь нужного вам ответа.

– Когда будет готово?

– Ну, мне осталось вот это, это и это доделать, и будет готово.

– Отлично, но когда именно будет готово?

– Ну, тут недолго, обычно…

– Так когда будет готово?

– Через 20 минут.

Еще один вариант – техника «А на какой вопрос ты сейчас отвечаешь?!» Если человек не дает конкретики и пускается в объяснения или изложение последовательности своих действий, просто скажите ему напрямую, что он отвечает не на тот вопрос, который вы задали.

– Когда будет готово?

– Ну, мне осталось вот это, это и это доделать…

– Ты сейчас на какой вопрос отвечаешь?

– Будет готово через 20 минут.

Кстати, если задаете несколько вопросов письменно (в одном письме или чате) – скорее всего, вам ответят только на самый удобный. Заведите себе привычку нумеровать вопросы, приучите своих орлов нумеровать ответы. Это несложно.

2.5.2. Генерализация частных случаев

Генерализация частных случаев – когнитивное искажение, из-за которого человек расширяет поставленную задачу. При этом он даже не осознает, что ее можно выполнить проще и быстрее.

Генерализация фиксится варан-менеджментом, который пришел к нам из дикой природы. Существует байка, что варан кусает свою жертву, отравляя ее. Яд действует не сразу, поэтому после укуса варан ходит за жертвой и ждет, когда та умрет. И уже тогда обедает.

Как и варан, менеджер кусает человека задачей. А после садится рядом и внимательно смотрит в его мониторчик. Он может даже не понимать, что делает подчиненный – это необязательно. Главное – быть рядом и следить. И – о чудо! – через непродолжительное время исполнитель поймет, что таки существует более простое и быстрое решение, чем то, которое он придумал сначала.

> **Варан-менеджмент** иногда обоснован, в отличие от двух других животных стилей управления: чайки и дятла.
>
> **Чайка-менеджмент** – это когда менеджер прилетел, наорал, нагадил и улетел. Такой стиль поведения никогда не решает проблем, зато всегда понятно, кто виноват, а у подчиненных много активной бурной, но чаще всего нерезультативной деятельности.

Дятел-долбоклюй – это ну очень эффективный менеджер: вместо того, чтобы поставить задачку в трекер и назначить конкретный срок, он ходит и каждые пять минут спрашивает сотрудника: «Уже готово?! А когда будет? А сейчас готово? А теперь?!»

Грамотное делегирование – не самая простая задача на земле, но это не повод вести себя как животное:)

2.5.3. Это невозможно!

Среди когнитивных искажений можно выделить целую группу, которые мешают взяться за задачу. И все они сводятся к тому, что человек, чуть поковырявшись в постановках, упирается и заявляет: «Это невозможно!»

У такой реакции несколько причин:

▶ исполнителю просто лень;

▶ сработал эффект сопротивления, и человек хочет доказать, что он сам волен решать, что делать, а что нет;

▶ задача противоречит его чувству прекрасного.

В любом случае менеджеру нельзя верить в то, что задача невыполнима. Наоборот – он должен выяснить, какие ресурсы нужны исполнителю, чтобы ее сделать. Снова и снова долбить подчиненного вопросом: «Расскажи, пожалуйста, что тебе потребуется, чтобы сделать эту задачу?»

Опыт показывает, что со временем подчиненный понимает, что ему не верят, да и действительно задача не так уж и невыполнима… И в конце концов он ее выполняет.

В будущем этот случай можно будет использовать как тыкательную палку в аналогичных ситуациях. На категоричное «Это невозможно!» у вас будет кейс, когда сотрудник тоже считал задачу невыполнимой, а потом сделал ее за 20 минут.

2.5.4. Критика как личное оскорбление

Это искажение часто встречается у личностей творческих, особенно невыспавшихся и в плохом настроении. Чтобы вырулить ситуацию в конструктив и никого не обидеть, нужно действовать по следующему алгоритму:

1) Отделить хорошее и плохое.

2) В работе, пусть ее и нужно переделать, все равно были какие-то положительные моменты – вспомните их и похвалите исполнителя.

3) Объясните, в чем ошибка. Возможно, человек не знает, как эту ошибку исправить, и именно поэтому сопротивляется и бунтует.

4) Инициируйте повторное обсуждение или брейншторм по уже пройденным вопросам.

5) Попросите помощи коллег или дайте дополнительный ресурс – например, время на рисерч.

6) Настаивайте на переделке плохого.

7) Закрепите пройденный урок: выясните причину, почему искажение активировалось, и запомните, как вы его пофиксили.

К сожалению, многие начинающие менеджеры в этот момент либо боятся доводить дело до конца, либо ленятся. Но если следовать этому алгоритму – можно выйти на конструктив даже быстрее, чем вы ожидали.

2.5.5. Проклятие знания

Вот пример когнитивного искажения, которое называется «проклятие знания». Это когда человек более информированный не может рассмотреть проблему с точки зрения человека, который знает меньше. Отсюда столько непонятых гениев.

Проклятие знания устраняется самодрессировкой. Нужно отлавливать свое нелогичное поведение и наступать себе на хвост. Пытаться выстроить конструктивный диалог, даже если очень не хочется. А то всю жизнь можно прожить, думая, что все вокруг глупые, а ты один в пальто стоишь красивый. А на деле окажется, что все совсем наоборот.

2.5.6. Эффект генерации

Не все когнитивные искажения – причина проблем и непонимания. Бывают и полезные. Например, «эффект генерации». Благодаря этому искажению человек лучше запоминает информацию, когда воспроизводит ее сам, а не воспринимает извне.

Это искажение используется в авиации. Например, когда диспетчер общается с пилотом. Если диспетчер на земле передал какую-то информацию, пилот в самолете должен повторить все параметры, которые были заданы. Менеджерам тоже полезно использовать этот прием, чтобы проверять, правильно ли подчиненные поняли задачу. Называется – «выписать квитанцию».

2.5.7. Слепое пятно

Еще одно когнитивное искажение: слепое пятно в отношении когнитивных искажений. Человек, который в курсе, что искажения существуют, отрицает их влияние на свое поведение. А последствия списывает на обстоятельства и на глупость окружающих. И, соответственно, не делает ничего, чтобы пофиксить свои когнитивные искажения.

Так что, если подчиненный говорит вам, что вы начитались книг по психологии, а задачу «Ну правда невозможно никак выполнить» – может быть, пора отдать задачу другому исполнителю.

2.6. Для тренировки

Выработка требовательности и уверенности, если вы не обладаете этими качествами, должна стать для вас приоритетным проектом.

Начинайте с выдачи небольших заданий, но добивайтесь по ним 100 % выполнения. Сначала ставьте задачи тем исполнителям, которых вы психологически сильнее. Опирайтесь на власть своего руководителя, правила компании, парадигмы, стандарты отрасли, регламенты. Затем – на здравый смысл и эстетику. Постепенно переходите к иррациональному уровню.

В течение недели следите за собой и за своими коллегами: подмечайте, какие неконкретные слова и выражения и в каких ситуациях они применяют. Когда вам, вроде бы, что-то ответили, но легче от этого не стало. Иными словами, отлавливайте отмазки: «не знаю», «я подумаю», «скоро» и так далее. Сюда же – три импотентских глагола: попробую, постараюсь, попытаюсь. «Попробую» с русского на менеджерский язык переводится – «отстань, я этого не смогу».

Составьте свой словарик таких слов и натренируйте слух, чтобы, как только эти слова прозвучат в вашем присутствии, у вас рефлекторно появлялось желание ~~дать леща~~ попросить человека, как минимум, конкретизировать эти слова до сроков, цифр и конкретных артефактов.

И постарайтесь не перегнуть палку – уж больно хлопотно исправлять управленческие ошибки.

Литература

▶ Нассим Николас Талеб, «Рискуя собственной шкурой. Скрытая асимметрия повседневной жизни».

▶ Александр Фридман, «Вы или хаос. Профессиональное планирование для регулярного менеджмента».

▶ Александр Фридман, «Управление мышлением подчиненных: центрирующие парадигмы», аудиокнига.

▶ Павел Сивожелезов, «Сложные переговоры с подчиненными».

▶ Максим Дорофеев, «Джедайские техники» и «Путь джедая».

Пояснения

Рекурсия – ситуация, когда объект является частью самого себя. Если у вас украли кредитную карту, позвоните по номеру на обороте кредитной карты.

Архитектура – это общее устройство кода сайта или приложения: используемые библиотеки, модули, классы, функции и их отношения. Иными словами, общее описание, внутри которого разработчик пишет код. Вопрос об идеальной архитектуре философский, и многие разработчики стремятся к совершенству архитектуры как йоги к Нирване (зачастую, зря тратя время).

Большинство современных промышленных систем реализованы с использованием паттерна проектирования приложения **MVC (Model View Controller)** или его производных. Не все, это не догма, есть и альтернативные варианты, но этот подход чаще всего применяется. Model-View-Controller или MVC, или «Модель-Представление-Контроллер» – предполагает разделение данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента:

Модель (Model) предоставляет данные, как правило, лежащие на сервере, например, в базе. И эти данные со временем могут как-то модифицироваться. Например, на сервере хранится информация о товаре.

Представление (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели. То есть, как этот товар показывается на странице.

Контроллер (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.

Чтобы было понятнее, пример. В базе данных у вас лежит товар. Его можно вывести и на страницу с карточкой товара, и на страницу списка товара и в корзине пользователя. Физически в базе это будет один и тот же товар, но отображаться он будет по-разному. За хранение данных о товаре отвечает модель. За то, как этот товар можно показать пользователю – представление, или View, их может быть несколько. Ну и есть некие операции, которые можно выполнять с товаром: положить в корзину, удалить из корзины, удалить из базы, добавить остатки на складе и так далее. За них отвечает контроллер.

Понимание паттерна проектирования может быть важно, когда вы оцениваете проект: расписываете его по экранам и операциям с каждой сущностью данных.

Scrum – передовой фреймворк (платформа), созданный в 90-е специально для разработки, передачи и поддержки сложных продуктов. Сейчас используется и в других сферах. Суть: весь объем работы делится на короткие этапы в 2–4 недели (спринты), в рамках которых выполняется конкретный перечень задач из бэклога (списка всех задач, упорядоченных по приоритетности). Подробнее о Скраме мы поговорим в главе 3.

Карта компетенций – таблица со списком и уровнем необходимых навыков по каждому сотруднику. Благодаря ей можно оптимально распределять людей по командам, следить за прогрессом каждого из них и давать задачи на прокачку недостающих навыков. Подробнее о картах компетенций говорим в главе 7.11. Пример карты компетенций ищите в Приложении 1.

Хакатон – форум для разработчиков, во время которого специалисты из разных областей разработки программного обеспечения сообща решают какую-либо проблему на время.

CSS-файл – каскадная таблица стилей, которая применяется для оформления веб-страниц. С помощью CSS-файла задается цвет, шрифт, положения отдельных блоков на странице.

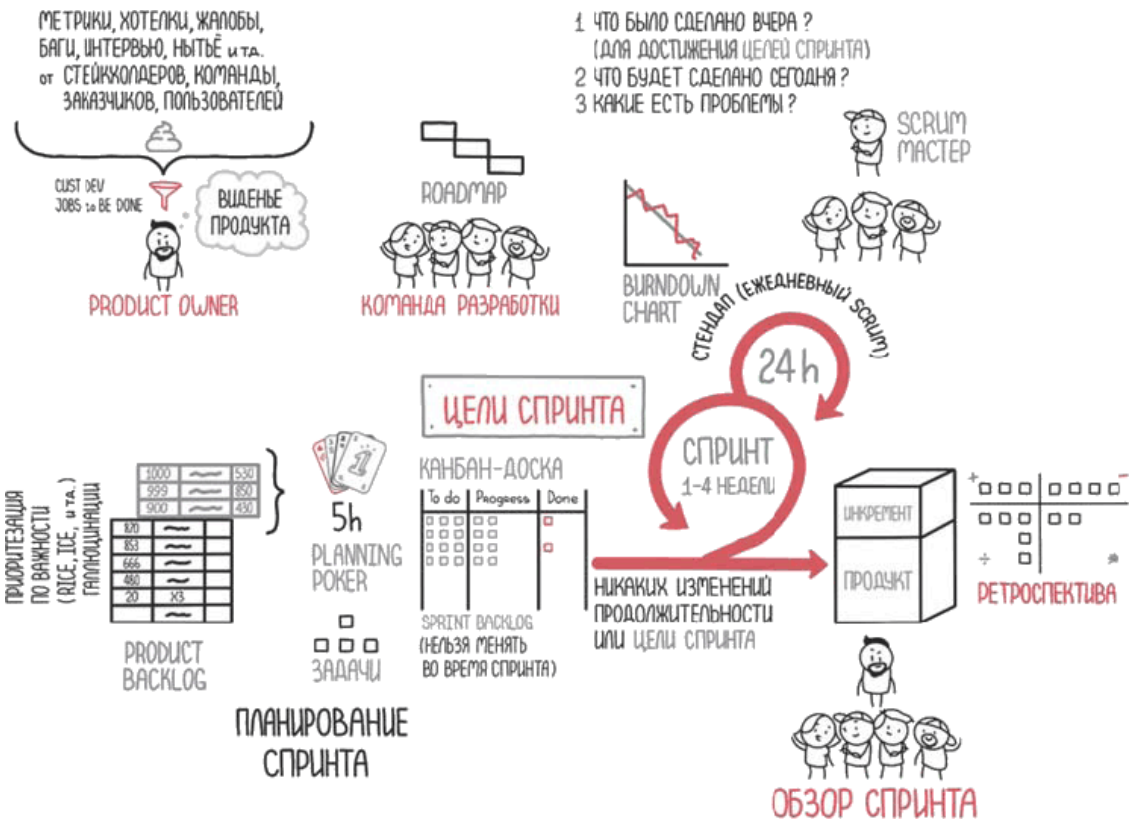
Developer Tools – панель инструментов веб-разработчика. Обычно встроены по умолчанию в современные браузеры, чтобы можно было легко просматривать исходный код сайта.

Часть 3
Пятиминутный scrum

Настало время кратко посмотреть на Scrum – фреймворк разработки проектов. «Фреймворк» означает, что в чистом виде, по книге, у вас это вряд ли заработает. Ладно-ладно, заработает, но не с первого раза. Его нужно будет настраивать и сращивать с вашими текущими процессами.

Фреймворк уже зрелый: по нему много литературы, курсов и сертификаций. Поэтому здесь мы поговорим о нем коротко – только в той мере, в которой он нам нужен для повседневной, практической работы. Фреймворк не без косяков, с известной зоной действия и ограничениями. Но пока принципиально лучше ничего не придумали. И Scrum хорошо работает с удаленными командами.

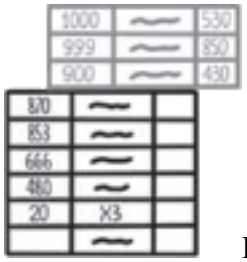
3.1. Схема scrum. Артефакты. Роли. Процедуры



Итак, продукт выпускается поэтапно. Сначала минимальная версия. Затем постепенно наращиваем функциональность.

Каждый цикл разработки называется **Спринтом**. Рекомендованная продолжительность спринтов – от 1 до 4 недель, подбирается экспериментально. Нужен ритм «рывок-отдых». В итоге каждого спринта должна получиться новая версия продукта с приростом функций – **инкрементом**.

Product Owner (Владелец Продукта) отвечает башкой за стратегию и приоритеты разработки. К нему поступают «хотелки» пользователей, метрики, жалобы, баги, идеи стейкхолдеров. Коллективная ответственность в расстановке приоритетов не работает. Но рутину вроде сбора обратной связи можно делегировать.



В Scrum-е нет нудного толстого технического задания, которое выполняется от корки до корки. Нет попытки предусмотреть все и сразу. Вместо ТЗ хотелки складываются в специальную табличку – **Product Backlog** – и сортируются по приоритетам с точки зрения бизнеса. Можно по ходу работы менять, выкидывать и добавлять функции.

Примерно так это выглядит:

	A	B	C	D
1	**Priority**	**Feature**	**Description**	**Est, Days**
2	900	Главная страница	Слайдер. Меню разделов "Партнеры", "О компании", "Каталог товаров".	10
3	800	Страница "О компании"	Текстовая страница с инфографикой	2
4	700	Каталог товаров	Список товаров с картинками и основными характеристиками	5
5	600	Карточка товара	Название, текстовое описание, цена, фотогалерея	2
6	500	Добавление товара в корзину	Список товаров в корзине. Возможност ьменять количество	1
7	450	Страница корзины	С карточки товара	2
8	400	Оплата через банковские карты	Платежная система Yandex.Money	2
9	300	Форма контактов	Карта + форма обратной связи	2
10	220	Раздел "Партнёры"	Список партнёров с логотипами	1
11	200	Поиск по каталогу		1
12	100	Расчет стоимости доставки	Почтой России или слоупоками	2
13	10	Оплата через расчетный счет	Выставление счета на заказанный товар	2
14		Регистрация, авторизация, восстановление пароля		3
15		Личный кабинет		

Простой бэклог в Google Docs

Минимальный состав бэклога: хотелка и приоритет. Можно добавлять свои поля. Например, описание и предварительная трудоемкость. Чаще всего, в условных единицах – Story Point. Берем за 1 балл самую простую хотелку, все остальные оцениваем относительно нее. Бэклоги удобно хранить в Google-таблицах. Можно дать доступ команде и синхронизировать с таск-трекером типа Jira.

Приоритет – чем больше число, тем выше. При ручной расстановке приоритетов удобно делать между ними зазоры (100, 200, 500). Так проще будет вставлять хотелку между двумя другими. Одинаковых приоритетов, по классике, быть не должно.

Чем ниже спускаемся по бэклогу – тем меньше необходима степень проработки. Пустая трата времени формализовывать то, до чего годами не дойдут руки. И наоборот, хотелки вверху списка должны быть ясные, с высокой степенью готовности к работе.

Все заинтересованные лица могут добавлять что-то в бэклог. Но только Product Owner определяет приоритеты. Для этого нужно периодически просматривать бэклог, выкидывать оттуда мусор, обновлять приоритеты и улучшать формулировки.

Есть специальные методики приоритизации, снижающие субъективное мнение (галлюцинации) Product Owner о важности той или иной хотелки. Подробнее о техниках поговорим в главе 4.

RoadMap – предварительный календарный график выпуска релизов. Этого нет в Scrum, но без него картинка проекта теряется.

Версия	Июль 2030	Июль 2031	Июль 2032	Июль 2033	Июль 2034	Июль 2035	Июль 2036	Июль 2037	Июль 2038	Июль 2039	Июль 2040
САЙТ (v1) Главная страница, каталог, контакты (v1)											
Личный кабинет (v2) Личный кабинет для физических лиц											
САЙТ (v2) Корзина, заказ, интеграция с ERP											
Мобильное приложение (v1)											
Личный кабинет (v2) Дилеры, оптовики, геймификация											

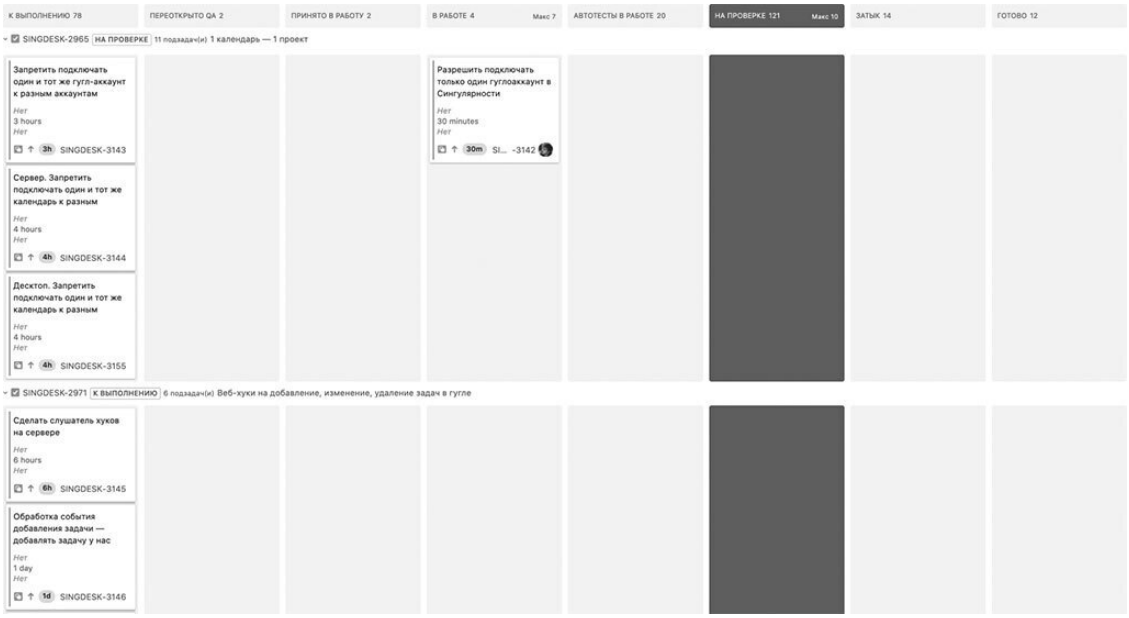
Как выглядит Roadmap

Команда разработки. По умолчанию имеются в виду программисты. Команда забирает верхнюю часть бэклога в работу, дербанит каждую хотелку на технические тикеты и оценивает. Мы используем Planning Poker, о нем чуточку позже. Так формируется бэклог спринта (**Sprint Backlog**) – то, что команда считает реальным сделать за следующий Спринт.

Задачи, попавшие в Sprint Backlog, менять нельзя. В отличие от задач из Product Backlog, в котором можно менять все что угодно до тех пор, пока команда разработки не заберет и не запланирует верхние хотелки.

Обычно **Sprint Backlog** у нас попадает на ***канбан-доску*** в тикет-системе. Это привычный многим инструмент, где есть карточки с задачами и колонки, соответствующие статусам работы. Как минимум это Plan, In Progress, Check, Done. Чертовски напоминает цикл Деминга.

Можно придумывать свои колонки, добавлять критерии готовности, чек-листы, ограничения одновременно выполняемой работы (WIP, work in progress) – тут все гибко.



Канбан-доска спринта в Jira

Команда обычно работает с этой доской: каждый разработчик забирает интересный ему тикет. Выбирает сам, а не назначает начальство. Пишет код, перемещает карточки по доске, трекает время и так далее. Доска может быть как физической, так и электронной.

Команда – такой мини-спецназ ~~в тылу врага~~. Компактная: рекомендуется не более семи человек. Кросс-функциональная: есть все компетенции, чтобы сделать проект. Самоорганизующаяся (упс!), самообучаемая и ответственная. Большие проекты дробятся на компоненты и распределяются по своим командам.

Ежедневно, в одно и то же время и в одном и том же месте, команда собирается на **Стендап (Daily Scrum),** где каждый по очереди отвечает на три вопроса:

1. Что было сделано вчера (для достижения целей спринта)?
2. Что будет сделано сегодня?
3. Какие есть проблемы?

Если вдруг случится амнезия, и вы забудете все из этой книги – хотя бы эти три вопроса оставьте при себе. Помогают вернуть управляемость в самом гиблом деле вроде затянувшейся стройки.

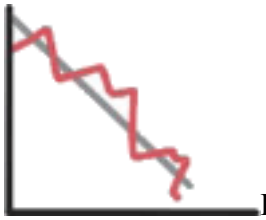
Вопросы задает специально обученный человек, **Scrum Master**. Он следит за соблюдением процедур Scrum. Например, может отбиваться от Product Owner, возжелавшего поменять состав спринта прямо по ходу работы. Scrum Master также помогает команде оперативно решить возникающие проблемы.

Как проходят стендапы в «Сибирикс»

Важно модерировать такие встречи, не давать уходить в технические дебри и не говниться. Если есть вопросы, требующие детального разбора, их нужно выносить на отдельные встречи.

Тоскливо, когда на вопрос: «Что было сделано вчера», вам отвечают что-то вроде: «Я размышлял об архитектуре», перечисляют номера тикетов или закидывают техническими деталями. Пальцем, дружок, на экране покажи, что там нового напрограммировалось!

На стендапах удобно повесить перед глазами сформулированную кратко **цель спринта** – помогает сфокусировать команду не на микро-тикетах, а на цели.



И **Burndown Chart** (диаграмму сгорания), по которой видно, успеваем ли мы, или все пошло «через плохо». Эти и другие метрики полезны, но нос держим по ветру и чутко реагируем на то, что говорит команда. Управлять только через метрики не получится.

В конце спринта команда проводит **демонстрацию** – показывает Product Owner и всем заинтересованным прирост функциональности. Тут же получает обратную связь. Иногда жест-

кую – это, кстати, бодрит. И идет на **ретроспективу**. На ретроспективе все вспоминают, что было хорошего, что плохого, и думают, как улучшить рабочий процесс. По итогам ретроспективы команда может насоздавать себе тикетов на самоулучшение.

Намылить, смыть, повторить.

3.2. Внедряем!

Мы внедряли у себя Scrum в начале 2000-х, когда это еще не было мейнстримом, а про Agile не говорил Греф из телевизора. Наломали, конечно, дров, но быстро все исправили. Первое, что внедрили – итерации и стендапы с тремя каверзными вопросами. Скорость и управляемость сначала выросли, а потом упали. Команды начали либо грустить, либо бунтовать. В воздухе витало «ща долбанет». И долбануло бы!

А что не так? Не было ретроспектив. У команд копились вопросы «а зачем все это» и ощущение, что с них выдаивают результат. Этакая вечная сессия.

Начали проводить ретроспективы, объяснять, как устроен Scrum. Реально собирать обратную связь от команд. Решать их проблемы. Дали почитать литературу. Обучили Scrum-мастеров внутри команд и сказали, что теперь менеджеру можно говорить слово из трех букв («Цыц!»), если он нахальничает. И начало получаться.

Примерный план действий при внедрении:

1. Дать команде почитать про Scrum, какие есть роли и артефакты, и какие плюсы он дает.

2. Нулевая ретроспектива. Собрать команду. Обсудить ее проблемы. Посмотреть, сможет ли Scrum улучшить ситуацию, обсудить, как именно. Рассказать еще раз весь процесс. Ответить на вопросы. Распределить роли.

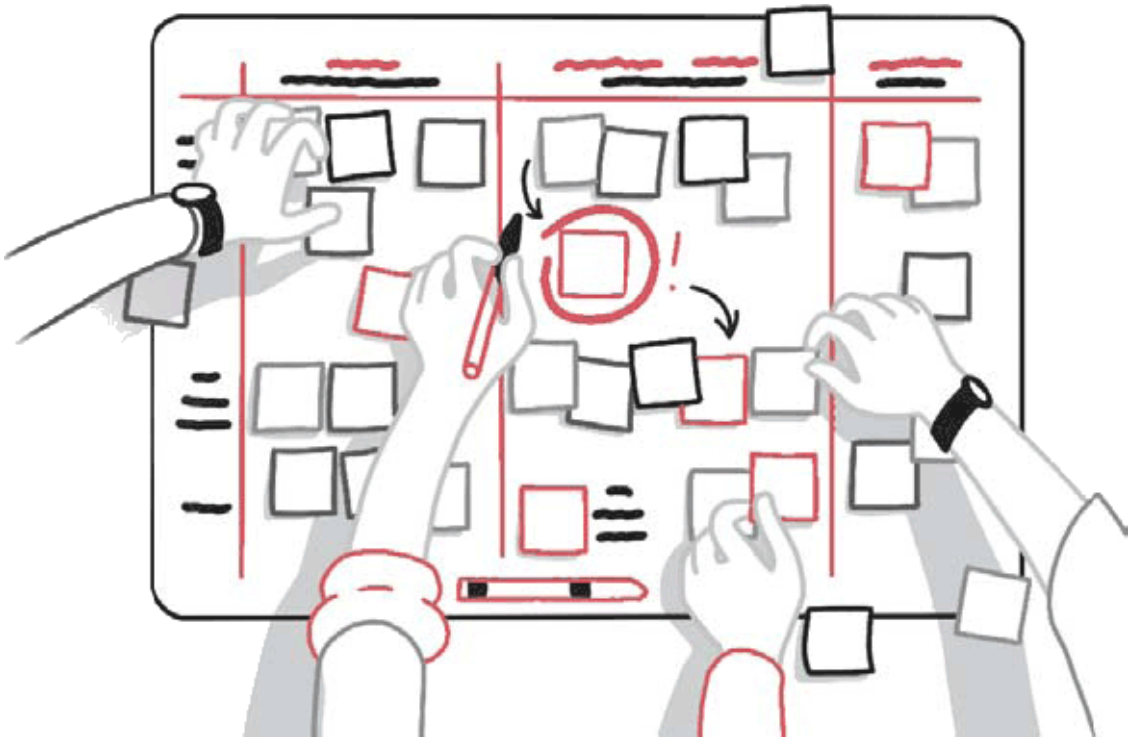
3. Организовать итеративную работу (спринты) и стендапы. Запретить менять постановки внутри спринтов. Для начала выбрать недельные циклы. Потом – откорректировать.

4. Приучиться расставлять приоритет, создать бэклог и регулярно его пересматривать. За основу можно взять ТЗ (если оно было) или смету, или просто собрать все хотелки проекта из головы в один файлик.

5. Внедрить регулярные ретроспективы и корректировать процесс. Чем более зрелая команда, чем меньше проблем в проекте, тем меньше времени нужно на ретроспективы (если не накопилось фундаментальных косяков, которые непонятно даже с какой стороны решать).

Я бы вообще начал внедрение Scrum-а с ретроспектив. Раз в неделю-две собирал бы команду, обсуждал проблемы и решал их, улучшая рабочий процесс. Глядишь, через месяц весь Scrum нормально заработает.

3.3. Подготовка и планирование спринта



Вы примерно понимаете, что именно хотелось бы успеть за спринт. Не факт, что по итогам планирования вы заберете в спринт все эти хотелки. Но предварительное понимание все-таки должно быть.

Очень важно, чтобы к моменту планирования были проработаны постановки задач, проведена промежуточная аналитика, готов весь необходимый дизайн, были подключены все необходимые доступы, ключи к API или описания протоколов и всякое такое.

Бывают задачи или темы, к которым вообще непонятно, с какой стороны подходить. Нужен рисерч. Тут две стратегии: кому-то из команды потратить пару часов на исследование перед спринтом, либо взять задачу на рисерч в спринт, а реализацию делать уже в следующем спринте.

Если проект только-только начинается, там есть инфраструктурные задачи, без которых команда не сможет стартовать: подготовка репозитория, развертка фреймворка, создание базы данных, проектирование архитектуры. Все это стоит наладить немного заранее, за день-два до планирования первого спринта.

Подготовку лучше поручить самому опытному бойцу. Волей-неволей ему придется погрузиться в проект и продумать детали. У него могут возникнуть правильные вопросы, на которые лучше сразу найти ответы. Например, он найдет нестыковку интерфейсов и API. Если это обнаружится в ходе спринта – скорее всего, места для маневров будет мало.

В итоге один из ваших орлов проведет разведку боем, будет знать карту местности. И сможет отвечать на вопросы команды во время планирования. Это сильно помогает и ускоряет планирование.

Опытным путем мы пришли к тому, что нужно дать команде за час-два до планирования прочитать постановки и сформулировать вопросы. Ребята делать этого не любят и читают уклончиво. Логика такая: зачем напрягаться-читать, если на планинге все равно голосом проговорим. Мозг экономит энергию. Но если этого не проделать – планирование спринта может

превратиться в многочасовой склочный базар. Приходится заставлять читать и думать. Как тренер в спортзале заставляет сделать пару дополнительных приседаний.

В итоге команда приходит на планирование с предварительной картиной спринта в голове. Осталось сверить картинки, декомпозировать хотелки на тикеты, оценить и понять, сколько задач мы реально успеем за спринт.

Нужно стремиться так декомпозировать задачи, чтобы их легко можно было посмотреть визуально. Это не всегда возможно, особенно в проектах с обилием математики или интеграций. Но для веба и мобильных приложений, в основном, получается.

Размер задач зависит от опыта команды. Мне нравится, когда каждый день можно глазами посмотреть, что поменялось в проекте. Постарайтесь делать декомпозицию, чтобы задачи было просто проверить, а трудоемкость была от 1 до 8 часов (если вы оцениваете в часах, а не в Story Points). Опять же, не всегда получается, да и опытная команда будет сопротивляться такой мелкой разбивке. Опытным нужны крупные куски. Но для молодых и дерзких управляемость сильно возрастает.

Сугубо технические задачи, типа «Удали поле id из таблицы Users» – дрянь. Формулировки лучше делать на уровне фич: «Форма обратной связи» или «Отправка письма о восстановлении пароля».

Долгий планинг, больше полутора часов, говорит о плохой предварительной подготовке, плохих формулировках или вовремя не проведенном рисерче. Или что вы откусили слишком жирный кусок. Пифии предсказали провальный спринт и геморрой.

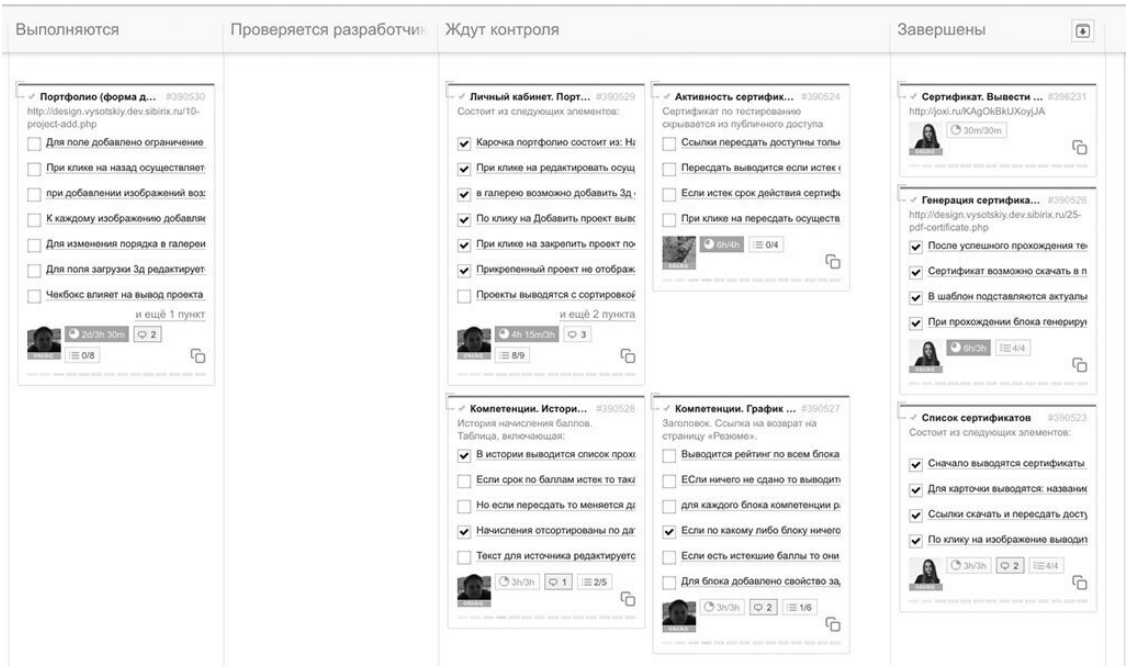
Не старайтесь забить время спринта под завязку. Например, в двухнедельный спринт команды из трех человек засунуть задач на 240 часов. Лучше иметь небольшую подстраховку на код-ревью, ***рефакторинг***, отладку и ***закон Мерфи***. Про него известно, что он точно случится. Сколько взять подстраховки – зависит от опыта команды и задач. Нужно подбирать эмпирически. Начните с 20 %: не 240 часов, а 190. Через пару спринтов нащупаете свою реальную производительность.

Такая подстраховка не нужна, если вы работаете в ***Story Point***. Она зашита внутри оценки.

Кроме разработчиков, на планирование я приглашаю тестировщика. Так он будет в контексте и меньше рисков, что под видом «багов» он насыплет команде отсебятины.

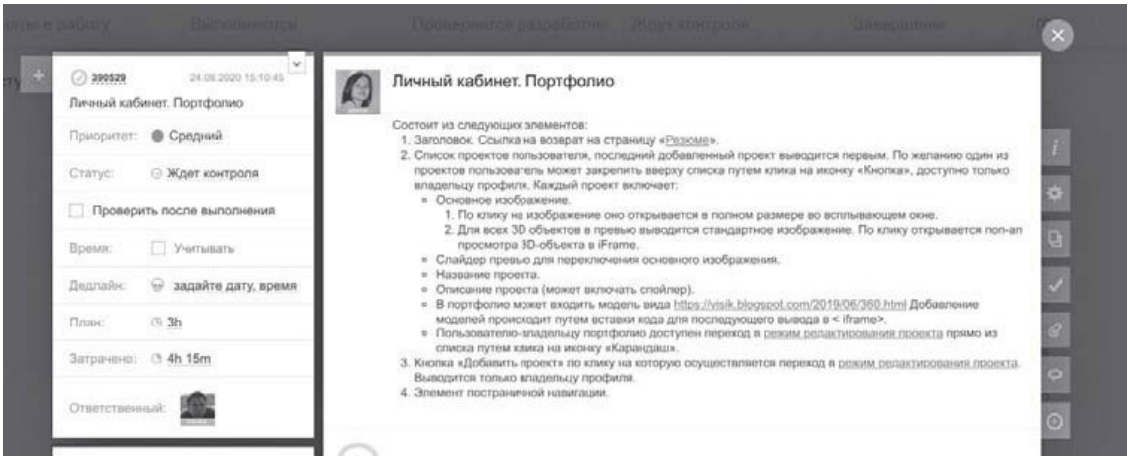
Фиксируем цели спринта. Две-три, не больше. Кратко описываем будущий прирост функциональности. «Реализовать личный кабинет дилера», например. Хорошо бы их вывесить на стене, чтобы спотыкаться о них глазами.

Сразу после планирования я прошу одного из разработчиков и тестировщика еще раз пройтись по задачам и составить краткие чек-листы с критериями приемки. Контекст еще очень свеж. По этим чек-листам разработчики смогут сделать самопроверку, да и тестировщику потом будет проще работать. В итоге карточки задач после планирования выглядят примерно так:



Карточки задач в канбан-доске спринта

Внутри каждой карточки уже развернутое описание.



Карточка задачи с детальным описанием

Очевидно, что процедура планирования занимает кучу времени. Поэтому я не люблю короткие спринты. Две недели, на мой взгляд, – в самый раз.

3.4. Planning Poker

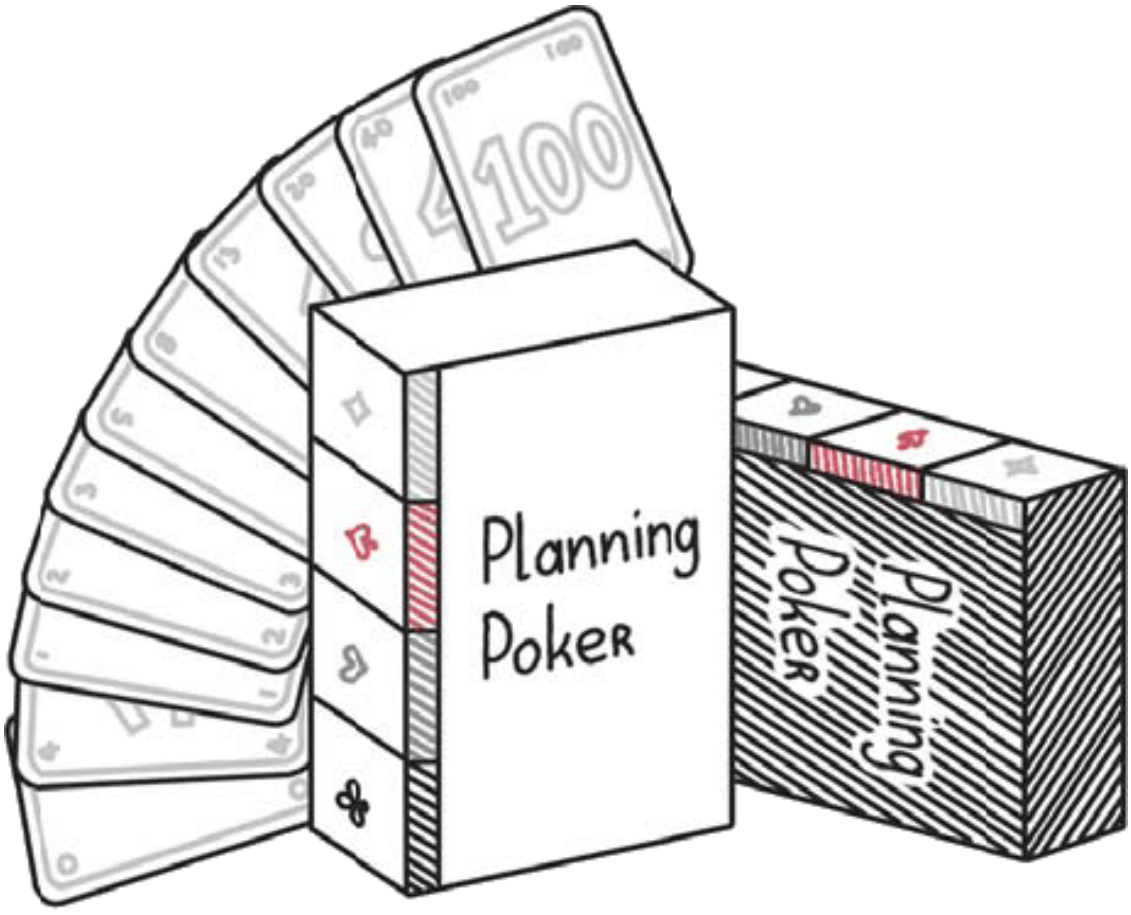
Planning Poker – инструмент для оценки задач. Карты удобны тем, что участники команд не могут ориентироваться на мнение своих коллег – так оценки получаются максимально очищенными от субъективизма и влияния «авторитетов».

В колоде 4 разноцветных набора для 4 участников. Если участников больше – добавляем колоду.

Достоинства карт: 0 (значит задача готова или слишком мелкая), 1/2, 1, 2, 3, 5, 8, 13, 20, 40, 100. Бывают карты с числами Фибоначчи или другой развесовкой, но мне нравятся эти.

Цифры – это либо часы, либо Story Point, в зависимости от того, как вы привыкли. Нам не нужно пытаться сделать сверхточную оценку. Это невозможно, скоро будут доказательства. Нам нужна адекватная оценка.

Итак, ведущий планинга зачитывает задачу. Участники выкидывают карты с оценкой. **Рубашкой вверх!** – это важно, иначе самый авторитетный товарищ продавит свои оценки, а робкие тихони отсидятся.



Карты Planning Poker (planningpoker.ru)

Далее карты вскрываются. Если оценки плюс-минус совпали – фиксируем их в задаче спринта. Если нет – надо обсудить дополнительно.

Например, кто-то выкидывает оценку гораздо большую, чем остальные. Он либо про эту задачу что-то знает-замышляет. Например, был хреновый опыт в прошлый раз или там в коде ТАКОООООЕ! И надо его расспросить. Либо спит (разбудим). После обсуждений – уточняем нюансы и переигрываем кон.

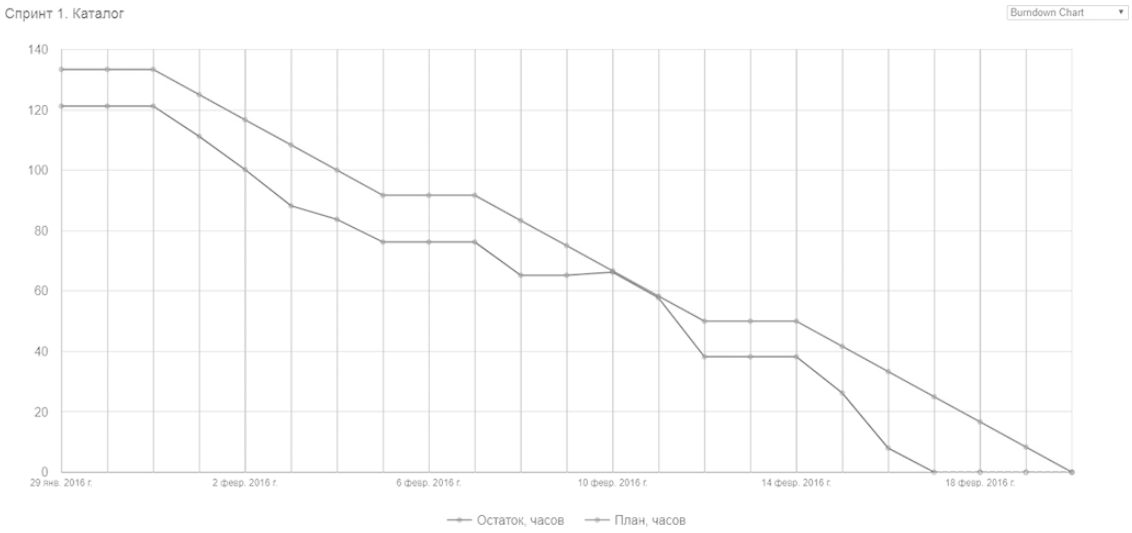
Еще две специальные карты: Coffee/beer – если все затянулось, и надо сделать перерыв, и WTF – для джуниоров, которые вообще ни в чем не уверены.

При планировании спринта я предпочитаю физические карты. С удаленными – online-сервис или простой видеочат, куда на «раз-два-три» все скидывают свои оценки задач.

3.5. Стендапы. Burn Down Chart

Это простая диаграмма «сгорания» спринта. Ее удобно держать под рукой на стендапах.

Синяя линяя – фактически оставшееся время оцененных задач. Красная – плановое время. Мы видим, как по ходу спринта «сгорала» работа. Команда доделала все намного раньше плана, однако во второй трети спринта линии совпали. Тут нужно внимание.



Сначала все шло хорошо. Потом стало плохо.	Очевиден факап. Внезапно работы стало больше, чем казалось в начале. Так бывает, если по каким-то из задач спринта увеличили оценки.	Почти сразу все пошло «через плохо».

Управлять только через диаграммы не получится, но такие простые, наглядные метрики держите под рукой. И задавайте каждый день три волшебных вопроса.

3.6. Стратегии тестирования

Две крайности: отложить тестирование на конец спринта или тестировать каждую закрытую свежевыполенную задачу. Как говорил товарищ Сталин – обе хуже.

Откладывать на потом – плохая идея, однозначно! Вы можете забуксовать в самом конце, и у вас получится сырой спринт. Сдавать стыдно, выбросить жалко. Product Owner будет недоволен. А баги все равно придется фиксить.

Тестировать каждую мелочь сразу – тоже не круто. Во-первых, проверенную задачу могут поломать, когда будут делать какую-то другую. Это называется регрессией (regression bugs). Брукс в «Мифическом человеко-месяце» писал, что это фундаментальная (значит, толком нерешаемая) проблема – исправление одной ошибки с вероятностью 20–50 % влечет появление новой. Полное покрытие ***автотестами*** стоит неоправданно дорого, да и никто не даст на него времени. Надо дело делать, а не абстракциями заниматься.

Во-вторых, часто отдельный тикет бессмысленно проверять, пока не готовы еще парочка связанных. В-третьих, держать тестировщика «на стреме», чтобы сидел и ждал, когда же наконец на него свалится пара задач на проверку – малопродуктивная затея. В-четвертых, пока проект в руках разработчиков, его постоянно ломают и чинят.

Рекомендую проводить ручное тестирование раз в несколько дней и сделать еще один-два финальных, приемочных теста ближе к концу спринта. Как часто тестировать – решите экспериментально. Начните с раза в два-четыре дня. Ошибки исправляйте сразу после тестов, пока воспоминания о коде еще свежи.

Вы помните, мы добавили к задачам критерии приемки. Этакий чек-лист. Пусть его один раз прочекает разработчик при переносе задачи на проверку. Для самоконтроля. А затем еще раз проверит тестировщик.

Если команда зрелая, а бюджеты позволяют (для сайтов это почти всегда не так – всем дорого), покрывайте рутинные проверки автотестами. Или, как минимум, формализуйте тест-план для критических маршрутов проведением ***смоук-тестов*** на ***продуктиве***. Например, в интернет-магазинах покрываем тестами маршрут Каталог → Карточка товара → Добавление в корзину → Корзина → Чекаут. Критических тестов не должно быть много, и они не должны занимать много времени. Но должны вселять уверенность, что ключевые функции в порядке.

В заказной разработке редко, но встречается клиент, который не видит ценности в тестировании. Речь идет не о манипулятивном приеме, попытках отжать скидки или загнать раз-

работчика под плинтус («Я что, должен за ваши косяки платить?!»). А об искреннем непонимании, как возможно, что после тестирования, тест-кейсов и всех этих довольно дорогих процедур – я, как заказчик, нахожу ошибки. Причем такие, которые кажутся мне очевидными. Они меня бесят! Я не понимаю, на что тратятся время и деньги.

Совет

Немного помогает, если у заказчика есть доступ на чтение баг-листов, и он видит, что команда нашла и исправила несколько сотен ошибок. Но утешение слабое, осадочек остается. Отправлять баг-листы нужно до того, как претензия созреет, а не после. Действовать от силы, а не от слабости. При этом должна быть определенная культура ведения баг-листов: смотрите главу «Правила письменной контрацепции». Однако лучше всего попадать в ожидаемое качество и давать гарантию на свою работу.

3.7. Демонстрация продукта. Завершение спринта

Говорят, в Царской России при испытании нового железнодорожного моста под него ставили всех строителей. А сверху пускали паровоз. Мосты век стояли. А разработчик – либо герой, либо – труп.

Нужно демонстрировать результат работы лично, ставить шкуру на кон! Это бодрит. Подстегивает ответственность и рост качества продукта. Боли, правда, будет больше. Но это полезная боль.

Согласуйте демонстрацию заранее. Если проект долгоиграющий – запланируйте стандартное время, в которое будете показывать спринты. Например, каждый вторник, 16:00. Времени резервируйте около часа. Нужна будет либо личная встреча, либо видеосвязь с демонстрацией экрана.

Со стороны бизнеса присутствует как минимум Product Owner. Пользователи и другие сочувствующие – допускаются. Со стороны разработки – вся команда. Понятно, что бывают исключения, но стремимся к этому.

Для начала я вкратце рассказываю, что успела команда за спринт. Вспоминаем цели спринта. Дальше, с экрана, показываю инкремент. Команда помогает, рассказывая, что, как и почему было сделано. В идеале каждый рассказывает про свой вклад. Разработчики помогают с ответами на технические вопросы и обоснованием решений.

Ради чего все это затевается: обратная связь в режиме реального времени. Пусть жесткая, но честная. По сути, обсуждается только первое впечатление, но оно не врет. WYSIWYG – What You See Is What You Get, или «если тебе кажется, что что-то не так, – скорее всего, тебе не кажется».

По опыту большая часть обратной связи будет конструктивной и позитивной. Особенно с англоговорящими заказчиками. Бояться демонстраций не надо. Надо готовиться. Продумайте чуть заранее:

- По каким путям проведете зрителей в продукте.
- Какие кейсы покажете.
- Какие могут понадобиться данные для демонстрации: контент, тестовые пользователи и так далее.
- Какие вопросы скорее всего возникнут. И как на них отвечать.

Позднее Product Owner может потребоваться время помедитировать, поразглядывать продукт, поиграться с ним. Но это он будет делать в одиночестве, сублимируя в бэклог. Возникшие вопросы ему будет не с кем обсудить, и, наверное, он будет предполагать только худшее. Но с этой проблемой он должен справиться сам.

Мы же зафиксируем обратную связь и пойдем с ней на ретроспективу.

Аварийное завершение спринта

Редко, но гадко. Бывает, приходится останавливать спринт. Дело идет медленно и не туда. Команда упарывается, нужных доступов нет, кризис у клиента или еще какая-нибудь гадость. Дергаем стоп-кран, останавливаем спринт. Проводим ретроспективу, решаем проблемы. Делаем рескоупинг (перепланируем спринт). Едем дальше. Таких форс-мажоров допустимо не больше 5 %. Ибо нефиг.

3.8. Ретроспективы. Бородачи тоже плачут

Допустим, на демонстрации Product Owner разнес вашу работу в пух и прах. Справедливо, методично. Или просто очень эмоционально: он оказался злобным, неуязвимым говнюком. Команда сидит подавленная. Кто-то курит прямо в опенспейсе. Провал. Полный капут. «Что я тут делаю?» – читается в глазах бородатых программистов. Пятница, вечереет. Ваши действия?

Рохля разведет руками. Распустит команду по домам. Ага, щас! Мы пойдем в ближайший бар. И будем всякую фигню думать. Про процессы. Технологии. И ме-е-неджера. Рано его руководить поставили.

Люди будут смаковать неудачу, искать причины провала. На минуточку, не в себе, а в руководителе продукта, менеджере, проекте или процессах. Сами выберут такую позицию, когда они – Д'Артаньяны, а остальные – нет. И будут ныть. Пару-тройку раз повторятся такие ситуации, и дело разладится.

Сильный, во-первых, такого не допустит. Во-вторых, если уж подобное случилось, примет огонь на себя. Прикроет команду. И сразу начнет действовать: обсудит с сотрудниками ситуацию, вместе они разберут ошибки, выработают меры на будущее. У команды по итогу сложится чувство, что **меры помогут**.

В скраме есть специальная активность, где мы с командой подводим и обсуждаем итоги спринта. Ретроспектива. Цель ретроспективы – не поныть (хотя иногда хочется), не выпить (хотя кому-то иногда необходимо), не байки потравить. А улучшить рабочие процессы.

Давайте исходить из того, что **намеренно никто не гадит.** Ну ладно, ладно. За редким исключением отбитых чудаков на букву М, которых легко вычислить и отчислить. Но вы серьезно думаете, что программист специально пишет хреновый код? Дизайнер специально рисует дрянной интерфейс? Сложно управлять людьми, которых вы считаете упырями.

Все не так. Им может не хватать ресурсов: времени, мотивации, энергии. В том числе квалификации, чтобы сделать на должном уровне. Вот с этим уже можно работать.

Итак, намеренно никто не гадит. На основе этой идеи вы должны гарантировать безопасность команде на ретроспективах. У людей должна быть уверенность, что:

▶ по итогам ретроспективы никого не накажут;

▶ на ретроспективе ни на кого не наорут, не затроллят и морально не опустят;

▶ за обсуждение проблем не посчитают слабаком (но только на ретроспективе – ежедневных нытиков отправим к мамочке);

▶ и так далее.

Придется стать сильным и тактичным, чтобы вскрывать проблемы команды, не вскрывая при этом людей.

3.8.1. Формат ретроспективы

1. Подготовка

Мы заранее планируем встречу и собираем команду в одной комнате, без посторонних ушей. Сложно, знаете ли, исповедаться на базарной площади. Напомните ребятам, где и когда планируете провести ретроспективу – возможно, они захотят посмотреть свои записи, ***коммиты*** или еще как-то подготовиться.

На 2-х недельный спринт и команду в 3–4 человека планируем минут 40–60. На месячные спринты или большие команды может уйти часа полтора. Я бы не советовал делать ретроспективы еще длиннее – это контрпродуктивно.

Иногда уместна пицца: задушевные разговоры за едой сплачивают команду и снижают уровень агрессии.

Один человек будет ведущим. Как правило – скрам-мастер или менеджер, если он совмещает эти роли.

2. Настройка

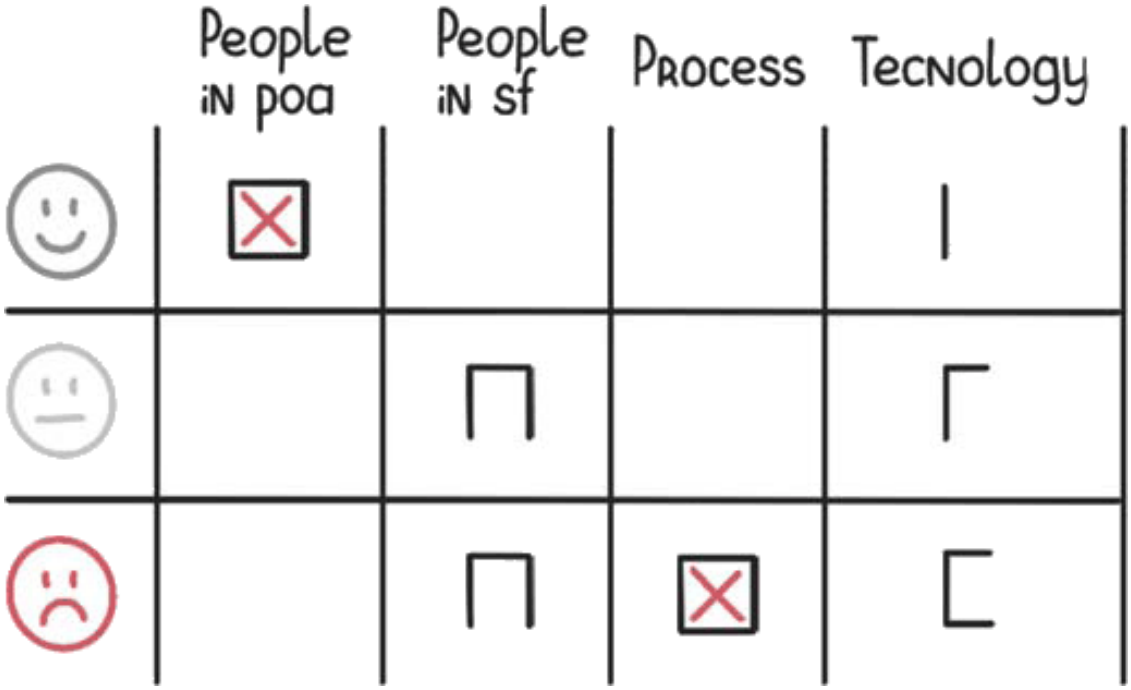
Первые пару минут включаем команду в ~~групповой транс~~ работу.

Здороваемся, налаживаем контакт. Например, задайте простой вопрос, типа «Как дела?». И получите хоть какую-то реакцию. Кивок головы или угрюмое: «Расскажи, дружок мой, Вова, отчего мне так хреново», это окей.

Альтернативные техники, которые мне нравятся:

1. 10-пальцевый опрос. Попросите всех выбросить на пальцах, насколько довольны текущим спринтом.

2. Happiness radar. Чертим матрицу 3×3. По вертикали – смайлики настроения. По горизонтали – Технологии, Люди, Процесс. Каждый ставит палочку, насколько удовлетворен каждым из аспектов. Стикеры нужны для фиксации предложений по ходу.



3. Проверка безопасности. Просим также на пальцах выкинуть, насколько каждый себя чувствует сейчас в безопасности.

Напоминаем цель ретроспективы. Если есть новые участники, не привыкшие к ретроспективам – рассказываем про формат и гарантируем безопасность. Рассказываем про «намеренно никто не гадит».

3. Накидывание на вентилятор

Далее просим по кругу рассказать о плюсах (хорошем) и минусах (плохом) на спринте. Начинайте не с новичков. Идеи приветствуются и фиксируются, но не критикуются.

Знаю пару ребят, которые ведут блокнотики и записывают туда всю бяку по ходу спринта. А потом зачитывают по пунктам. Боюсь, что если прочитать вслух блокнотик от корки до корки, – явится ноющий дьявол. Но раз им так проще – пусть пишут.

Модератор должен чувствовать настроение команды. Уметь разговорить. Уметь слушать. В споры не вступать. Не говниться. Не обесценивать предлагаемые идеи. Модерировать, если идет неконструктивный треп. Подталкивать к поиску решений. Параллельные потоки (когда параллельно обсуждается парочка-другая тем) и болтовню – закрывать. Вытаскивать тыкающихся в телефон наружу. Стараться выявить те проблемы, в которых команда старается не сознаваться даже сама себе.

Модерируем глупые споры, является ли озвученная проблема проблемой. Или про важность проблем. Вместо споров – просто фиксируем. Может, человеку это важно.

4. Подрывай!

Это не каноническая техника ретроспектив. Но иногда я так делаю.

Я люблю порой посидеть на чужих ретроспективах. Одна из интересных проблем, с которой сталкиваешься в сработавшихся коллективах, – ребята не любят взрывать медленно тлеющие конфликты. Всех бередит, но по-тихому.

Например, между тестировщиком и программистом возникла вялая напряженность из-за того, что тестировщик очень тщательно и придирчиво все проверяет. А программист считает, что это излишние придирки и пиксель-дрючево. Но вслух никто ничего не высказывает. Так, срутся в комментах к баг-листам, вяло препираются «баг-не-баг», на что улетает вагон времени и нервов. Копится недовольство: друг другом, проектом, менеджером, клиентом или погодой за окном.

Одна из интересных задач модератора – по косвенным признакам найти такой конфликт и вскрыть (взорвать) его. Еще дядька Макаренко завещал.

Впрочем, на ровном месте накалять не надо. И так, знаете ли, хватает!

5. Идеи. Генератор добра

Нам нужны идеи. Что поменять, чтобы стало чуть легче и светлее. На первой стадии годятся любые, самые безумные мысли. Критиковать идеи нельзя. Отсев будет дальше.

Тут важно выделить две фазы, как в брейнштурме:

1. обсуждение проблем и генерация идей по устранению проблем;
2. выбор среди тех идей, которые будем реализовывать.

В идеале, на каждый наш минус придумываем две-три идеи по его устранению. Собираем, аккуратно записываем. Могут помочь техники из главы 11 по работе с факапами.

6. Хороший план! Плохой план

Далее из идей выбираем несколько (5±2) конкретных улучшений, которые команда готова сделать. Устройте голосование, если идей целая куча. Желательно успеть за следующий спринт.

Некоторые типы идей я отсеиваю. Или прошу переформулировать.

1. «Мы работали плохо, теперь давайте работать лучше».

Спасибо за лозунги. Но я не знаю, как это реализовать.

2. «Не жрать после 18:00» или «Проводить стендап за 10 минут ровно».

Богатая идейка! Больше похожа на правило.

Если бездумно добавлять правила – место кончится. Или будет вагон правил, которые не блюдут. А это дискредитирует власть. Или это будут правила, которые помнит только их автор. И ненавидит коллег за несоблюдение.

Переформулируйте на локальное и исполнимое: «Не жрать после 18:00 в течение следующего спринта», например. Вот это реалистичнее.

3. «Давайте будем закладывать больше времени на подготовку и архитектуру».

Во-первых, это тоже правило. Во-вторых, не люблю, когда добавляются буферы времени.

На это есть Закон Паркинсона: работа занимает все отведенное на нее время. И даже чуть больше. Сколько буферов ни закладывай – будет мало.

Давайте лучше адекватно оценивать задачи и делать дело так быстро, как это возможно, не?

4. «Давайте проверять после тестировщика – другим тестировщиком».

Ну уж нет! Чем больше проверяющих, тем хуже качество. Зачем мне стараться, если за мной перепроверят и под носом подотрут? Рассеивается ответственность.

Я хочу «встроенное» качество как можно ближе к центру создания ценности. Я хочу встроенное в программистов качество. И они это могут!

5. «Пусть менеджер нам кофе приносит. С козинаками. И веером нас обдувает».

На ретроспективах менеджеру легко наловить себе обезьян на голову. Я видел ретроспективы, где команда навешивала на менеджера-слабака кучу правил, которые должен выполнять только менеджер.

С одной стороны, решение проблем команды и правда потребует менеджерского ресурса. С другой стороны, если после ретроспективы вы выходите с ворохом задач чисто для себя – тут явно какой-то косяк.

Надо помогать команде самостоятельно справляться с проблемами, а не быть еврейской мамочкой. Следите за тенденциями.

6. «Давайте все к черту перепишем».

Вот это происходит у меня прямо сейчас, в работающем проекте аптечной сети, где 5000 аптек, первая в стране доставка лекарств online, тысячи пользователей. Просто проекту 4 года, и там нет реактивного фреймворка. А это не так весело и круто. Программисту скучно.

То есть, идея предложена как бы во благо проекта, но чувствуется сильный личный мотив. Почесать ЧСВ, поиграть с технологиями, стать незаменимым и так далее. Словом, почувствовать себя крутым! Я не вижу ничего дурного в таких личных мотивах – они полезны. И без их удовлетворения не будет кайфа на работе.

Но формулировку мы поменяем. «Составить план рефакторинга» – уже теплее. Этот план я внимательно изучу на целесообразность, адекватность прогнозов. Согласую с клиентом и внедрю. Постепенно.

7. «Пусть дизайнеры всегда теперь делают по-другому».

Это когда проблемы пытаются решить за счет отсутствующих на ретроспективе людей. Иногда – оправданно. Но! Естественно, отсутствующие будут не согласны.

Или придумываем правила, которые нужно распространить на всю компанию, а не только на одну команду.

Во-первых, такие правила сложно внедрять: нужно сформулировать, донести до пользователей, придумать контроль и наказания и потом постоянно накачивать в них энергию. А где возьмете дополнительную?

Во-вторых, у менеджера нет полномочий решать за всю компанию. Можно **предложить** какое-то изменение руководству. С должным обоснованием и планом внедрения. В такой формулировке задача уже более вкусная. Но это же думать надо!

А можно собрать дизайнеров и разработчиков вместе, и пусть они глаза в глаза расскажут, что думают друг о друге. Взорвать ситуацию. Как-то сразу тональность критики и категоричность уменьшаются. Я не против глобальных изменений, но тут очень аккуратно работать надо. Нежненько.

8. «Не хочу я быть римскою папой,
А хочу быть владычицей морскою».
Ну, сорян:)

В план должны попадать конкретные, выполнимые идеи. Можно даже по SMART. С конкретными исполнителями. Следующую ретроспективу мы начнем с проверки, что из этого плана получилось. И стало ли от этого лучше (бывает, сделали, как просили, и стало хуже, ага).

7. Заключительная часть Марлезонского кордебалета

Подводим итоги. Кратко зачитываем план. Спасибо, все свободны.

Можно повторно замерить настроение команды, убедиться, что ребята считают, что меры помогут.

Можно спросить обратную связь на ретроспективу.

Если все недовольны решениями, ретроспективой – вы в беде. Очень жаль. Надо чинить ретроспективу.

3.8.2. Форматы фиксации

Мы используем три формата ретроспектив.

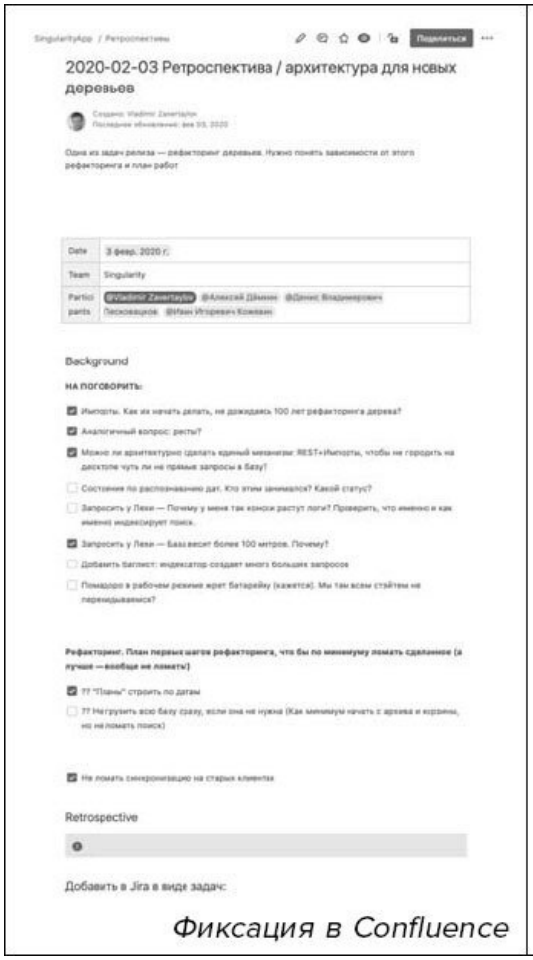
1) Неформальные. Короткая встреча на 15–30 минут, где по очереди даем высказаться всей команде. Плюсы, минусы, идеи, предложения. Конкретные решения фиксируем, если все с ними согласны – забираем в план работ. Подходит, если на проекте все хорошо. Просто градусник. Артефактов не остается.

2) Формат с доской 2×2: плюсы-минусы, идеи, план. И стикеры.

Самое муторное потом – перенабрать писанину со стикеров и завести реальные задачи в тикет-системе. Но вы взрослые, как-нибудь справитесь.

3) Электронные шаблоны.

Грамотный шаблон фиксации доступен в ***Confluence***. Мы используем чуть попроще. Главное – вывести его на большой экран, чтобы все видели, как идет фиксация, и что ничего не забыто и не переврано.

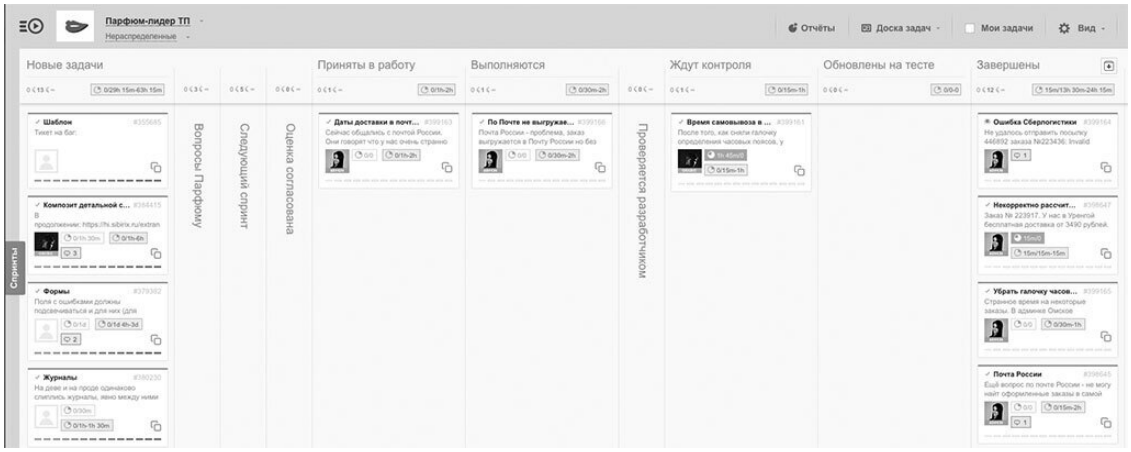


Фиксация в Confluence

Фиксация в Google docs

3.9. Канбан. Когда лучше, чем scrum

Канбан – легковесный и прозрачный процесс. Задачи по мере поступления вывешиваются на доску, с которой разработчик может их забирать и программировать.



Канбан-доска технической поддержки. Фрагмент.

В отличие от Scrum-а, тут нет спринтов. Канбан больше подходит для контроля текущей техподдержки, маркетинга или обработки лидов. В разработке новых функций и версий лучше работает Scrum.

3.10. Куда дели менеджера

В теории нет разницы между теорией и практикой. А на практике есть.
Йоги Берра

Допустим, мы создаем продукт in-house. Собрали разработчиков в одной комнате, сказали работать по Скраму. Самоорганизуйтесь там как-нибудь. Получится? Маловероятно. На хакатон дня в три, может, и хватит, а на полноценный продукт уже нет. И канонический скрам-мастер тут вряд ли поможет.

Самоорганизующейся команде нужен самоорганизатор.

Другой пример. Вот идет разработка. Ни шатко ни валко, по ТЗ. Скорость медленная, баги, Product Owner ругается, команда грустит. Решают попробовать Scrum, а вдруг поможет. Опять нужен кто-то, кто сможет все самоорганизовать, сшить старые процессы с новыми, перевести работу на новые рельсы. Это искусство, тут много дел.

Или, допустим, заказная разработка. Product Owner на стороне клиента есть. Как-то на одной из встреч представитель клиента сказал на полном серьезе: «Да, у нас есть Product Owner, их целых 5» (ой!).

Что ожидается со стороны подрядчика? Что будет кто-то ответственный за проект, с кем можно вопросы порешать. А то и вовсе Proxy-Product-Owner нужен, который за заказчика все порешает, всю обратную связь со всех этих пяти Product Owner (корректнее называть их стейкхолдерами) соберет, решит противоречия и в случае чего – по башке получит. П – перспективы…

Менеджер делает все то, что нужно делать, но что не делает никто другой.

Или, как минимум, организует и делегирует, чтобы делалось все то, что почему-то никто не делает.

Но вот команда взрослеет, Scrum уже отлажен, все привыкли друг к другу и процессу, есть внутрикомандный scrum-мастер. В этом случае роль менеджера действительно уменьшается. И во что он трансформируется – вопрос на вырост. Может, в скрам-мастера. А может, в продуктового менеджера, или аналитика, или помощника Product Owner.

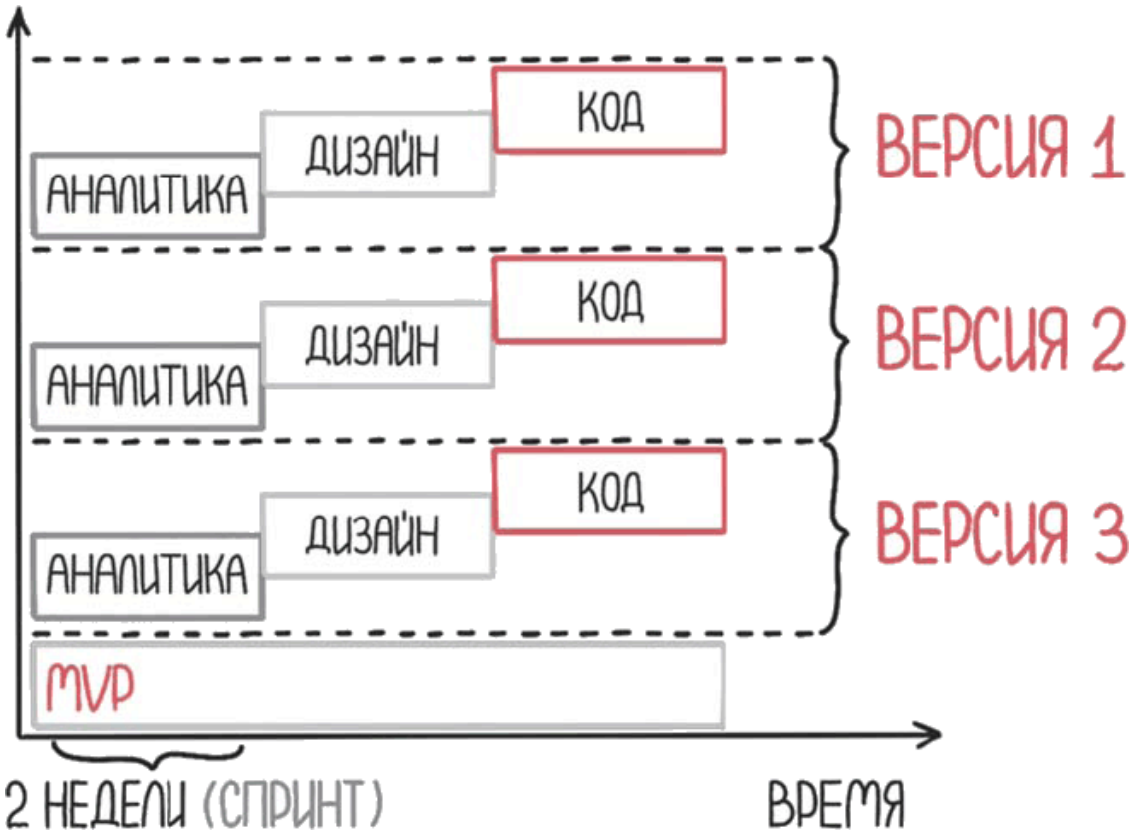
Кажется, редкие-редкие команды достигают такого дзена, где менеджер не нужен. Да и не в нашей ментальности работать без начальства. Короче, менеджер необходим и востребован, аллилуйя.

Можно ли совмещать в себе несколько ролей сразу? Быть и скрам-мастером и Product Owner? Технически – да, я такое видел в молодых командах. Но это антагонистические роли. В них специально зашит конфликт. Product Owner хочет больше и быстрее от команды, и менять все в любой момент. Scrum Master хочет, чтобы команда спокойно работала, и процесс не ломался. Сможете вы такое в своей голове удержать (быть и добрым, и злым), или начнет шизофрения развиваться? Хорошего-то мало. Я бы начал выращивать scrum-мастера внутри команды. Благо – это несложно, дня за три можно справиться.

3.11. Аналитика, дизайн, разработка – параллельно

Обычно нужно несколько спринтов, чтобы получился MVP (minimum viable product) – минимальный жизнеспособный продукт. MVP должен быть клевенький.

Цепкий, энергичный менеджер может организовать параллельную работу над аналитикой, дизайном и кодом. В Scrum-е спринт по аналитике запускаем чуть заранее. За ним – дизайн. И далее уже код. Примерно по такой схеме (правда не факт, что у вас будет так много дизайна и аналитики, но вдруг?).



При параллельной разработке нагрузка на менеджмент и коммуникации растет чуть ли не экспоненциально. Поэтому подход не для новичков, а для матерых организаторов.

3.12. Документация

Agile-манифест (которому сто лет в обед) – набор правильных лозунгов, которые как-то надо адаптировать к практике:

> **Люди и взаимодействие** важнее процессов и инструментов.
>
> **Работающий продукт** важнее исчерпывающей документации.
>
> **Сотрудничество с заказчиком** важнее согласования условий контракта.
>
> **Готовность к изменениям** важнее следования первоначальному плану.
>
> *То есть, не отрицая важности того, что справа, мы все-таки больше ценим то, что слева.*
>
> *Agile-манифест разработки программного обеспечения. 2001 год.*

В зрелом продукте документация нужна, Agile это не отрицает. В небольших web-проектах и приложениях – можно и без нее. Степень замороченности прямо пропорциональна объему проекта, педантизму заказчика и ресурсам. Правда, документация всегда немного отстает от реальности, и это окей.

Работая над ***SingularityApp***, например, я беру какую-то крупную хотелку из бэклога и расписываю, как бы я хотел, чтобы она работала с точки зрения пользователя. Это просто схемки интерфейсов на iPad, скриншоты похожих реализаций и немного текста. Если нужно докрутить формальности – отдаю аналитикам (с проверкой, чтобы не исказили идею). Там уже могут появляться прототипы, описание граничных случаев, интеграционные протоколы и так далее, если мне нужна такая степень проработки и формализма. Такие штуки удобно хранить в ***Confluence*** или ***википедии проекта***. Подойдет и Google Docs.

Уже эти постановки дербанятся на технические тикеты (sprint backlog), и по ним рисуются интерфейсы. Confluence позволяет отправлять задачи в бэклог прямо из текста документов (правда, довольно коряво, но все же). Мелкие хотелки из бэклога можно так детально не разжевывать.

Кодерская и техническая документация, в основном, лежит либо в самом коде (the code speaks), либо в вики проекта.

> Общее правило – положи документацию как можно ближе к месту ее использования.

Иначе про нее забудут, забьют и потеряют. Мне лениво лезть лишний раз за кусочком текста. Я хочу все понимать здесь и сейчас.

Документацию важно увязать с тест-планами, чтобы QA не плодил свои постановки и хотелки.

Пользовательская документация (инструкции, мануалы, вопросы и ответы) – обычно отдельный процесс. Полностью делегируем. В долгих и больших проектах, где важно поддерживать ***консистентность документации*** (и на это есть ресурс), во время написания пользовательских инструкций актуализируются и изначальные постановки: чертежи с iPad заменяются реальными интерфейсами, дополняются моменты, которые уточняются по ходу реализации.

На заказных web-проектах и мобильных приложениях такой контур слишком затратен, тут документацией должен стать сам код, бэклог, ТЗ или описания в ***Swagger***. Хотя время от времени сложные, интеграционные моменты приходится документировать. Особенно, если реализуется API, с которым будут работать сторонние разработчики.

Документация требует времени, ресурсов, отдельного контура управления и внимания. Ну а хорошего технического писателя – днем с огнем.

Нужна ли документация на вашем проекте? Будете ли вы ее писать по ГОСТ 19 или ГОСТ 34, или стандартам ISO? Будете ли использовать ***UML*** или BDD-подход (Behavior-driven development, дословно «разработка через поведение»), ***язык описания сценариев Gherkin***? Вопрос, на который не надо торопиться отвечать. Делайте минимально необходимое и дополняйте по мере необходимости. Не старайтесь заранее все предусмотреть.

ЧЕЛОВЕК, КОТОРЫЙ ВСЕ ПРЕДУСМОТРЕЛ

Однажды человек решил предусмотреть на сайте ВСЕ. ВСЕ тренды, вау-эффекты, поисковые рекомендации… да мало ли. Короче, все. Ну что б потом не переделывать. Это был очень предусмотрительный человек.

И на встречу со студией он решил прийти лично. Надел костюм. Пальто. Шляпу. Взял зонтик. Подумал и на всякий случай захватил солнцезащитные очки. Перчатки. Сотовый телефон. Две запасных батарейки. Это был очень предусмотрительный человек.

Взял пару бутербродов. Бутылку коньяка. Пачку презервативов. Хлоргексидин. Перочинный ножик. Вареных яиц. Ножовку по металлу. Бумажную карту Свердловской области. Аккуратно разложил это по карманам своей жилетки. И не надо думать, что это был какой-то уникум, типа Онотолия. Просто обычный человек, который всегда все предусматривал.

Аптечку. Лыжи. Акваланг. Ласты. Бинокль.

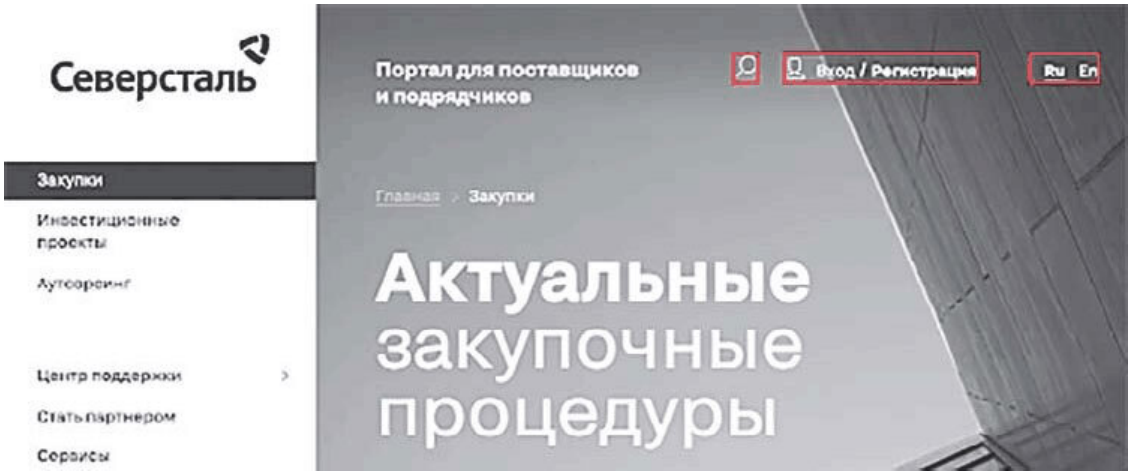
Вышел из дому. Стоит посреди лужи. И думает: «Блин, а все ли я предусмотрел на сайте?»

3.13. Метод красной рамки

Для программистов, если они не шарлатаны, уже привычно работать итерациями. Какая-то функция на экране готова. Какая-то – нет. В новых версиях в каждый спринт, добавляется что-то новенькое. Но в целом можно уже посмотреть, как ведет себя продукт.

Чтобы не возникало вопросов, что уже готово, а что – нет, рекомендую использовать **метод красной рамки**. Суть в том, что прописывается специальный стиль (например. todo), выводящий тонкую красную рамку вокруг нужного нам элемента. Этот стиль присваивается тем компонентам интерфейса – кнопкам, полям, формам и т. д. – которые отражаются на экране, но еще не реализованы в коде. Таким образом наглядно видно, что в данный момент готово, а что – нет.

Этот метод работает и в заказной итеративной разработке. Заказчику не нужно будет постоянно объяснять, куда можно тыкать, а куда – нет. Если что-то не работает – мы четко разграничиваем ситуации «сломалось, баг» или «все под контролем, еще не реализовано».



Одна из первых итераций портала Северсталь. Поиск, вход, регистрация и переключение языков еще не реализованы.

Итак, плюсы:

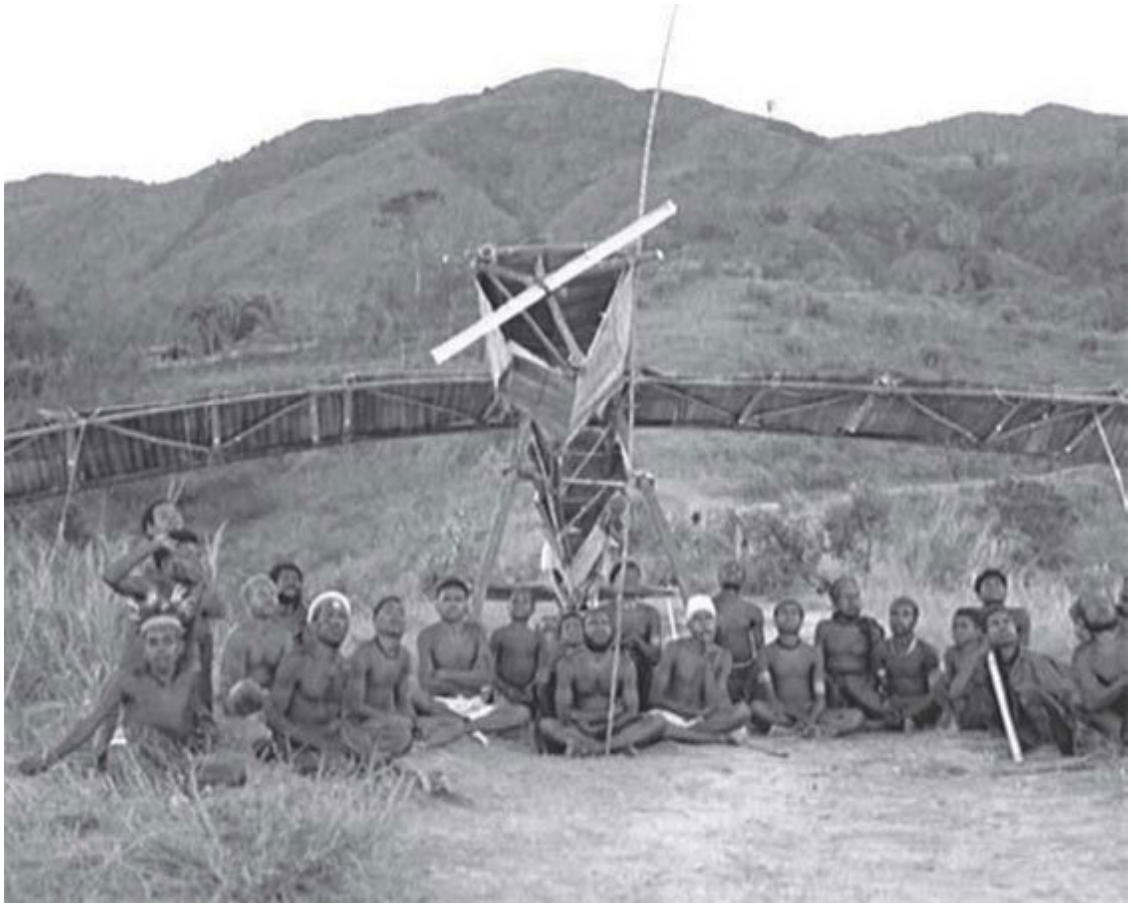
1. Заказчик сразу видит, что готово, а что нет – не тратим время на объяснения, почему не работают некоторые поля.
2. Тестировщик понимает, какие блоки еще сырые, и не тестирует их.
3. У разработчиков не остается шанса промотать задачу.

3.14. Потому что карго-культ!

Во время 2-й мировой войны США строили свои военные базы по всяким разным диким островам в Тихом океане. Продовольствие и шмотки возились самолетами, причем часть груза просто сбрасывалась вниз. Ну, и кое-чего перепадало диким человекам, живущим на этих островах. Причем, иногда перепадало столько, что аборигены полностью забивали на свою хозяйственную деятельность, выращивание бананов и скотоводство. Туземцы быстро уловили, что американцы сами ништяки не производят, а все им достается с неба, за верность духам предков.

Война кончилась, американцы свернули базы. И вот представьте их удивление, когда лет через 20, вернувшись на эти острова, они обнаружили марширующих негров с муляжами ружей, наушниками из дерева, копии самолетов из бамбука в натуральную величину, муляжи аэродромов из соломы… В общем, туземцы довольно четко эмулировали действия американ-

цев, с той лишь разницей, что духи предков почему-то не спешили больше сбрасывать с небес ништяки.



Туземцы не понимали, что стоит за происходящим, не понимали, как устроен мир, и почему с неба падает продовольствие. Поэтому выполняли бессмысленные ритуалы, которые не давали результата.

Не тем ли мы занимаемся, когда выполняем ритуалы, вроде стояния у канбан-доски по утрам или планирования с помощью карт Planning Poker, абсолютно не понимая смысла происходящего? Понимаете ли вы, какие выгоды дает каждый компонент?

Возможно, единственная проблема туземцев в том, что они остановились в своих поисках и экспериментах и не получали какую-либо обратную связь. Истина видимо где-то посередине: разбирайтесь в тех методологиях, которые используете, и постоянно экспериментируйте, оставляя только реально работающие подходы. И ништяки повалятся к вам.

> **Упорство** отличается от **Упертости**. Упорный, если не получилось, пробует другой подход. Упертый продолжает делать то же самое, рассчитывая на другой результат.

3.15. Для тренировки

Итак, мы разобрали Scrum – гибкий простой фреймворк для организации работы команд над современным софтом! Он задыхается в угрюмой, бюрократичной среде и хорош там, где нужно регулярно улучшать продукт, реагировать на изменения в мире и обратную связь. Мобильные и веб-приложения – самое то!

Задание – иди и внедряй!

Попробуйте это внедрить у себя. Посмотрите, что заработает сразу, а что – ни в какую. Проведите несколько спринтов и ретроспектив.

И постарайтесь не скатываться в СКРАМНО («у нас Скрам, НО без итераций») и Карго-культ.

Успехов!

Литература

▶ Джефф Сазерленд, «Scrum: Революционный метод управления проектами».
▶ Фредерик Брукс, «Мифический человеко-месяц».
▶ Антон Макаренко, «Педагогическая поэма».
▶ Скрамгайд в доступном переводе от Сибирикс (QR-код слева).

https://blog.sibirix.ru/2018/10/31/scrumguide-all/

Пояснения

Канбан-доска – инструмент метода разработки «Канбан», представляет собой доску, разделенную на этапы работ, с задачами в виде карточек, которые перемещают по доске по мере их продвижения по этапам.

Рефакторинг – перепроектирование или переработка кода, изменение его структуры, которые не производят функциональных изменений, но существенно облегчают его работу.

Закон Мерфи – шутливый философский принцип, который звучит как: «Все, что может пойти не так, пойдет не так».

Story Point – метод оценки сложности задач, когда за 1 балл принимается самая простая задача, а все остальные оцениваются относительно нее.

Автотест (автоматизированный тест) – это скрипт, имитирующий взаимодействия пользователя с приложением, для локализации ошибок в работе сайта, приложения или ПО.

Смоук-тест – минимальный набор тестов на явные ошибки, который обычно проводит программист. Без прохождения такого теста нет смысла затевать более глубокое тестирование.

Продуктив – уже запущенный сайт, живущий на сервере заказчика.

Коммит – в системах управления версиями коммит добавляет последние изменения в часть исходного кода в хранилище, делая эти изменения частью основной версии хранилища.

Confluence – софт для для командной работы, удобный для распределенных команд за счет опций совместной работы.

SingularityApp – мощный хаос-менеджмент планировщик, разработанный в студии «Сибирикс». Существует в виде онлайн-версии для ПК и приложения для смартфонов на Android и iOS.

Википедия проекта – база знаний, где хранятся подробные руководства функционала продукта.

Консистентность документации – цельность документации и согласованность документов друг с другом.

Swagger – язык описания интерфейсов для описания RESTful API, который используется для проектирования, создания, документирования и использования веб-сервисов RESTful.

UML – это специальный язык для описания разных бизнес-процессов, структур и прочих вещей, для которых необходимо последовательное описание. В запутанных ситуациях с большим количеством сущностей может сильно помочь. UML включает несколько видов диаграмм: диаграммы последовательности, компонентов, объектов, структуры, синхронизации и так далее.

Язык описания сценариев Gherkin – человеко-читаемый язык для описания поведения системы, который использует отступы для задания структуры документа (пробелы или символы табуляции). Каждая строчка начинается с одного из ключевых слов и описывает один из шагов. Обработчик разбивает файл с тестами на функции, сценарии и входящие в них шаги.

Часть 4
Аналитика и продуктовые техники

Большинство удачных идей оказывается неудачными.

Британские ученые подсчитали, что 92 % стартапов проваливаются в течение 3 лет. Главная причина – продукт нафиг никому не сдался (42 %).

Я чувствую так же. Мы плотно работали со стартапами с 2010 года. Некоторые из них живы, и даже неплохо живут. Но, оглядываясь назад, я вижу кладбище.

Наверное, всем, кто работал в заказной разработке достаточно долго, знакомо это чувство. Много времени, сил, энергии, интеллекта и отваги вкладывается в какой-то проект, в какую-то идею… И где это все годика через 4? Пшик, очень жаль…

Другое дело, когда проект создавался под конкретную задачу, было заранее понятно, что есть сегмент пользователей, у которых в нем есть потребность, и посчитана экономика. Зачастую это обычные, работающие бизнесы, решившие поглубже залезть в интернет или автоматизировать рутину. Тут, несмотря на кризисы, выживаемость и успешность близки к 95 %. Пойман **Product Market FIT (PMF)** – состояние, когда клиенты довольны продуктом и рекомендуют друзьям. А сам бизнес растет.

Я уже говорил об этом во введении, об экологичном пути руководителя проектов, но скажу еще раз. Моя религия и пятнадцать лет опыта выращивания руководителей проектов говорят, что если вы только осваиваете управление digital-проектами и не имеете глубокого технического бэкграунда, то **самый экологичный путь стать первоклассным специалистом – это поработать некоторое время в тестировании и аналитике.**

В этой части мы посмотрим на техники управления продуктом, которые помогут минимизировать галлюцинации основателей, нащупать сегменты пользователей, протестировать гипотезы и т. д. Этими инструментами должны хорошо владеть руководители продуктов и аналитики. В маркетянские штуки уходить не будем.

Для руководителей проектов инструменты полезны. Во-первых, для понимания, как Product Owner принимает решение. Во-вторых, на вырост. В-третьих, скорее всего, Product Owner что-то из этих штук сам делать поленится, или его нужно будет возвращать в реальность. А кто будет делать все то, что делать надо, но не делает никто другой – вы уже в курсе.

Итак. Путь к крутому руководителю проектов и продуктов лежит через аналитику. Разберитесь в этом. Погнали!

4.1. Цель аналитики digital-проектов

Если не знаешь, какой херней ты занимаешься на работе, – назови это «Аналитика».
Народная мудрость

К сожалению, вокруг аналитики digital-проектов в последнее время накрутили много хераборы. Методов и подходов стало слишком много. Непонятно, за что хвататься и что применять.

Этап аналитики нужен, чтобы верно понять ожидания заказчика, спроектировать оптимальное решение поставленных задач и грамотно спланировать бюджет.

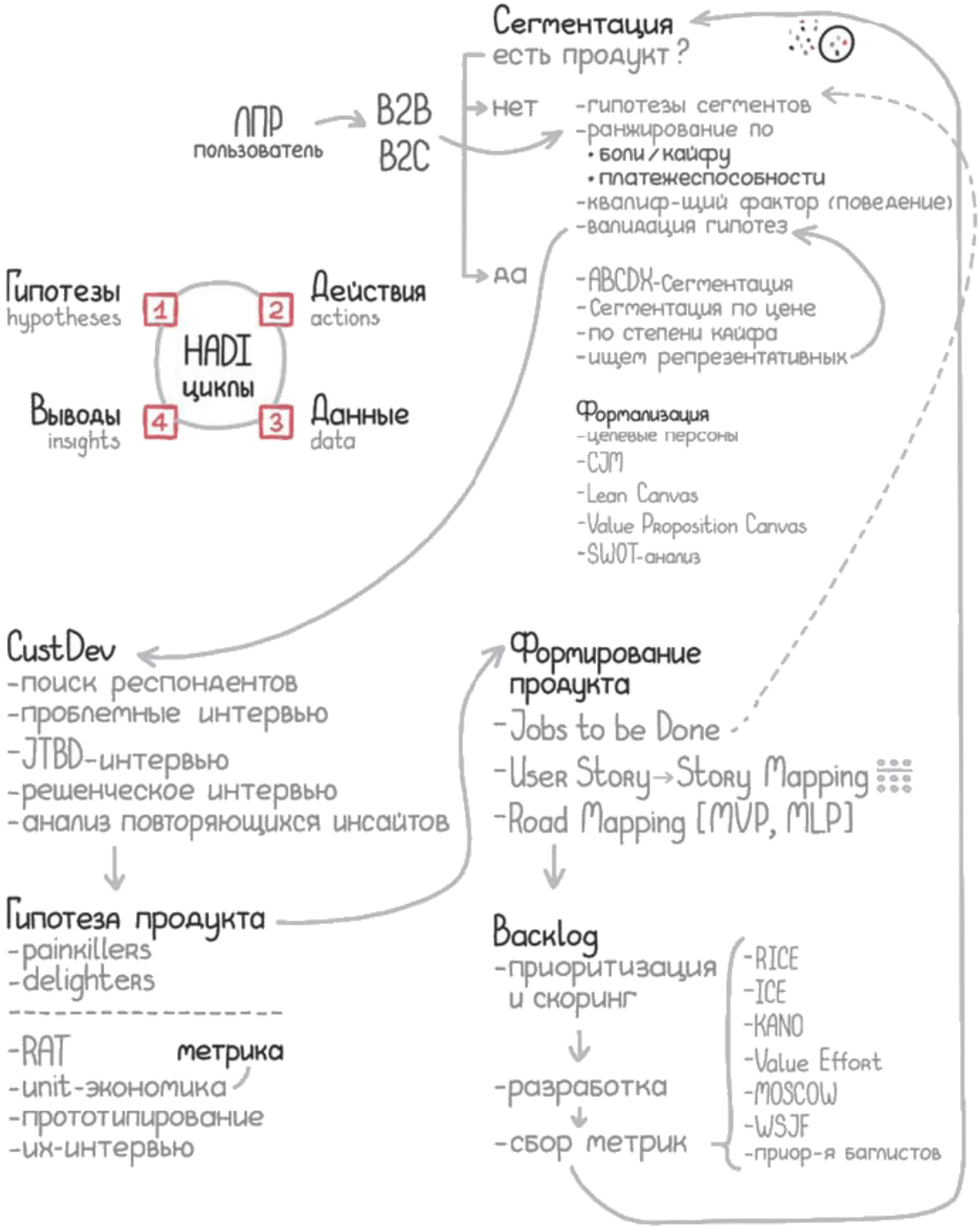
Если раньше аналитика была вспомогательной функцией, сейчас – стала чуть ли не самоцелью. Я видел команды, которые по полгода играют в «Дизайн-мышление» или еще какой-то новомодный метод, производя только трындеж. Бюрократия – такая бюрократия. Правда, кризис оставил бездельников за бортом.

Давайте исходить из того, что:

> Цель аналитики в digital-проекте – получить **структуру** будущего проекта и **четкий план действий**: сначала мы делаем это, потом то, а затем – вон то.

Понятно, что и план и структура со временем могут меняться. Понятно, что в сложном проекте лучше семь раз подумать. Поэкспериментировать. Но в какой-то момент нужно перестать анализировать все подряд. И начать писать код.

4.1.1. Что нас ждет в этой главе



Инструментарий руководителя digital-продукта

Для заказной разработки нам понадобится Агрегация требований, прототипирование и подготовка технической документации. Когда есть заказчик (или владелец продукта) – этого достаточно.

Для самого владельца продукта существует ряд дополнительных подходов: HADI-циклы, CustDev, методы сегментации пользователей и различные способы скоринга бэклогов.

Методы дополняют друг друга. Однако стоит к ним относиться как к арсеналу: снаряжаясь на войну, вы не сможете забрать на себе весь арсенал. Вы выберете один или два вида ору-

жия и что-то из защиты. Кто-то возьмет лук, поскольку меткий. Кто-то меч по руке. Но весь арсенал вы на себе не потащите.

Точно так же нет смысла тащить в проект все известные способы приоритизации бэклогов. Выберите один, который вам нравится. И заточите его под вашу ситуацию.

Как этим пользоваться: прочитайте главу один раз. Ознакомьтесь со всеми методами. Выберите те, которые вам понравились. Изучите их подробнее. Начните применять. Время от времени пересматривайте подходы. Возможно, в качестве эксперимента, захочется попробовать какой-либо другой метод.

Итак. Что нас ждет.

Во-первых, мы рассмотрим метод **Агрегации требований**. Формально он состоит из 5 шагов:

1. Видение проекта. Это основные характеристики будущего продукта. Его цели и задачи – так, как их понимают создатели продукта.
2. Целевые персоны. Здесь мы идем от пользователей. Сегментируем рынок. Выделяем боли и потребности пользователей. Проводим интервью. Строим сценарии поведения и CJM – карту путешествия клиента.
3. Конкурентный анализ.
4. Структура проекта. Какие экраны нам нужны. Что на них будет.
5. Идеи на будущее. Все то, что было бы неплохо сделать.

Во-вторых, посмотрим, как делать регулярную аналитику в продуктовых командах на основе HADI-циклов. Разберем 4 силы, действующие на продукт. Поговорим о фреймворках RAT и JTBD.

В-третьих, поговорим про юнит-экономику продукта. Эта информация будет нужна руководителю проекта, чтобы понимать, на основе чего руководители продуктов принимают решения. На самом деле, метрик бесконечное множество и тут легко запутаться.

В-четвертых, поговорим про техники скоринга и приоритизации бэклогов, так, чтобы галлюцинации основателей действовали минимально.

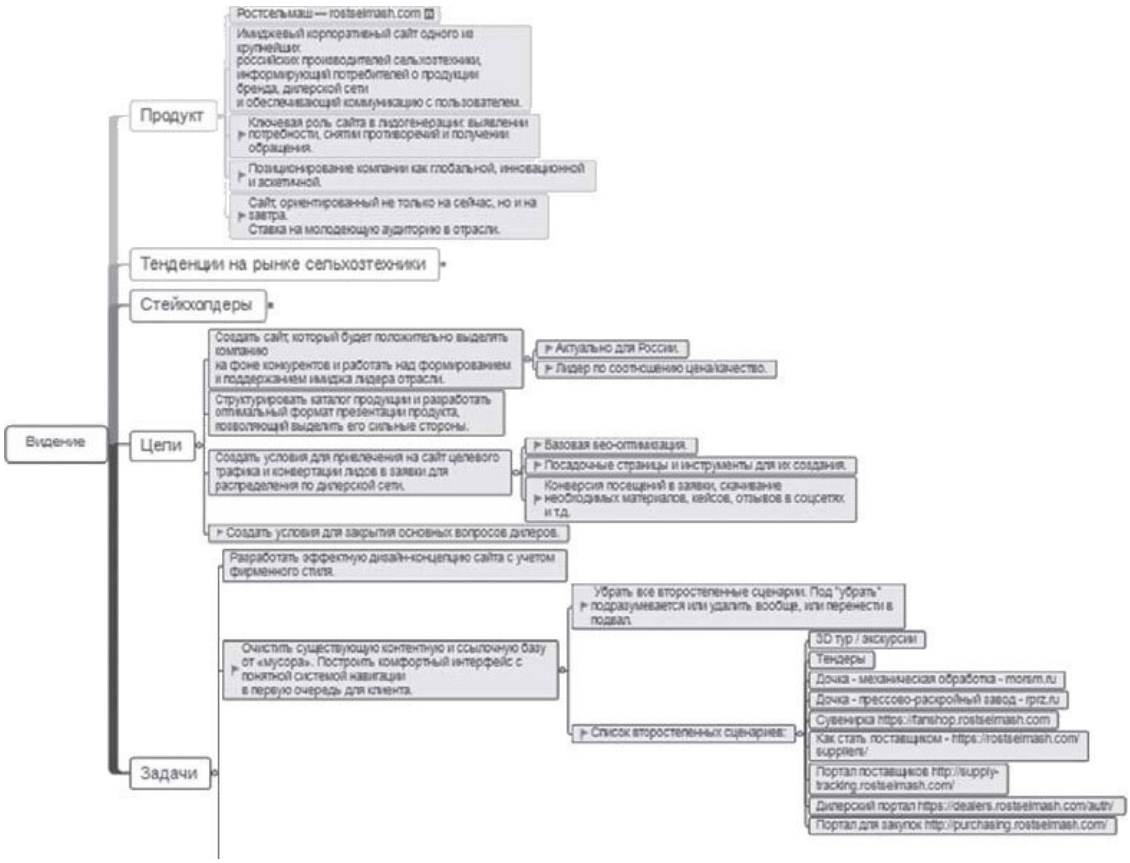
В-пятых, посмотрим, как UML-диаграммы помогают при проектировании программных продуктов, продумывании взаимодействия с пользователем.

4.2. Агрегация требований в заказной разработке

Эта техника лучше всего работает в заказной разработке. Либо когда уже в общих чертах понятна задача, осталось только вытащить детали из голов стейкхолдеров, примерить их на целевую аудиторию, убедиться, что не допущены ошибки конкурентов и учтены тренды отрасли. Подходит для большинства сайтов и мобильных приложений. Позволяет синхронизировать видение у заказчика и разработчика, а также довольно точно подсчитать бюджет разработки. Это, в основном, – кабинетный анализ, который можно подкрепить интервью с потенциальными пользователями (см. главу о Customer Development) и A/B-тестами. В итоге у нас должна получиться структура будущего проекта. Для фиксации предлагаю использовать формат интеллектуальных карт.

Агрегация требований – основа проекта в заказной разработке. Именно на этом уровне мы понимаем, что на самом деле нужно заказчику и его клиентам. Где границы возможного с точки зрения человеческих и финансовых ресурсов. Как лучше всего сделать сайт или мобильное приложение, какие системы надо сшить друг с другом, чтобы решить поставленные задачи. Как выделиться в конкурентной среде и при том уложиться в реальные для заказчика бюджет и сроки.

Чаще всего, в заказной разработке мы создаем Агрегацию требований в формате интеллектуальных карт. Например, XMind или XMind Zen. Альтернативные программы можно найти в конце главы, в списке литературы.



Агрегация требований РостСельМаш (фрагмент)

По итогам агрегации мы узнаем:

▶ какие цели и задачи у будущего проекта;

▶ что ждет от него целевая аудитория;
▶ что хорошего и плохого в проектах конкурентов;
▶ как все это учесть и совместить, чтобы одно другому не мешало;
▶ как лучше запустить проект: весь сразу или по частям, и с чего начать;
▶ какой будет структура сайта, на основе которой аналитик готовит прототипы и пишет техническое задание;
▶ рассчитываем довольно точно сроки и бюджет проекта.

4.2.1. Стейкхолдеры

Стейкхолдеры (stakeholder) – заинтересованные в проекте лица. В заказной разработке от них исходит большая часть требований.

Однако бывает, что стейкхолдеры явно не определены. Прячутся за корпоративной иерархией. Влияют, но не отвечают.

В процессе реализации проекта власть в организации, если взять ее за 100 %, как-то перераспределяется. Как только появятся первые результаты проекта – все, кто хоть как-то был затронут этим решением, выскажут свое мнение. Иногда этих мнений слишком много. Или мнения слишком громкие. Или противоречивые. Придется разбираться. Чем раньше, тем лучше.

Проекты, в которых заинтересовано первое лицо компании-заказчика, проходят гораздо ровнее. Нужно четко понимать, кто у заказчика Альфа, кто Бета. Проекты, в которых внутри заказчика постоянные споры и конфликты, и нет никого умного, кто мог бы во всем разобраться и принять твердое, окончательное решение, с треском проваливаются. Либо идут медленно и печально. Либо требуют молоть языком больше, чем писать код.

Итак. В заказной разработке важно как можно раньше выявить стейкхолдеров. Выявить их интересы. Скрытые и явные конфликты. Понять иерархию стаи. Заручиться поддержкой Альфа- и Бета-стейкхолдеров.

4.2.2 Видение проекта

Первый шаг Агрегации требований – это видение проекта или его основных будущих характеристик. Здесь мы определяем, какой продукт мы делаем, и что он из себя представляет. Видение должно быть конкретным: если при смене названия бренда видение клиента останется прежним, оно никуда не годится.

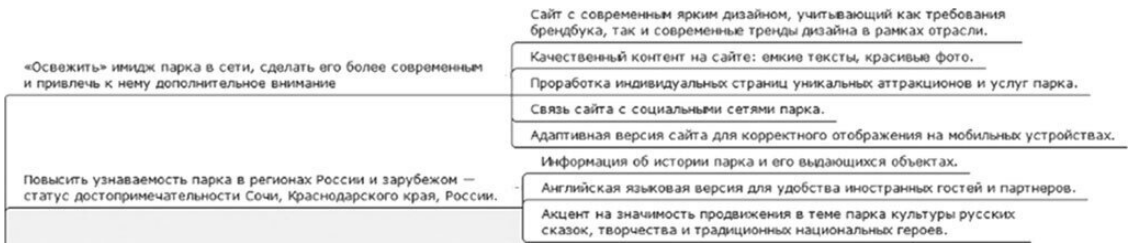
Плохо	Хорошо
Продукт: интернет-портал для обучения специалистов в сфере строительства.	Продукт: учебно-рекрутинговая платформа для проектно-строительного рынка. Служит для бесплатного обучения и тестирования специалистов по BIM-технологиям (личного или корпоративного). А также подбора сотрудников в сфере информационного моделирования зданий.

Обозначаем перечень и ожидания заинтересованных лиц (стейкхолдеров). На этом этапе целесообразно рассматривать стейкхолдеров только со стороны заказчика, поскольку пользователям сайта посвящена отдельная вкладка. Чаще всего заинтересованные лица делятся на несколько уровней:

- ▶ собственники и управленцы (могут совпадать, но не всегда);
- ▶ представители отделов продвижения и продаж;
- ▶ обслуживающий персонал (операторы, контент-менеджеры, администраторы).

Разумеется, у всех у них очень разные ожидания от будущего проекта. Важно обозначить главные направления, добраться до сути: зачем на самом деле создается проект, как он должен изменить жизнь причастных к его эксплуатации людей.

Необходимо развернуто зафиксировать суть ожидания и кратко обозначить, какие пути решения мы видим для него на этом этапе. В ходе дальнейшего анализа, в том числе после получения отклика со стороны заказчика, список решений может корректироваться. Часть идей может быть отложена до будущих этапов развития сайта.



На этапе видения определяются **цели** и **задачи** проекта. Цель – чего именно хотим достичь. Увеличение конверсии, увеличение числа интернет-заказов, оптимизация работы менеджеров. Задачи – что конкретно будем делать. Семантическая разметка, фишки в юзабилити, интеграция с ***CRM***.

Важно! Не нужно писать сюда все бизнес- и жизненные цели, о которых упоминал заказчик – только то, что реально касается разрабатываемого проекта, может быть объективно оценено и измерено. К примеру, если пишешь о росте конверсии, сразу подумай, какие наши действия к этому гарантировано приведут, и как вы с заказчиком измерите «было-стало».

Вкладка формируется на основании данных, полученных при изучении первичных материалов от заказчика, его брифования, общения со стейкхолдерами. С поправкой на опыт и здравый смысл.

Кстати, про здравый смысл и галлюцинации!

4.2.2.1. Галлюцинации основателей

Мы с тобой свернули не туда вообще
И все закончится для нас теперь так себе.
Znaki

Вы когда-нибудь пробовали отговорить основателя от его идеи? Большинство из них настолько сильно убеждают себя, что на их продукт есть спрос, и настолько харизматичны (или властны), что проще ~~дать~~ сделать то, что они хотят, чем объяснить, почему нет.

Вот несколько галлюцинаций основателей, которые мешают:

▶ «Мы знаем нашу целевую аудиторию, просто сделайте, как мы сказали». По факту, аудиторию они не знают, а мантра «мы все знаем», повторенная несколько раз, дает ложную уверенность контроля. Типичный итог – отсутствие потребности в продукте. Нет сегмента покупателей.

▶ «Мою идею украдут». Сразу тревожный звоночек. Идеи не стоят ничего. Артемий Лебедев писал про «Идею на минус миллион» и вообще считает, что идеи имеют отрицательную ценность.

▶ «Мой продукт должен быть идеальным». Перфекционизм. Во-первых, это дорого. Во-вторых, за этим прячется боязнь показать продукт рынку. А вдруг обидят?

https://www.artlebedev.ru/kovodstvo/sections/161/
«Идея на минус миллион» от Артемия Лебедева

В ту же копилку:

▶ Тантрический стартап: три года без релиза, ни дня без ***чендж-реквеста***!

▶ Обратная связь от пользователей откладывается на месяцы.

▶ «Сделайте нам прибыльный проект за процент от будущей прибыли». Нет денег на фоне повышенной хитрожопости. Либо нет денег на разработку. Либо на маркетинг. Либо на то и другое.

▶ Усложнители. Затея либо сразу настолько сложная, что в деталях путается даже сам основатель. Либо пытаются использовать какую-то технологию там, где она нафиг не нужна.

Блокчейн для учета шкурок крупного рогатого скота, например. Фаза анализа проблемы и потребности пропускается, идем сразу в архисложное решение.

▶ Немасштабируемые продукты.

▶ Подражатели. Уже давно замечал, что заявки на разные типы стартапов приходят волнами. Был когда-то всплеск на социальные сети. Аналоги купонаторов. Скандинавских аукционов. Как-то осенью нас засыпало заявками «Продайте долю в игре в соцсетях». Потом были маркетплейсы. Где-то людей «Бизнес-молодость» зомбирует, честное слово!

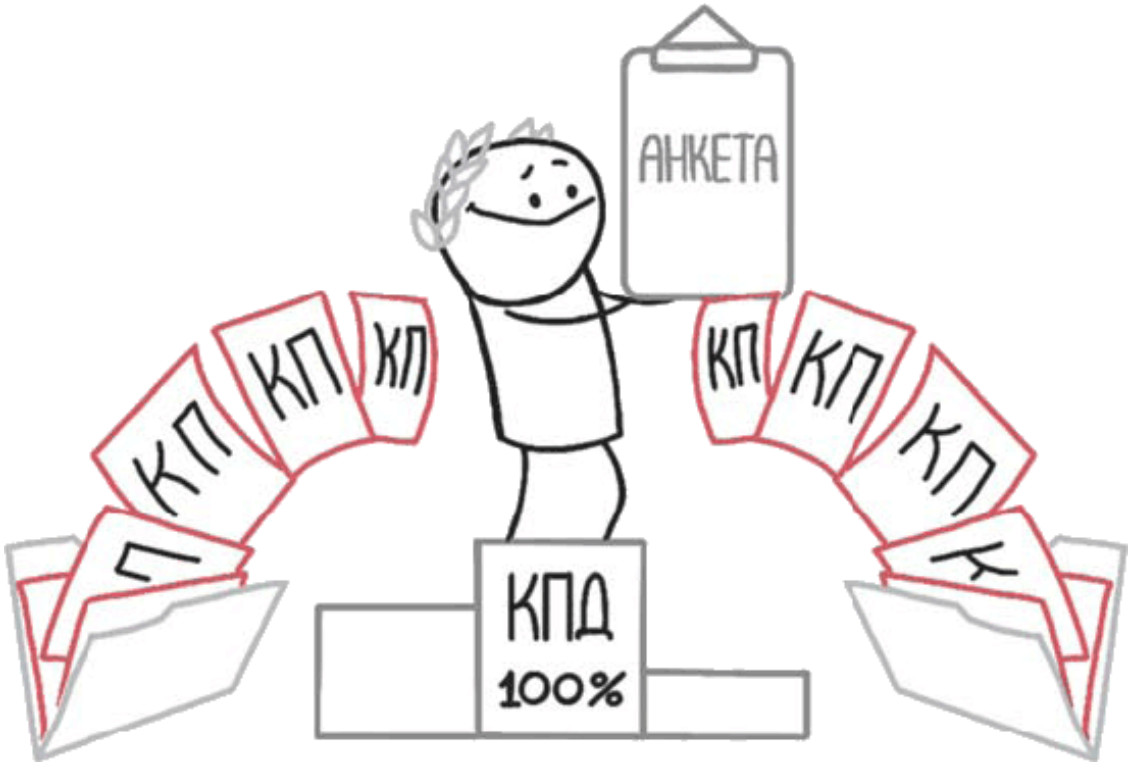
Давайте исходить из того, что у основателя есть определенная стратегия, видение, а **наше дело – помочь все это реализовать по красоте, исходя из нашего опыта и знаний**.

4.2.2.2. Почему брифы – зло

Я не люблю письменные брифы. В своей работе мы их почти не используем. Брифы не дают ясной картины. Теряется много важных деталей. И главное!

Брифы мешают строить долгосрочные отношения с клиентом!

С другой стороны, я понимаю заказчиков. Вернее, менеджеров на стороне клиента, которых отправили «найти подрядчика». Они (агентства), блин, все одинаковые! Гораздо проще заполнить один раз анкету. Разослать в сотню агентств. Получить КП-шки. Распечатать, отсортировать по «Итого:». И положить их боссу на стол. Минимальная ответственность. Минимальная трудоемкость. Божественный КПД – результата можно целую кучу навалить.



С одной стороны, так было и будет всегда. С другой стороны – понятна степень заинтересованности проектом. Агентствам нужно уметь под это подстраиваться.

Любые проекты, стоимостью в миллионы и длительностью в месяцы, обязательно обсуждаются устно. И не один раз. Есть примерный и понятный круг вопросов, с которых можно

начать. Однако нужно иметь достаточно опыта, чтобы погружаться в глубину по любому из вопросов.

Компания
Какие есть компетенции?
В чем суть бизнеса? Что приносит доход?
Узнаваем ли бренд? Какова лояльность аудитории?
Какие есть сложности?

Продукт
Сильные стороны. Почему это должны купить?
В чем отличие от аналогов?
Есть ли товары-заменители (не прямые)?
Жизненный цикл. Как часто пользуются? Как долго?

Покупатель (ЦА)
Кто он?
По сегментам аудитории: объем группы, темпы роста и т. д.
Чего он хочет? Потребности?
Какая цена приемлема?
Как привык потреблять? Получать информацию?

Конкуренты
Кто конкуренты? Какие у них сильные стороны?
Насколько наполнен рынок?
Чем вы лучше других? (Осторожнее с этим вопросом – тут горят пуканы!)
Каковы преимущества? Есть ли уникальные продукты, фичи?
Какие есть трудности и барьеры?
Планы развития?

Это общие направления вопросов, которые нужно адаптировать под каждый конкретный проект после изучения вводной информации. Может быть много нюансов.

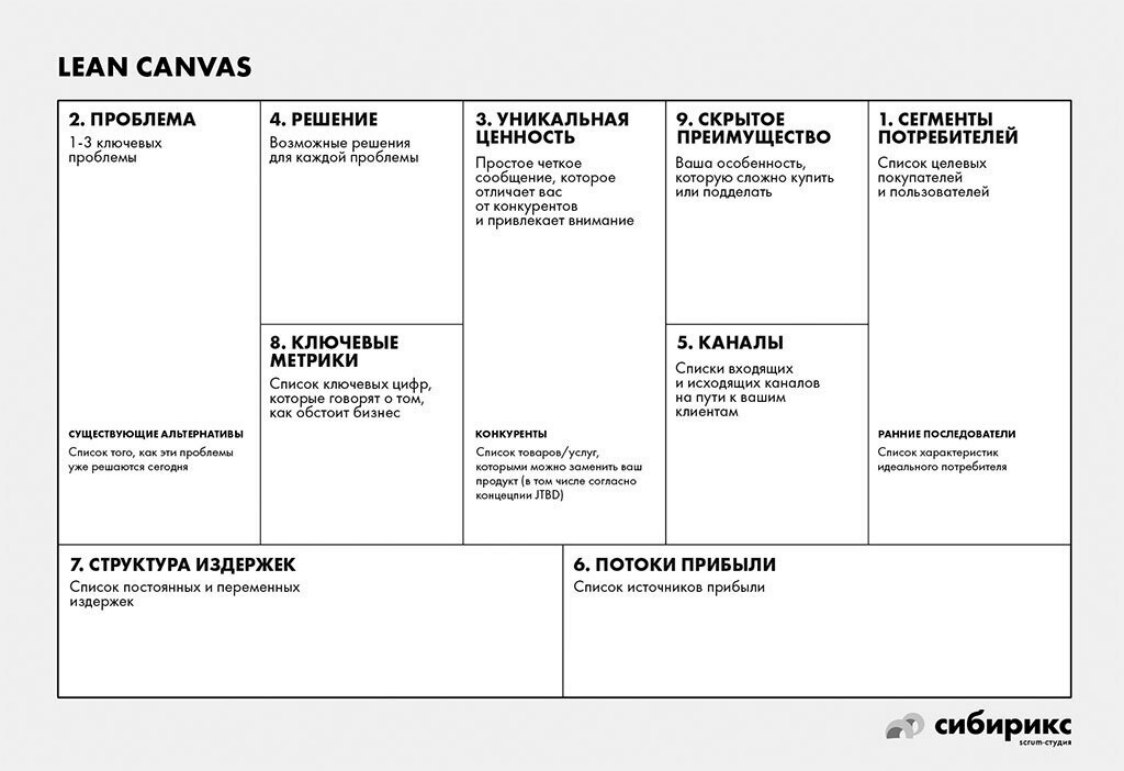
Например, на e-commerce-проектах возникает несколько десятков вопросов по экономике, налогам, логистике, возвратам, интеграциям с внешними системами и т. д. Нужно включить вопросы по всему, что кажется непонятным и странным в проекте. А также затронуть все вопросы, ответы на которые лягут в основу вкладок Видение и Анализ целевых персон.

Итак. Ни один бриф не заменит голову. Письменные брифы – скорее, зло. Какова альтернатива?

4.2.2.3. Lean canvas

Lean Canvas – это популярный шаблон для фиксации целей продукта. В его основе – шаблон бизнес-плана Александра Остервальдера, оптимизированный под стартапы. Главный плюс – всю концепцию проекта можно разложить на один лист А4. И конечно, эта бизнес-модель вдохновлена методологией Lean Startup (читайте книгу Эрика Риса) и пропитана философией бережливого мышления.

https://disk.yandex.ru/i/VsCmklWdrAm7Ng
QR-код, по которому можно скачать шаблон Lean Canvas



Сегменты потребителей

Впишите сюда не только тех, кто покупает продукт, но и тех, кто им будет пользоваться (родители покупают игрушку, а ребенок ей играет). Еще лучше – собрать реальную группу людей, которые будут первыми тестировать продукт и помогут проверить гипотезы.

Проблема

Какую потребность ваш продукт или услуга закрывают? И кто еще это делает на рынке уже сейчас? Не забудьте, что помимо прямых конкурентов есть вторичные и косвенные (подробнее об этом поговорим в параграфе о Jobs To Be Done).

Уникальная ценность

По-другому – ***УТП***, или почему клиенты захотят купить именно ваш продукт? В чем его ценность?

Решение проблемы

Пригодятся интервью с потенциальными пользователями и исследования, чтобы подтвердить ваши гипотезы.

Каналы

«Детям – мороженое, бабе – цветы» – для каждой аудитории ищите свои каналы.

Потоки прибыли

Кто платит. Кому. Когда. И за что. Если продукт или услуга бесплатны, решите, как будете монетизировать проект.

Структура издержек

От зарплаты работникам и арендной платы до затрат на рекламу и создание сайта.

Ключевые метрики

Подумайте, как оценивать результат: как будете измерять интерес к продукту, на какой стадии жизненного цикла клиента вы готовы назвать его постоянным и т. д.

Скрытое преимущество

Найти его не так-то просто, поскольку это может быть не самая очевидная вещь: прямые руки работников или крутые поставщики.

4.2.2.4. Видение. Итоги

Итак.

1. Для фиксации самой верхнеуровневой информации о проекте удобно использовать Lean Canvas. Это может быть что-то типа паспорта проекта. Его отправной точки. Можно охватить взглядом и сразу понять суть.

2. Длинные письменные брифы лучше не использовать. Обсуждайте проект устно. Общайтесь. Погружайтесь в детали. Фиксируйте. Обычно для формирования видения нам нужно 1–3 сессии с клиентом. Важно быть на одной волне и строить долгосрочные отношения.

3. Для фиксации видения проекта удобно применять ***Mindmap***. Важно учесть мнения всех стейкхолдеров. Выявить противоречия. Разрешить их. Сформировать цели и задачи проекта.

4.2.3. Потребители, сегменты, аватары и целевые персоны

Допустим, вы продаете стулья. Значит ли это, что ваша целевая аудитория – все люди, у которых есть ягодицы? Не значит. Да и достучаться до всех вы, скорее всего, не сможете.

Реальное объявление в детском парке. На какой сегмент рассчитывает автор?

Сегментация – разделение всего рынка на сегменты из потенциальных клиентов со схожими потребностями и поведением. Для digital-проектов сегменты будут нужны, чтобы определиться, какие функции нужны в продукте и как расставить приоритеты разработки.

> Еще раз: сегментация нам нужна, чтобы понять, как поведение пользователя влияет на продукт. Поэтому **дифференцирующим критерием** (чем же вы, елки-палки, на самом деле различаетесь) **предпочтительнее брать поведенческий фактор**. Как разные пользователи по-разному пользуются вашим продуктом и какие разные задачи они решают.

Жестких правил и критериев для сегментации нет. Если у вас ведется подробная база пользователей и обращений, можно попробовать загнать ее в систему анализа бигдаты (вроде RapidMiner), поиграть с ней в кластерный анализ и попробовать выделить сегменты.

Довольно часто для сегментации используют критерии: география, демографические признаки (пол, возраст, семейное положение, доход, образование, профессия, религия, национальность и т. д.), психографию (хобби, образ жизни и т. д.), платежеспособность (LTV – Lifetime Value, сколько денег приносит клиент за все время сотрудничества).

Это все, конечно, замечательно. Особенно для повышения среднего чека. Но для разработки минимально жизнеспособного digital-проекта – почти бессмысленно. Ищите дифференцирующий критерий в поведении.

Периодически нужно пересматривать сегментацию, искать новые сегменты и уточнять имеющиеся.

4.2.3.1. Анализ текущего поведения пользователей

Допустим, у вас задача – редизайн существующего проекта. Перед стартом работ нужно собрать срез статистики. Нам понадобятся доступы к счетчикам типа Яндекс Метрики и Google Analytics. Статистика нужна примерно за полгода-год.

Если на старом сайте есть хоть какой-то трафик – изучите его. Попытайтесь понять, кто реально ходит на текущий сайт, является фактической онлайн-аудиторией бизнеса. И в чем возможные причины проблем, на которые жалуется клиент.

Могут всплыть противоречия. Например, заказчик говорит: «Наш журнал читают представители всех возрастных и социальных групп». Типичная, кстати, галлюцинация. А по метрикам видно, что добрая часть визитов от женщин сильно за 50.

На это нужно обратить внимание. Обозначить задачу: сделать будущий сайт привлекательным, в том числе для ожидаемой аудитории – то есть, сделать так, чтобы тот журнал захотели прочитать, например, 16-летние хипстеры. И тут уже не редизайном попахивает. А сменой концепции журнала. «Спасибо» вам за такие выводы не скажут. Но цели могут скорректировать.

Тут же стоит обратить внимание на:

1. Точки входа.
2. Какие страницы пользуются популярностью. Какие – никому не нужны.
3. Показатели отказов.
4. Читаемость. Время сессии.
5. Поведенческие факторы. Кликабельность тех или иных элементов.
6. Реальную демографию.
7. Интересы пользователей.
8. Поисковые запросы (их можно взять за основу семантического ядра).
9. Какие события настроены в аналитике.

Безусловно, это нужно учесть при редизайне. Например, сохранить адреса популярных страниц (или, как минимум, поставить с них ***301 редирект***).

Как правило, мы делаем такой срез по статистике с краткими выводами и вставляем его в отдельную вкладку Агрегации требований. Скриншотить все экраны Google Analytics смысла нет. Акцент делаем на те моменты, которые **влияют на структуру** будущего проекта.

4.2.3.2. Аватары. Анализ целевых персон

Анализ целевых персон (он же – метод Аватаров) – ключевая вкладка Агрегации требований. Метод перекочевал в digital из классического маркетинга.

Суть: нужно выделить несколько (три-семь) целевых групп пользователей продукта и одушевить их. Сделать реальных людей: с фотографией, именем, мотивами и проблемами. Кто будет (должен быть) аудиторией нашего сайта или приложения, чем эти люди живут, какие имеют потребности и боли и что мы можем сделать, чтобы эти боли разрешались в нашем проекте.

> Аватар – это вымышленный персонаж, в котором отражены основные характеристики целевой аудитории.

4.2.3.3. Как выделить группы

Все люди «как бы» разные. И у каждого проекта своя, «как бы» неповторимая аудитория. Ключевое слово – «**как бы»**. Анализируя частные случаи, можно выделить закономерности и общие черты, которые позволят объединить людей в крупные группы.

Группы нужно выделять по значимым для проекта особенностям, и демографические характеристики почти никогда этими особенностями не являются, отталкиваться нужно от поведенческих и социальных моделей.

Нет смысла выделять группу «женщины средних лет», поскольку она одинаково подходит и для онлайн-магазина Ozon, и для сайта стоматологической клиники. То есть, не ответит на вопросы о потребностях в отношении вашего проекта, если он, к примеру, посвящен продаже

швейных машин. Зато в аудитории проекта помогут разобраться такие персоны как «Новички» и «Вышивальщицы», так как у них очень разные запросы в отношении продукта, а значит, и схема действий при его выборе на сайте.

4.2.3.4. Гипотезы сегментов. Эксперты

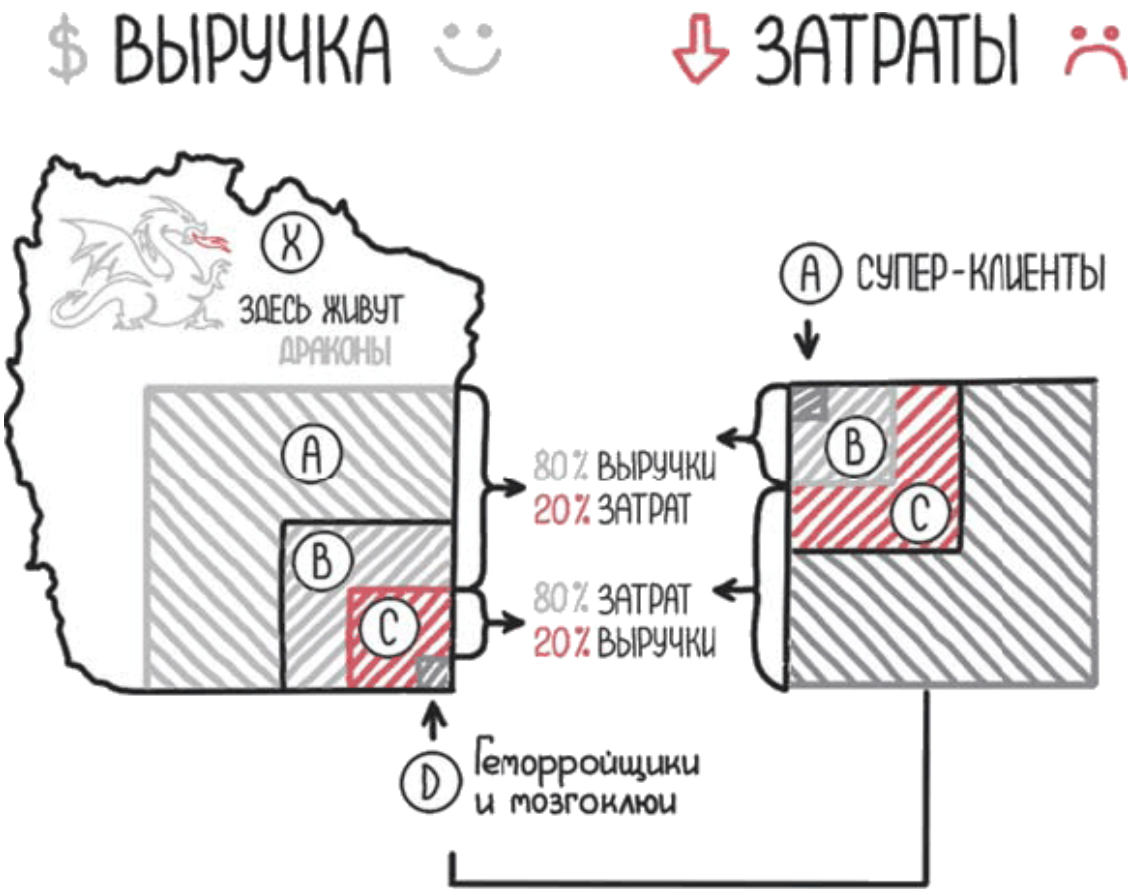
Итак. Сегмент классифицируется в первую очередь поведением. И в любом случае сегментация – это гипотеза. Однако как составить такие гипотезы, если у вас нет никаких идей на этот счет?

Поговорите с экспертами. В заказной разработке это сам заказчик. Его продажники, саппорт, доставка, стейкхолдеры. Для продуктовой разработки мы можем использовать JTBD-подход (об этом ниже, в параграфе 4.3.7), чтобы сформировать и протестировать гипотезы.

4.2.3.5. ABCDX-сегментация

Мы всегда готовы прийти к вам на выручку.
Была бы выручка!

Как в заказной, так и в продуктовой разработке (да и, наверное, не только в разработке) мы можем сегментировать клиентов по выручке и затратам. Ребята из ФРИИ назвали это ABCDX-сегментация.



▶ A-сегмент. Супер-клиенты. Замечательно платят, быстро покупают и принимают решение. С ними просто по кайфу работать и решать вопросы ♥

▶ B-сегмент. У них есть возражения. Чего-то не хватает. Но в целом все ОК. Средний чек – высокий. Цикл сделки – относительно короткий. Платят – регулярно.

▶ C-сегмент. Продавать им можно сто лет. Продажники в мыле. Саппорт в ужасе. Бухтеть они будут по любому поводу. Что бы вы ни делали. Все не то. Все не так. Но иногда покупают. В заказной разработке на поддержке – один такой проект весит как пять нормальных. Разработка завалена невменяемыми тикетами. Заработаете на них вы три копейки. А отвалятся они запросто. И останутся недовольными при любом раскладе. В продуктовой разработке, если идти у них на поводу – еще и продукт испортите, и A/B-сегменты потеряете.

▶ D-сегмент. Просто выносят мозг. И ничего не покупают. И, кстати, слава богу! Если они что-то купят – тут вы проклянете все. Проджект-менеджеры сгорают как свечки на таком проекте. Программисты тихо матерятся и разбегаются как тараканы. Дизайнеры спиваются.

▶ X-сегмент. Потенциально ваш сегмент. Такой же, как и супер-клиенты из сегмента A. Но по каким-то причинам вы до них не можете добраться. Продукт не очень подходит, продажник рожей не вышел и т. д. Это наша неизведанная территория. Там живут драконы. Направляем туда спецназ и потихоньку отвоевываем новые земли.

A+B-сегменты приносят 80 % выручки при 20 % затрат. Именно на них нужно фокусировать сильные команды, ставить хороших менеджеров и сэйлзов. Сейлзам надо доплачивать за таких клиентов и давать медали.

C+D-сегменты приносят 20 % выручки, но сжигают 80 % сил.

D-сегмент должен, по умолчанию, гореть в аду. Ваш звонок очень важен для нас, пи-пи-пи.

C-сегмент сложно вычислить в моменте. По крайней мере, я тут часто ошибаюсь. Нужна статистика за 3–6–12 месяцев. К сожалению, в заказной разработке этого времени достаточно, чтобы спалить команду. Именно это делает работу с C-сегментом крайне нерентабельной в долгосрочной перспективе. В продуктовой разработке C-сегмент еще можно попробовать автоматизировать. Главное, не берите от них ничего в бэклог – испортите продукт. Но в заказной разработке автоматизировать ничего не получится. Либо страдать с пользой, либо сматывать удочки. Ничего личного, просто бизнес.

Итак. Один из способов сегментации – ABCDX, где мы учитываем выручку и затраты. И концентрируемся на тех сегментах, с кем легко и по кайфу делать классные вещи.

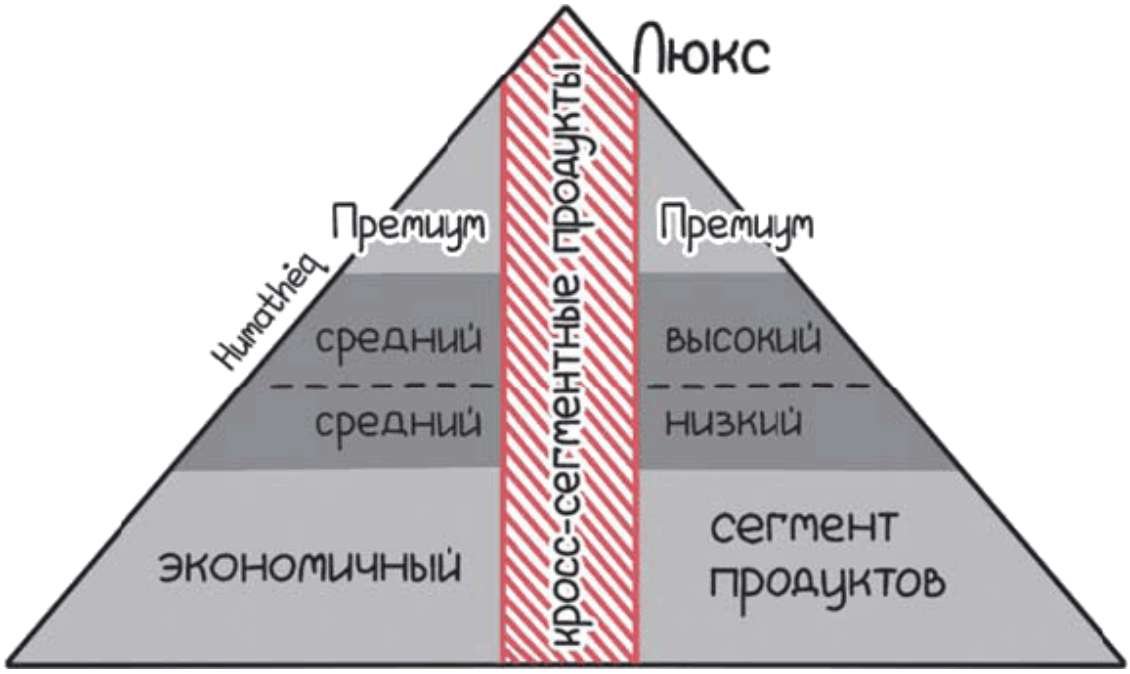
4.2.3.6. Сегментация по цене. Как менять цену продукта

Это самая очевидная сегментация. Оптовики-розница. Премиум-дешман. Чем выше сегмент, тем, как правило, он меньше.

В экономичном сегменте все решает цена.

В сегменте чуть выше экономичного (Low middle) сидят самые геморройные клиенты (помните сегменты C и D?). Задают много вопросов, вечно всем недовольны, меняют поставщиков как перчатки и ничего не покупают. А если купят – начнется ад! Но если вы научитесь с такими работать и зарабатывать – это прорыв.

High middle – выше среднего. Прекрасный сегмент. Цена не так чувствительна, а выбор идет между средним и премиальным аналогом. Им важен бренд.



Premium. Эти ребята будут выбирать скорее долговечные и надежные продукты и готовы за это доплачивать.

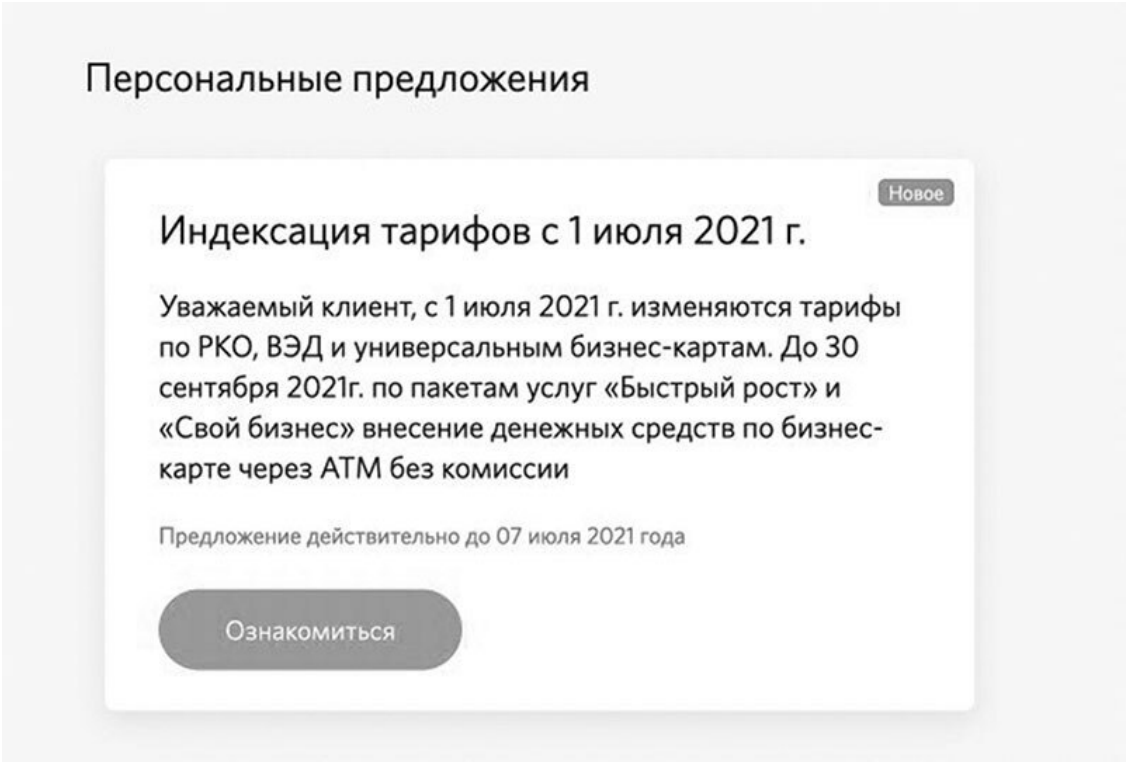
Люкс или Luxury. Роскошь. Большинство людей не может себе это позволить. Вертолет «Еврокоптер» или автомобиль Maserati, например. Бренд, эксклюзивность, высокая цена, маленькая тиражность. Это то, что важно для этого сегмента.

Однако иногда случается чудо, и продукт используют все сегменты. От экономичного до премиального. В этом случае использовать ценовую сегментацию сложно (только опциями). Пример: популярные операционные системы, социальные сети, iPhone. Пожалуй, нет какого-то способа специально «попасть» в кросс-сегментный продукт, но это потенциально самый большой рынок.

Вообще решение о ценовой политике продукта – одно из самых ключевых. Будете вы продавать мало, но за дорого? Использовать ***фримиум-модель*** или настаивать на продаже сразу и за дорого? Подписка или единоразовый платеж? Все это очень сильно влияет на продукт и попадание в сегмент.

Менять ценовые политики после запуска – довольно сложно. Старые пользователи очень чувствительны к таким изменениям. Как минимум, вы должны сохранить для них цены на достаточное время. Посмотрите на эту прелесть:

Персональные, мать его, предложения. Внутри – декларативное увеличение тарифов. Милота какая, видимо, нашли раздел, который никто не читает.



Не делайте так в своих продуктах. Да, бывает не угадали с юнит-экономикой (о ней поговорим позже, в параграфе 4.4). Или пришло время. Но не надо прятать такие вещи. Говорите открыто с пользователями, как есть. Кто-то отвалится, кто-то будет нечувствителен к повышению. Кто-то все поймет.

Старайтесь сохранить на какое-то разумное время (месяц/квартал/полгода/год) условия и тарифы для старых пользователей. Ведите себя этично в своих продуктах. Аминь.

Итак. Ценовая политика продукта – один из самых важных параметров. Помните об этом!

4.2.3.7. B2C, B2B, B2G-персоны

Если с B2C (business-to-client – бизнес для клиентов) персонами все более-менее понятно, для B2B (business-to-business – бизнес для бизнеса) и B2G (business-to-government – бизнес для государства) есть одна типовая ошибка: обобщение.

Конечным программным продуктом будут пользоваться не компании. А конкретные люди в этих компаниях. Бухгалтер. Менеджер по закупкам. Директор по логистике. И так далее.

Строить персоны здесь нужно исходя из **роли** пользователя, бизнес-процессов или структуры **информационных потоков** предприятия-клиента. Как принимаются решения, кто на них влияет, какая информация нужна на каждом из этапов.

В моделировании такой персоны следует учитывать следующие характеристики:

- ▶ личные качества;
- ▶ должностные обязанности;
- ▶ стремления;
- ▶ недостатки.

4.2.3.8. Гиперсегментация

Болезнь редкая, но встречается. Заказчик просит выделить десять и более целевых персон. И всех их рассмотреть детально. О чем нам это говорит?

1. Менеджер на стороне клиента мог начитаться книг, проникся идеей сегментирования и хочет сегментировать все и вся, или просто хочет получить как можно более толстую аналитику. Чтобы аналитик наработался. Плохо, потому что требует кратного увеличения ресурсов.

2. Нет вменяемого квалифицирующего фактора, который бы действительно выделял сегменты по поведенческим паттернам. Разные персоны могут отличаться полом, демографией и т. д., но это толком ни на что не влияет. В продукте они будут вести себя одинаково. И нет смысла их расписывать. Это только все запутает и создаст белый шум. Аналитикой будет невозможно пользоваться.

3. Если продукт такой, что там действительно 20 принципиально разных квалифицирующих факторов – скорее всего, основатель очень сильно расфокусирован. Делает все для всех. В итоге получится гипер-сложная, никому не нужная ерунда.

Начните с пяти-семи сегментов. Но проработайте их хорошо. В дальнейшем, после запуска продукта и при работе с рекламными компаниями, вы можете досегментировать рынок, например, по каналам. И усовершенствовать продукт или рекламные кампании. Однако на старте это только все запутает и сожжет ресурсы.

Итак. Гиперсегментация на старте разработки – это вредная затея. Рассматривайте пять-семь персон – этого достаточно.

4.2.3.9. Бредосегментация

Как-то в офис к нам пришли три представителя заказчика. Отец – собственник завода сельхозтехники. Сын – исполнительный директор этого же завода. И маркетолог. Отец, сын и маркетолог – обычное, кстати, дело. Аминь.

Реальная аналитика целевых персон их мало интересовала. У них было свое видение проекта. Причем, у всех – разное, и этот конфликт мы вскрыли и всех помирили. Но факт остается фактом – в заказной разработке обычное дело, когда влияние целевых персон из аналитики на продукт будет ничтожным по сравнению с влиянием стейкхолдеров. Мы поговорим еще про это подробно в главе про галлюцинации основателей на аналитике (§ 4.2.6). Однако нельзя отбрасывать тот факт, что в заказной разработке приходится вскрывать и решать конфликты стейкхолдеров и увязывать их мнение с реальностью. Это сложно. Но нужно.

Итак. В заказной разработке нужно принимать во внимание не только целевых пользователей продукта, но и целевые персоны заказчика. Мы их рассматривали во вкладке «Видение». Однако если между интересами стейкхолдеров и пользователями есть потенциальный конфликт – нужно его решать.

4.2.3.10. Как описать персону

Так, чтобы в нее поверили.

Если назвать персонажа Вася Пупкин и добавить дурацкую фотку со стока, в Васю никто не поверит. Как и в выводы, сделанные такой персоной. То есть, дурацкая фотография или нелепое имя напрочь убивают доверие к аналитике. Какая бы качественная работа ни была дальше.

Подбирайте характерные для реальной группы фотографии. Придумывайте нормальные человеческие имена. И отмечайте в описании детали, которые придают персонажу характер. Что-то, что позволит слушателю узнать в нем реального человека, с которым приходилось работать.

В идеале стоит проиллюстрировать сказанное реальными скриншотами с форумов, социальных сетей и отзовиков. Выдернуть настоящих потребителей. Например, из CRM-системы клиента. Это не всегда возможно. Но попытаться стоит.

Описывая характер, ограничения и подход к принятию решений, говорите именно с точки зрения взаимодействия с будущим программным продуктом. То есть, если наш сайт посвящен продаже товаров на B2B-рынке, для нас неважно, что Иван – крепкий семьянин. Но важен его опыт в закупках и глубокое знание рынка.

Персоны удобно хранить в Google-таблицах или интеллектуальных картах.

Целевая персона РостСельМаш в XMind

Пример.

Для компании в музыкальной индустрии (скажем, звукозаписывающей студии) можно смоделировать таких персонажей:

> *1. Михаил, 45 лет, преподает игру на скрипке в музыкальном колледже, пользуется интернетом менее года. Выходит в сеть исключительно с домашнего компьютера через широкополосное соединение. Никогда не совершал покупки посредством интернета, предпочитает делать заказы по телефону.*
>
> *2. Анна, 29 лет, исполнительный директор. Активный пользователь интернета в течение последних пяти лет, для выхода в сеть использует все, что есть под рукой: MacBook, iPad или свой iPhone.*

Таким образом, потенциальными клиентами компании являются совершенно разные типы людей с абсолютно разными потребностями.

В некоторых случаях смоделированные персоны можно применять и в таком виде. Однако полностью проработанные персонажи имеют более подробное описание с указанием тех целей, ради достижения которых они зайдут на сайт компании.

Оживить персонажей помогают небольшие истории:

> *«Николай всю жизнь в сельском хозяйстве. На ферме работал еще при совхозе, теперь предприятие частное, а Николай на том же месте. Знает свое дело, но в последние годы с развитием хозяйства старая советская техника сменяется новыми машинами, сложность которых и роскошный внешний вид пугают».*
>
> *«У Андрея советский тип воспитания, его отец, а теперь и он сам считает зазорным нанимать кого-то для ремонтных работ, это не только затратно, но и попросту недостойно. Семья расширяется, получены ключи от ипотечной трешки без ремонта. Андрей закупает материалы небольшими партиями, по шагам под каждую задачу. Так легче для семейного бюджета, да и движется процесс небыстро из-за работы и времени в пути до квартиры».*

4.2.3.11. Верхнеуровневые мотивы

Зачем пользователь зашел на ваш сайт или открыл мобильное приложение (или что вы там разрабатываете?). Какую задачу он на самом деле решает? Вот это и есть – мотив. Чуть позже мы посмотрим на концепции CJM и Jobs To Be Done, чтобы разбивать мотивы на шаги.

То есть, для нас важен мотив некого Андрея найти магазин для выгодных и удобных онлайн закупок стройматериалов, а не то, что он хочет сделать ремонт в своей квартире. Это, конечно, тоже важно и легло в основу истории. Но мы моделируем персону именно с точки зрения взаимодействия с сайтом. А значит, формулируем мотив посещения сайта.

На этом этапе обдумайте, как пользователь мог узнать о сайте. Какое действие на сайте будет ключевым для данного пользователя. И как в целом будет строиться его путь.

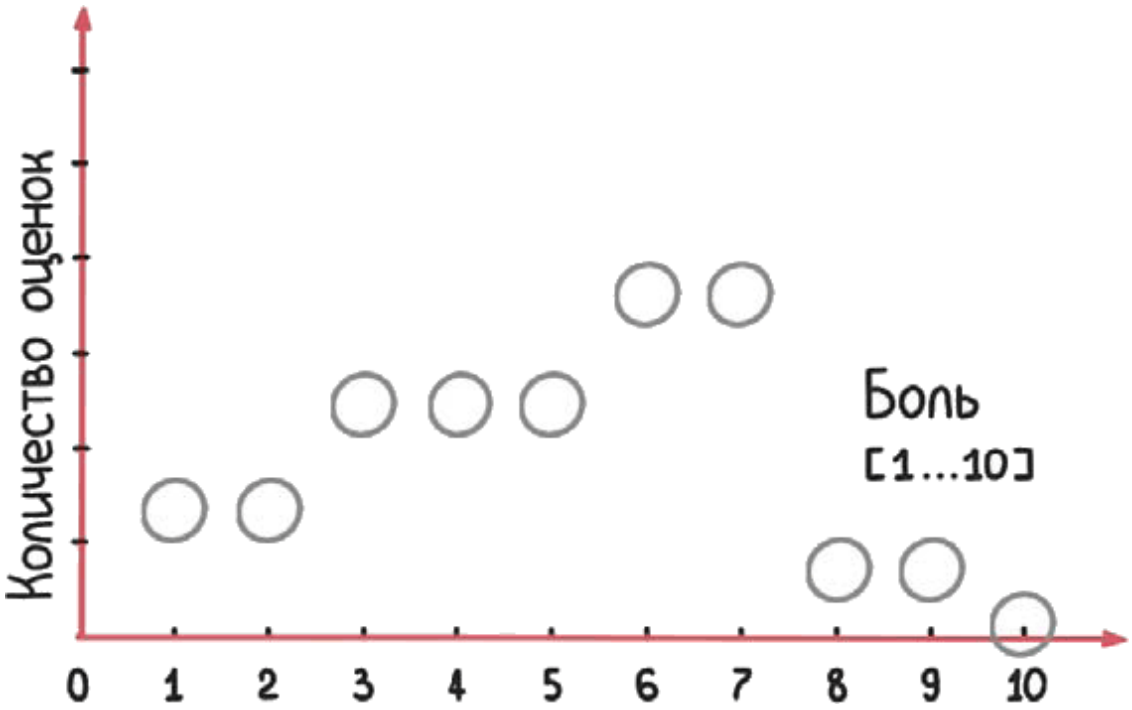
4.2.3.12. Боль. Сила боли

Нет проблемы – нет продажи.

По каждой целевой персоне нам нужно понять, что у нее «**болит»**! Именно из этого вытекает набор функций, который мы должны предусмотреть на нашем проекте.

Например. На B2B-сайте мне как менеджеру по продажам важно параллельно работать с несколькими корзинами, поскольку я формирую и уточняю одновременно несколько заказов в течение дня. Для B2C такие опции, наоборот, навредят – получим слишком сложный интерфейс и брошенные корзины. Если в проекте есть обе персоны и они равнозначны – значит, нужно предусмотреть выбор профиля. Ух!

Под болями мы подразумеваем препятствия, переживания, проблемы, барьеры, с которыми сталкивается человек на пути к своей цели. Нам необходимо устранить эти барьеры. Формулируя боль, сразу отвечайте себе на вопрос: каким образом она будет закрываться в нашем программном продукте.



Для определения приоритетов разработки нас будет интересовать два параметра: как часто «болит» и насколько сильно. И то и другое помогает выявить Проблемное интервью CustDev (см. § 4.3.8).

Как ни странно, даже в медицине, боль – субъективный фактор. Объективного «болиметра» не придумали. Кто-то от прививки в обморок падает. А кому-то лом пролетел через голову, в результате чего у него слегка испортился характер. Многие люди склонны драматизировать силу боли. Особенно свежих событий или событий, на которых они сфокусированы прямо сейчас. Отсекайте такие «выбросы» относительно средних величин.

К примеру, в офис заказчика вашего программного продукта пришел дедушка и устроил скандал, что «на сайте нет версии для слабовидящих». Он чего-то там не разглядел и не туда нажал. Откомпостировал мозг всем по дороге. Досталось даже генеральному директору. Его все очень хорошо запомнили. Однако, если это первая такая жалоба за 10 лет – бежать и экстренно реагировать не стоит. Может, дедушка со справкой, и он так свою значимость миру показывает?

Итак. В Агрегации требований для каждой персоны мы фиксируем боли. Если есть возможность собрать статистику по частоте и силе – это хорошо.

Не формулируйте боли в виде вопроса. Почувствуйте разницу:

> *«Не испачкаю ли я руки, когда буду есть чебурек?»*
>
> *«Чебуреки жирные! С них капает сок. Их невозможно есть так, чтобы ничего не текло во все стороны. Каждый раз, когда я покупаю чебурек, я боюсь, что испачкаю одежду. Жир очень плохо отстирывается. А уж ходить целый день в грязной одежде – это мерзко. Что обо мне подумают люди? Особенно вон та мымра из соседнего отдела! Мерзко, на десять из десяти!».*

То есть, боль не в том, что чебурек капает. А в том, что человеку стремно запачкать одежду – что подумают люди! И это может быть главным мотивом отказа от покупки.

4.2.4. CJM. Карта путешествия пользователя (Customer Journey Map)

В заказной разработке клиенты чаще и чаще стали спрашивать Customer Journey Map – карту путешествия клиента. Это наглядный инструмент для анализа проблемных точек, которые мы создаем пользователям своим сервисом. Инструмент старый. Пришел в digital из классического маркетинга. И очень вероятно, что руководитель проектов с ним рано или поздно столкнется. В чем идея?

В идеале для каждой целевой персоны (аватара) нам нужно создать и проанализировать его путь от того момента, как у него возникло смутное подозрение, что есть какая-то проблема, которую нужно решать, до того момента, как он становится приверженцем бренда и готов его рекомендовать. Впрочем, многие так далеко не думают – ограничиваются первой покупкой. Но мы будем настойчивы!

Итак. Наш покупатель проходит через 6 крупных, условных фаз:

1. Awareness. Осознание. Осведомленность. Что-то его смутно мучает, но что именно – он не очень понимает. Например, человеку хреновенько на работе. Он не так жжет, как раньше, и ничего не успевает. На этой фазе он начинает гуглить, думать, что с ним не так, читать статьи про эффективность и т. д.

2. Consideration. Изучение, осведомление. Например, человек понял, что ему нужен тайм-менеджмент, и именно в самоорганизации его проблема. Теперь он ищет инструмент, в котором бы мог структурировать все свои дела. Наивный!

3. Decision. Выбор и принятие решения. Решение для человека в целом понятно. Хочется выбрать что-то конкретное. Для тайм-менеджмента это может быть выбор между блокнотом,

календарем или программами ведения списка дел и проектов. Он читает рекомендации, смотрит, экспериментирует, проверяет и пробует сам.

4. Purchasing. Покупка. Наконец, все понятно. Выбор среди множества вариантов сделан. Победит тот продукт, который будет проще купить. Будут ли вежливы продавцы? Нужно ли будет куда-то идти или с кем-то разговаривать? Все это влияет на легкость покупки и пользовательский опыт.

5. Loyalty. Лояльность. Появляется первый пользовательский опыт использования продукта. С офлайн-продуктами это может быть как опыт доставки и монтажа, так и опыт использования самого продукта. В случае продуктов, которые пользователь покупает регулярно – он может остаться с этим поставщиком, стать лояльным. А может в ужасе сбежать в поисках другого решения.

6. Advocacy. Адвокат бренда. Наконец, если все очень хорошо – человек может рекомендовать бренд.

CJM обычно представляет собой табличку. По горизонтали мы выписываем шаги или фазы, через которые проходит пользователь. По вертикали – те параметры, которые хотим отслеживать. Как правило, это:

1. Цели и ожидания пользователя на каждом этапе. Его активности.

2. Вопросы, которые задает себе покупатель.

3. Мысли и чувства пользователя в этот момент.

4. Точки контакта. С чем приходится взаимодействовать пользователю. С сайтом, Инстаграмом[2] компании, службой поддержки, электронным письмом и т. д. Тут есть одна тонкость, про нее напишу отдельно.

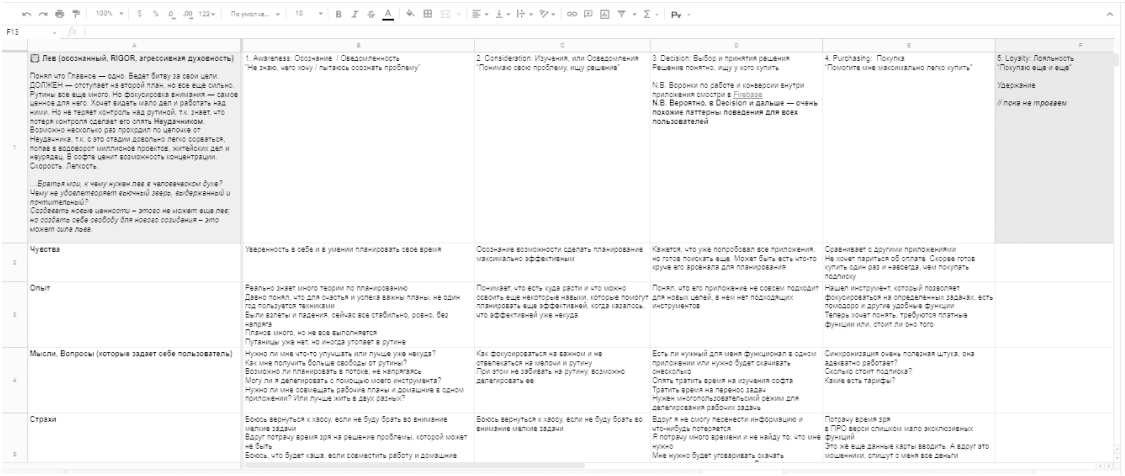
5. Уровень счастья / удовлетворенности пользователя на каждом из этапов.

6. Ожидания. Только не в духе Аршавина: «Ваши ожидания – ваши проблемы».

7. Барьеры. Что мешает, какие есть сложности. Провести анализ глазами клиента, проанализировать отзывы и обратную связь, пожелания и жалобы клиентов.

8. Решения. Способы преодоления барьеров. Штормим, думаем, как облегчить жизнь.

9. Дополнительные параметры. Например, ответственные за этап, перечень ключевых запросов и т. д.

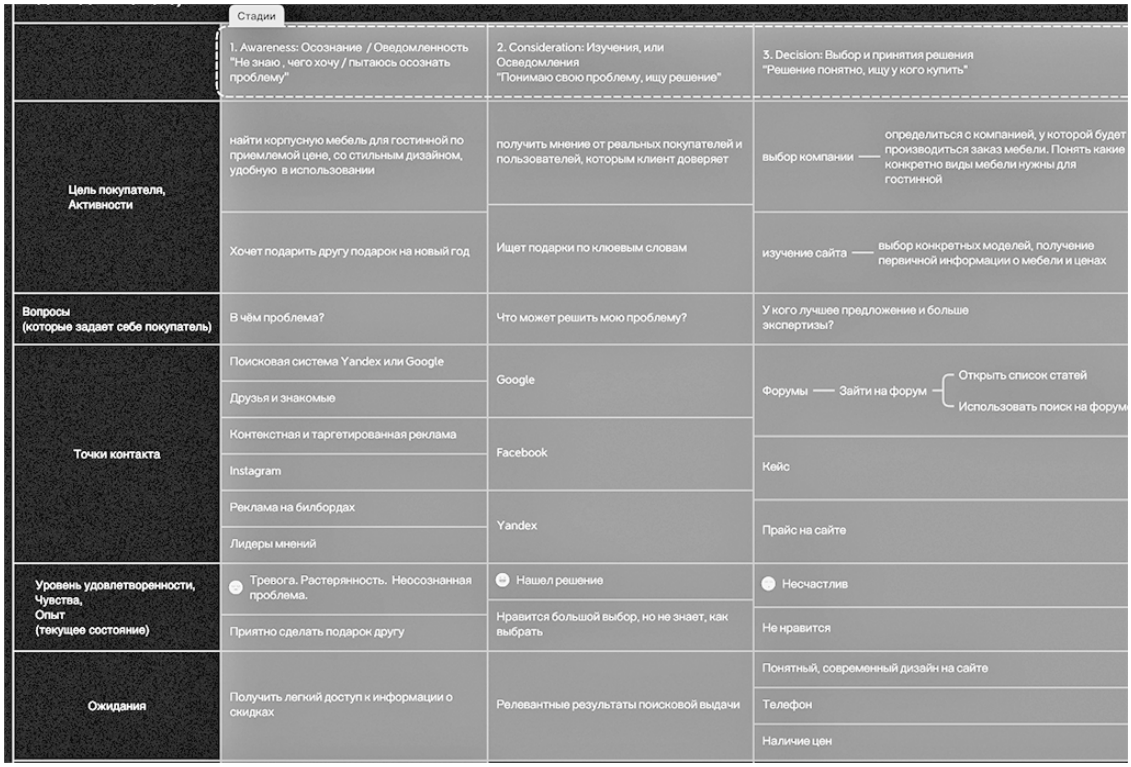


	A	B	C	D	E	F
1	Лев (осознанный, RIGOR, агрессивная духовность) Понял что Главное — одно. Ведет битву за свои цели. ДОПЖЕН — отступает на второй план, но все еще сильно. Рутины все еще много. Но фокусировка внимания — самое ценное для него. Хочет видеть мало дел и работать над ними. Но не терпит контроль над рутиной, т.к. знает, что потеря контроля сделает его опять Неудачником. Возможно несколько раз проходил по цепочке от Неудачника, т.к. с это стадии довольно легко сорваться, попав в водоворот миллионов проектов, житейских дел и мелочей. В софте ценит возможность концентрации. Скорость. Легкость. ...Братья мои, к чему нужен лев с человеческом духе? Чему не убовлетворяет вьючный зверь, выбережанный и почтительный? Сосздавать новые ценности – этого не может еще лев: но создать себе свободу для нового созидания – это может сила льва.	1. Awareness. Осознание / Свершенность "Не знаю, чего хочу / пытаюсь осознать проблему"	2. Consideration. Изучение, или Осведомление "Понимаю свою проблему, ищу решения"	3. Decision. Выбор и принятия решения Решение понятно, ищу у кого купить N.B. Воронки по работе и конверсии внутри приложения смотри в Firebase N.B. Вероятно, в Decision и дальше — очень похожие паттерны поведения для всех пользователей	4. Purchasing. Покупка "Помогите мне максимально легко купить"	5. Loyalty. Лояльность "Покупаю еще и еще" Удержание // пока не трогаем
2	Чувства	Уверенность в себе и в умении планировать свое время	Осознание возможности сделать планирование максимально эффективным	Кажется, что уже попробовал все приложения, но готов поискать еще. Может быть есть что-то круче его аналога для планирования	Сравнивает с другими приложениями Не хочет париться об оплате. Скорее готов купить один раз и навсегда, чем покупать подписку	
3	Опыт	Реально знает много теории по планированию Давно понял, что для счастья и успеха важны планы, не один год пользуется техниками Были взлеты и падения, сейчас все стабильно, ровно, без напряга Планов много, но не все выполняется Путаницы уже нет, но иногда уплывет в рутину	Понимает, что есть куда расти и что можно освоить еще некоторые навыки, которые помогут планировать еще эффективней, когда захочось, что эффективней уже некуда	Понял, что его приложение не совсем подходит для новых целей, в нем нет подходящих инструментов	Нашел инструмент, который позволяет фокусироваться на определенных задачах, есть помодоро и другие удобные функции Теперь хочет понять, требуются платные функции или, стоит ли оно того	
4	Мысли. Вопросы (которые задает себе пользователь)	Нужно ли мне что-то улучшать или лучше уже некуда? Как мне получить больше свободы от рутины? Возможно ли планировать в потоке, не напрягаясь Могу ли я делегировать с помощью чего-инструмента? Нужно ли мне совмещать рабочие планы и домашние в одном приложении? Или лучше жить в двух разных?	Как фокусироваться на важном и не отвлекаться на мелочи и рутину При этом не забивать на рутину, возможно делегировать ее	Есть ли нужный для меня функционал в одном приложении или нужно будет скачивать очеколько Опять тратить время на изучения софта Тратить время на перенос задач нужен многопользовательский режим для делегирования рабочих задач Вдруг я не смогу перенести информацию и что-нибудь потеряется Я потрачу много времени и не найду то, что мне нужно Мне нужно будет уговаривать скачать	Синхронизация очень полезная штука, она адекватно работает? Сколько стоит подписка? Какие есть тарифы?	
5	Страхи	Боюсь вернуться к хаосу, если не буду брать во внимание мелкие задачи Вдруг потрачу время зря на решение проблемы, которой может не быть Боюсь, что будет каша, если совместить работу и домашние	Боюсь вернуться к хаосу, если не буду брать во внимание мелкие задачи		Потрачу время зря в ПРО версии слишком мало эксклюзивных функций Это же еще данные карты вводить. А вдруг это мошенники, спишут с меня все деньги	

CJM по одному из типов пользователей приложения SingularityApp. Фрагмент из Google Docs.

[2] Социальные сети Facebook и Instagram запрещены на территории Российской Федерации на основании осуществления экстремистской деятельности.

Проще всего построить такую таблицу в Google Docs. Получается практичный инструмент, в который нужно регулярно заглядывать и обновлять.



CJM. Xmind. Фрагмент

Однако CJM часто красиво визуализируют, рисуют чуть ли не в Фотошопе, добавляют графики. И из рабочего инструмента CJM вырождается в маркетинговую пыль в глаза для красивых презентаций. Это первый минус CJM: из инструмента он превращается в имитацию инструмента. Хотя, если важны красивые презентации, CJM производит впечатление.

Второй минус – точки контакта. Их может быть очень много на каждом этапе. Сценарий перемещения по этим точкам может быть нелинейным и крайне запутанным. И содержать агентов влияния (вроде жены, мамы, подружки), общение с оператором, пересылку ссылок на избранное, корзину или оплату.

По-хорошему, нам нужно развернуть весь путь пользователя в цепочку шагов. Но это будет безумно широкая таблица, которой будет невозможно пользоваться. А как учесть разные маршруты и возвраты на предыдущие шаги – вообще не очень понятно. Поэтому ограничиваются наиболее типовыми маршрутами.

Третий минус – трудоемкость. По-хорошему, нам нужно строить такие карты для каждой целевой персоны и для каждого маршрута. Но это недели, если не месяцы, работы аналитика. Дорого и непонятно зачем. Если же рассмотреть какой-то общий сценарий – карта получается слишком неконкретная.

В заказной разработке обычно мы концентрируемся на той части CJM, которая касается взаимодействия пользователя с сайтом. И рассматриваем лишь самые важные маршруты для самых частых персон. В такой форме это оправданно и позволяет улучшать пользовательский опыт в программном продукте.

Итак. CJM – инструмент для анализа пользовательского опыта на каждом шаге и точке контакта при взаимодействии пользователя с брендом. Инструмент неплох для заказной разработки или для анализа готового программного продукта. В продуктовой разработке лучше использовать фреймворк RAT+JTBD (об этих аббревиатурах – в параграфах 4.3.3 и 4.3.7). Важно найти баланс между трудоемкостью и практичностью. А еще CJM любят вставлять в красивые презентации. Если это произошло – значит, CJM уже мертв.

4.2.5. Решения

Аналитика никому не нужна.
Нужны выводы из аналитики.

Итак. Мы определились с сегментами. Описали целевые персоны. Выявили их боли и мотивы.

Все это было проделано, чтобы понять, какие вещи (функции, разделы, экраны) мы должны предусмотреть в будущем проекте для того, чтобы эти боли закрыть.

Какими должны быть предложенные решения?

В первую очередь, не случайными, а закрывающими выявленные выше потребности персоны. Например: мы делаем отзывы со ссылками на соцсети не просто так, а потому что выяснили, что наш персонаж Леночка не решается покупать косметику без отзывов, но не так наивна, чтобы верить анонимным отзывам на сайте интернет-магазина.

Важно, чтобы решения отвечали уровню проекта. Если заказчик хочет простой корпоративный сайт, можно предложить интеграцию с чем-то диковинным, калькулятор с 3D-моделями. Но аккуратно, с оговоркой, что это идея на развитие. Иначе у заказчика останется неприятный осадок, что его не слышат либо пытаются развести на дополнительные опции.

Как правило, мы описываем предлагаемые решения сначала рядом с каждым аватаром, чтобы четко прослеживалась связь решений, болей и мотивов. Для каждой персоны решения размечаем цветными рамками (цветовое кодирование: одна персона – один цвет). Используем иконки для разметки этапов, акцентов или вопросов, которые нужно разобрать.

И ниже, в этой же вкладке Агрегации требований делаем сводную таблицу всех функций, также размеченную цветами. Не только чтобы было легче воспринять будущий проект, но и чтобы можно было отследить источник боли – то, ради чего эта функция была придумана. Пока не увлекаемся структурированием. На этом этапе нам важен сам список. Компоновать по разделам и продумывать перелинковки будем чуть позднее.

Описывайте решения достаточно подробно, чтобы было ясно, в чем фишка и как она будет работать на решение боли.

Итак. Из болей и мотивов следуют решения (но ими не ограничиваются). Предложите решение на каждую боль. И далее сведите все решения в общий список.

4.2.6. Анализ сайтов конкурентов

> *Кто знает врага и знает себя, не окажется в опасности и в ста сражениях. Тот, кто не знает врага, но знает себя, будет то побеждать, то проигрывать. Тот, кто не знает ни врага, ни себя, неизбежно будет разбит в каждом сражении.*
> ***Сунь-Цзы. Искусство войны***

Нам нужно посмотреть и изучить проекты конкурентов, чтобы понять:

1. что у них есть прикольного, что имело бы смысл сделать у себя;
2. в чем у них есть лажа, и каких ошибок следует избегать.

Тут важно отметить и для себя, и для заказчика, что это не анализ рынка с попыткой разобраться, кто и почему зарабатывает на этом рынке больше, и не анализ маркетинговых кампаний – мы смотрим на конкуренцию только с позиции сайтов/мобильных приложений. Выявляем позитивные и негативные тренды, отмечаем особо интересные фишки.

Первое, что нужно рассмотреть – проекты, на которые сослался сам заказчик. В большинстве случаев заказчик знает конкурентов. Однако с тем же успехом имеет ряд галлюцинаций по этому вопросу. Вот ТОП-3:

1. «У нас нет конкурентов». Не верьте. Либо что-то не договаривает, либо что-то не понимает. Логика там примерно такая: «Мы делаем Инстаграм для котиков со встроенным маркетплейсом. Ни у кого больше нет Инстаграма для котов со встроенным маркетплейсом. Поэтому у нас нет конкурентов!» Логично? Хоть глупо, но логично! Посмотрите, как люди решают ту же самую проблему, которую решает компания-заказчик.

2. «Наши главные конкуренты-враги – компания "Копыта и Рога" с соседней улицы! Они даже логотип сделали почти как у нас. С рогами и копытами». Это обычно высказывается на эмоциях. Заказчика очень бесит какой-то местный, локальный конкурент. И он считает его главным врагом. На самом деле, в интернете он будет конкурировать с какой-нибудь Икеей, Wildberries или Озоном. Но думать про это – бздляво. А вот «Копыта и Рога» с соседней улицы – в самый раз.

Анализируйте не только конкурентов по отрасли, но и по поисковой выдаче, и по рекламным кампаниям, и по позициям в популярных маркетплейсах. Смотрите и прямую, и косвенную конкуренцию.

3. «Все сайты наших конкурентов – ужас и кошмар. Там нечего смотреть». Редко, но бывает. Особенно в B2B. Тогда стоит посмотреть смежные отрасли:

a. Как аналогичные вещи решаются на зарубежных сайтах?

b. Какая отрасль похожа на эту? Есть ли там сильные игроки? Как выглядят их решения?

По реальным конкурентам мы смотрим как плюсы, так и минусы. Но минусами не нужно увлекаться, описывая каждую опечатку. Только значимые для процесса продажи ошибки.

Описывая плюсы и минусы, не забывайте маркировать их иконками, иллюстрировать картинками и, главное, – понятно рассказывайте, в чем суть фишки или ошибки, как это влияет на продажи, стоит ли повторять решения и в каком виде.

Если же среди реальных конкурентов что ни сайт, то ужас и кошмар, разработанный на заре интернета (а в некоторых отраслях до сих пор такое случается), то не тратьте время на подробный разбор – дайте общее заключение.

Дополнительно можно рассмотреть сайты в смежных тематиках. Например, если ваш проект про доставку пиццы, сайты по доставке роллов также могут быть источником хороших идей.

В зарубежном сегменте, в идеале, нужно смотреть ту же сферу, что и ваш проект, но если там ничего примечательного нет, то смотрите просто хоть сколько-нибудь релевантные теме крутые сайты, предлагайте фишки.

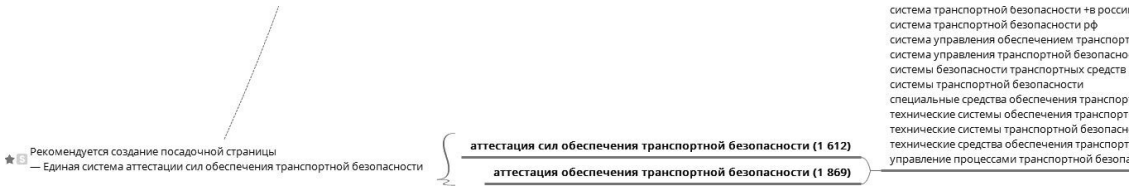
4.2.7. Семантическое ядро (опционально)

Дополнительно вы можете проанализировать, по каким запросам пользователи ищут решения своих проблем. Я считаю, что на этом этапе такой анализ несколько избыточен, так как касается, скорее, не структуры проекта, а его контентной части. Нас интересуют функции с точки зрения программного кода, а не конкретный перечень товаров в каталоге. Более того, пока проект в разработке, ситуация десять раз поменяется. Однако, если ресурсы проекта, бюджеты и компетенции позволяют (а там добавляется 20–40 часов работы) – можно собрать предварительное, базовое семантическое ядро и посмотреть, нужно ли предусматривать в структуре проекта какие-то функции.

Существует множество сервисов автоматического сбора семантических ядер вроде Key Collector. Проще всего собрать запросы вручную с помощью онлайн-сервиса Яндекса – Wordstat. Фиксируйте не только текст запроса, но и число обращений к нему.

Наша задача на этом этапе – понять и показать значимость ключевых страниц и перечень основных запросов, по которым на эти страницы может идти трафик.

Для некоторых интересных, но недостаточно раскрытых в структуре запросов по тематике проекта можно рекомендовать создание отдельных посадочных страниц. Примерно так:



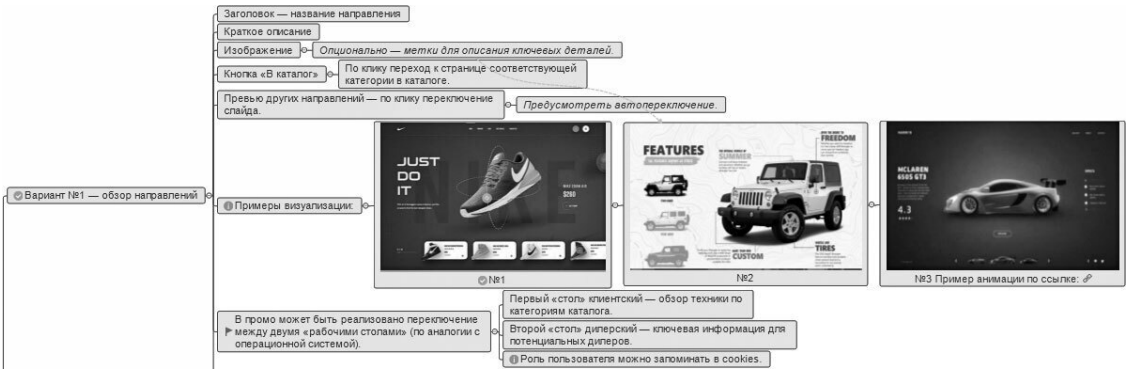
4.2.8. Структура будущего продукта

Кульминация процесса, то, ради чего делались все предыдущие вкладки – составление оптимальной структуры проекта. Оптимальной – значит в ней будут все необходимые страницы и функции для решения поставленных задач и закрытия болей аудитории, но не будет лишних.

Не стесняйтесь отговаривать от реализации очевидно избыточного функционала, который мог попасть в смету по историческим причинам или как хотелка заказчика. Если на сайте три страницы, и все они сплошь картинки, то поиск ни к чему. Заказчик оценит, что вы не только предлагаете функционал, что приводит к росту бюджета и сроков, но и советуете пути оптимизации, что, наоборот, экономит его деньги.

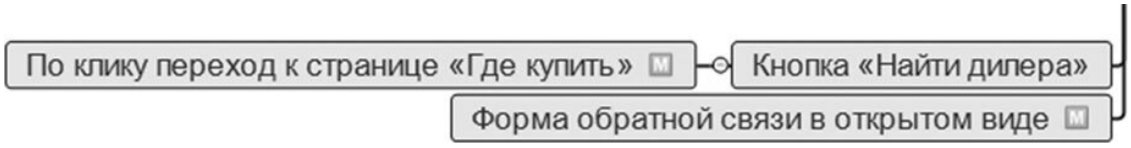
Мы описываем структуру достаточно подробно. Не только перечень страниц или блоков на страницах. Вплоть до состава этих блоков: заголовков, подзаголовков. Часто это помогает составить карту контента. Кроме того, по такой структуре потом очень просто сделать смету, бэклоги или контролировать состав элементов на прототипах и дизайне.

Описывая структуру страницы, не забывайте подбирать примеры для сложных и важных элементов. Это могут быть текстовые примеры – например, возможный слоган или перечень преимуществ – и примеры визуальные. Картинка особенно важна, когда показываешь какую-то визуальную фишку, например, в промо-блоке. Оформляйте так:



Если для анимации будет живой пример (ссылка) – вообще отлично.

Не повторяйтесь, используйте перелинковку.



Как выглядит перелинковка: иконка «М», по клику на которую переносишься в нужный раздел агрегации

Мы еще раз вернемся к декомпозиции проектов в главе 6, поскольку нам важно понимать не только набор функций, но и техническую возможность их реализации. Например, ***ERP-система*** заказчика может быть просто не готова к выгрузке данных в нужном нам виде, и потребуется очень много работы, чтобы это реализовать. Но пока не забиваем этим голову.

4.2.9. Планы на будущее. Развитие проекта (опционально)

Это последняя вкладка Агрегации требований. В ней мы отражаем функции, идеи и предложения, которые нет смысла реализовывать прямо сейчас. Может, через полгода к этому вернемся. Может – через год. Но идеи хорошие, их жалко выбрасывать.

4.2.10. Строим процесс агрегации требований в заказной разработке

Итак. Мы начали с того, что описали видение проекта. Далее проанализировали текущее состояние проекта (если такой был). Посмотрели демографию, точки входа. Описали целевых персон (аватары). Выявили их боли и мотивы. Предложили решения по каждой из болей. Проанализировали сайты конкурентов (что у них можно украсть, а что – ни в коем случае не повторять). Посмотрели семантику. Сделали подробную структуру проекта, возможно, сразу разметив этапность запуска. Что-то вынесли в долгосрочные планы развития. Тут много кропотливой работы, в том числе – с заказчиком. Как нам ее лучше организовать?

Нам понадобится аналитик, который сможет это все структурировать, зафиксировать и выдать предложения. Заказчик ожидает, что аналитик будет проактивен, и большая часть идей будет исходить от него. Именно так и нужно строить процесс.

1. В начале изучите всю доступную информацию: сметы, предварительные брифы, вводные от клиента. Сформулируйте вопросы к клиенту. Дайте изучить эту же информацию вашему аналитику.

2. Проведите установочный звонок, совместно с клиентом и аналитиком.

а. Если вам передают проект от отдела продаж – в разговоре также должен участвовать тот человек, который совершил продажу.

b. Важно на установочный звонок привлечь лицо, принимающее решения. У него может быть гораздо более полная картина мира, чем у руководителя проекта, назначенного со стороны клиента.

c. Предупредите клиента, что часть информации, которую обсуждали на пресейле, могла потеряться во время передачи. Это, к сожалению, неизбежно. Поэтому сейчас нужно еще раз полностью обсудить вводные по проекту.

d. Пройдитесь по всем артефактам.

e. Пусть аналитик задаст все нужные вопросы.

f. Очень внимательно слушайте, что говорит клиент.

3. Далее работайте вкладка за вкладкой. Сделали видение. Показали клиенту со своего экрана. Обсудили устно. Внесли корректировки. Переходим к следующей вкладке. Персон лучше разбирать не более трех за раз (и не городить их суммарно больше семи). Обычно нужно два звонка в неделю, чтобы ровно выстроить такой процесс.

На агрегацию требований планируйте не менее двух недель.

Однако если клиент редко выходит на звонки – процесс может сильно растянуться.

4. Владеть документом должен аналитик. У него должна быть целостная и актуальная картина по проекту. Все правки нужно вносить в документ его руками. Иначе будут нестыковки в структуре, которые потом выйдут боком.

5. Как правило, у клиента уже согласован предварительный бюджет на проект. Пусть с какой-то вилкой. Но есть верхняя граница. Однако в процессе работы над структурой может много чего добавиться. Аппетит приходит во время еды. Сразу помечайте функции, требующие дополнительного бюджета. Это должно быть наглядно. Например, ставьте значок доллара рядом с такой функцией.

6. Редко, но бывает: не все клиенты способны воспринимать формат ***Mind Map***. Презентация в Power Point – это их предел. Мне лично тоже нравится вместо вкладки «Видение» использовать формат Lean Canvas. Однако когда дело доходит до структуры – Mind Map просто незаменим. А размазывать агрегацию по нескольким файлам («Видение» у нас в Lean Canvas, целевые персоны – в Powerpoint, структура – просто документ в Word) – откровенно плохая затея. Теряется логика, нужные связи. Тут придется потерпеть.

Вот и все. У нас получится логичный, цельный, подробный документ, по которому понятно, что за продукт, для кого мы его делаем, и почему состав функций именно такой.

4.2.11. Как продавать агрегацию требований

Когда мы говорим об аналитике, то подразумеваем три основных артефакта, которые появляются в ее результате: агрегация требований, прототип и техническое задание на разработку проекта.

Когда заказчик только приходит с каким-то планом будущего сайта или мобильного приложения, он ожидает услышать сумму сделки, хотя бы примерную: «от» и «до». От этого зависит, состоится контракт или нет.

В разработке софта стоимость любого проекта зависит от необходимого количества реализованных функций, их сложности и заложенного качества. Качество – это сложный комплексный параметр, который трудно объяснить двумя словами. И, конечно, стоимость проекта зависит от маржи компании-разработчика – в среднем по рынку это рентабельность в 20–25 %. Рейты (вместе с зарплатами программистов) только растут, и, похоже, конца этому не будет. В ближайшее время – точно.

На этапе продажи аккаунт-менеджер строит общий каркас проекта, накидывает основные возможности сайта и рамочно их оценивает (подробно об этом мы говорим в главе о деком-

позиции). У заказчика на этом этапе появляется примерное понимание по бюджету и точная стоимость первого этапа разработки – аналитики. Собственно, на ее основе уже можно заключать сделку.

Почему так? Почему нельзя сразу сказать точную стоимость разработки? Допустим, вы решили построить уютный коттедж. Выбрали несколько подходящих компаний, которые могут это сделать. И в первом разговоре с прорабом задаете ему вопрос: «Сколько стоит дом?» Тут велик шанс, что вы получите обтекаемый ответ: «От 300 тысяч до 50 миллионов». Слишком мало данных для оценки, потому и такой ответ. Бюджет на основании этого вы вряд ли спланируете.

Возвращаясь к разработке и аналитике, на этапе продажи мы определяем, какой это будет дом и сколько у него будет этажей: один или пять. На этапе агрегации решаем, какие в этом доме будут лестничные пролеты, как будут расположены комнаты, и разбираемся, что еще нужно предусмотреть, чтобы в доме было удобно и комфортно всем членам семьи. На этапе прототипирования – делаем детальный чертеж. На нем показываем, что и где будет расположено, какого размера будут комнаты, где находятся лестницы и какой высоты и ширины будут дверные проемы.

В физическом мире проектирование зданий, сооружений – это целая отрасль, в которой в России работает порядка полумиллиона человек. Мало кто делает детальный чертеж, пока не подписан договор на проектирование. Это отдельная большая работа, которая требует времени и профессиональных качеств. Планирование и проектирование дома – не подарок, за это берутся деньги. Вы обсудите задачу, заплатите за отдельный этап, и затем уже перейдете к работе.

То же самое происходит и в сфере проектирования сложных коммуникаций, интернет-сетей и во многих других отраслях. Вы в общих чертах понимаете задачу. Можете дать прогноз по бюджету, за всю работу. Но будет определенный разбег. Однако вы можете сказать точную стоимость аналитического этапа – проектирования. После которого уже можно будет просчитать стоимость и сроки всего проекта. Либо определить, что сможет сделать клиент за свой бюджет, а что придется выкинуть на следующие этапы.

Более того, аналитику время от времени заказывают вообще отдельно. Например, чтобы сформировать тендерное задание. Понятно, что у компании, которая делала аналитику, будет самое глубокое погружение в проект и максимальная ответственность. Но, в принципе, это отчуждаемый этап, который можно сделать по отдельному контракту и дальше уже решать, на одной волне или нет.

«Назовите точную стоимость без проектирования и аналитики и впишите ее в контракт, бу-бу-бу. А еще нарисуйте нам пять вариантов дизайна. Прямо к завтрему». Я рекомендую отказаться от таких клиентов. Шалаш без плана построить можно. Высотный дом – нет.

Итак. Стоимость проекта может быть понятна только вилочно, приблизительно. Однако стоимость аналитики можно просчитать точно. И точно показать, за что именно клиент заплатит. Вынести эти работы в отдельный этап. После чего уточнить стоимость. И попасть в ожидаемый диапазон цен.

4.2.12. Что делаем после агрегации

В заказной разработке после агрегации нам нужно:

1. Актуализировать смету. Теперь у вас гораздо более четкое видение проекта, и вы можете либо вообще убрать вилки, либо сократить разбег. В смете вы также можете выделить релизы и спринты. Согласуйте смету с клиентом. Кстати, важно либо четко попасть в вилку, либо четко показать, за счет каких добавлений и новых идей ценник получился выше плани-

руемого. Дальше уже принимать решения – оставляем ли все придуманные функции, или что-то выносим на будущие этапы.

2. Сделать прототипы ключевых страниц. Подробнее о том, как построить по ним работу, как выполнить тестирование прототипов и что нужно предусмотреть – мы поговорим в главе 7, посвященной дизайну.

3. Написать бэклог (или, в зависимости от контракта, – ТЗ, которое затем будет разделено на бэклог).

4. Рисовать. Кодить. Запускать версию за версией. При этом без проблем отклониться от изначального плана: что-то сделать раньше, что-то убрать на следующие этапы.

Этот алгоритм проверен сотнями проектов в заказной разработке. И дает стабильно хороший, предсказуемый результат и высокую управляемость. Однако в продуктовой разработке, когда нет внешнего заказчика, нет никакого видения, все зыбко и непонятно (но дорого) – помогают еще несколько техник.

4.2.12.1. Прототипирование

Итак, агрегация требований готова и утверждена с заказчиком. Самое время приступать к прототипу – черно-белой интерактивной схеме сайта, по которой можно поводить мышкой, покликать по кнопкам и посмотреть, как будет работать проект без затрат на дизайн и программирование. На этом этапе уже видно, в каком порядке будут располагаться основные блоки на странице и где будут находиться кнопки, предполагающие целевые действия.

Цели прототипа:

▶ согласовать структуру сложных страниц с заказчиком;

▶ проверить, будет ли проект удобен для конечных пользователей с точки зрения персон;

▶ убедиться, что все участники проекта одинаково представляют зафиксированные в структуре идеи, а сами идеи жизнеспособны – понятны и удобны для пользователя;

▶ предложить альтернативные решения для нежизнеспособных идей (самое время).

Прототипа достойны не все страницы сайта, а только важные и сложные. Список страниц к прототипированию определяется сметой. Главная страница – всенепременно. Если делаем интернет-магазин – то также проектируются список товаров, карточка товара, корзина, форма заказа и личный кабинет.

Если в разделе «О нас» задумана презентация сотрудников с портретами и мини-биографией, чтобы клиент мог выбрать к кому обратиться и тут же оставить ему запрос на обратный звонок (заполнив для этого форму с обязательными и невероятно важными полями) – прототипируем. Если планируется фото фасада и три строчки адресов-телефонов – просто обсуждаем с заказчиком и затем фиксируем в ТЗ.

Программ для прототипирования примерно миллион, причем, есть и онлайн-ресурсы, и десктопные программы. В последнее время мы все чаще делаем и прототипы, и проекты в Figma – уж больно она хороша и удобна. Вы же можете выбрать какую угодно программу, главное – следовать принципам:

Думайте, как этим будут пользоваться реальные люди

Будет ли это удобно? Решит ли это их проблему? Поставьте себя на их место.

Выравнивайте за собой везде и всегда

Когда вы готовите прототип, обязательно выставляйте в программе линейки, по которым проверяйте расположение элементов. Это стандартный функционал большинства специализированных программ. Прототип должен быть не только функциональным, но и красивым. Критерий красоты выделить трудно. По крайней мере, при просмотре не должно быть ощущения, что работу выполнили наспех, потому что «все равно же дизайн впереди».

Используйте реальный контент

Проверяйте орфографию. Да, ошибки в словах не исказят суть прототипа, но вот впечатление они подпортят. Особенно, если просматривать его будет человек грамотный и внимательный к мелочам. А мелочь ничего не прощает большому.

Пишите комментарии по неочевидным элементам

Если непонятно, как будет работать какая-то кнопка, – добавьте к ней комментарий прямо в прототипе, чтобы не потерять этот нюанс и заранее подготовиться к вопросам, которые вам могут задать. Оставлять комментарии позволяют очень многие программы для прототипирования.

Не шутите в аналитических документах

Смешные картинки или веселые фразы воспринимаются занятыми дядями на стороне заказчика как попытка их разозлить. А вдруг получится? Корректнее всего использовать деловой стиль общения.

Используйте метод прогрессивного джипега

Сначала накидываем общую картину, без детализации, основными блоками. Затем – постепенно детализируем каждый блок до нужной степени.

Бывают ситуации, когда на главной странице нужен супер-вау-эффект. Что делать в таких ситуациях? Рисовать в промо-блоке большой зачеркнутый прямоугольник с подписью: «Здесь будет креатив» – как-то не очень удобно, плюс порождает сотни вопросов у заказчика. Двигать этот вопрос на этап дизайна (мол, потом разберемся) – не всегда экологично: а, собственно, зачем тогда в этой схеме прототип? Совсем без прототипа – велик риск не попасть в ожидания на дизайне, и в дальнейшем поиметь множество проблем с перестановкой блоков на готовом макете.

Мы поступаем следующим образом:

1. Перед началом прототипирования проводим брейншторм. На нем сразу придумываем какую-то фишку проекта. На брейншторм обязательно зовем дизайнера и аналитика.
2. Дизайнер накидывает карандашный скетч идеи, которую мы придумали.
3. Привлекаем студийного копирайтера: он готовит тексты для страницы.
4. Аналитик делает прототип: вставляет реальные тексты, схематично показывает идею.
5. Презентуем прототип и скетч заказчику вместе с дизайнером.

У нас этот процесс работает. Но если ваш проект с действительно креативной главной страницей, которую никак не запрототипировать, – следуйте здравому смыслу. В таких случаях достаточно только скетча.

4.2.12.2. Пишем ТЗ. Годные шаблоны

Агрегация есть, прототип готов и утвержден – самое время перейти к техническому заданию на разработку проекта. ТЗ – артефакт известный. Это толстый документ, в котором очень детально описано, что за проект в итоге получится. Беда с этим документом в том, что им очень неудобно пользоваться. Огромный объем сурового технического текста со специальной терминологией крайне сложно читать. А уж представить себе итоговую картинку в голове, понять, как это будет работать физически, – практически нереально. Особенно если опыта в чтении таких ТЗ немного.

Поэтому в digital-разработке случаются ситуации, что заказчик вдумчиво изучает ТЗ и утверждает его, а потом искренне удивляется, что в итоге получилось именно то, что и было описано.

> Редко какие готовые проекты хотя бы на 70 % соответствуют техническому заданию. Юзабилити, бюджет, санкции – что угодно может стать причиной изменений в проекте.

В чистом Scrum-е как такового технического задания нет. Там есть бэклоги. Бэклог – это список функционала (по-другому, фич), отсортированный по приоритетам, по которым идет разработка. Можно менять фичи в любой момент, и это не вызывает проблем на стороне команды. Приоритеты при этом регулярно пересматриваются.

У бэклогов есть свои нюансы и проблемы, и мы их уже разобрали в главе про Scrum.

Но с классическим ТЗ, в котором жестко «морозятся» все требования, проблем, на самом деле, гораздо больше. Дело в том, что невозможно описать четко и однозначно требования так, чтобы люди их поняли точно так, как вы того хотели. В любом ТЗ на спор на коньяк находятся какие-то прорехи, которые можно трактовать двояко. Причем, чем толще и подробнее ТЗ, тем больше таких вещей встречается.

Казалось бы, ответ простой: еще больше конкретики! Идея откровенно так себе: если вы будете подстраховываться и описывать каждую деталь максимально подробно, то скатитесь до регламента завязывания шнурков и инструкций, что кота нельзя сушить в микроволновке.

Особенно худо, если подробное техническое задание пишут несколько человек (а такое бывает) – как правило, получается шизофрения листов на 300–400. В определенный момент (часов этак через восемь чтения) человек, который внимательно изучает эти 400 листов, хватается за голову и смотрит в стену с единственным желанием – выбросить это все и начать заново. Потому что полностью разобраться в документации – займет месяцы. А этих месяцев у проекта просто нет.

Конец ознакомительного фрагмента.