История ошибок менеджера

ЧЕРНАЯ КНИГА СКРАМ

Записал Иван Селиховкин

Предыстория

Иван Селиховкин нашел этот блокнот при переезде в новый офис. Раньше здесь располагалась крупная компания-интегратор. Они съезжали спешно и много бросили — Иван обнаружил записи, переставляя мебель. Автор, похоже, то ли рвал, то ли жег свое творение прямо на рабочем месте — не успел и забросил остатки в щель между кулером и стеной.

Говорят, в некоторых компаниях практикуют внезапные увольнения — вызывают к директору, и, пока ты там, служба безопасности уносит комп, блокирует учетную запись и пропуск. По возвращении — охрана помогает собрать личные вещи и провожает до двери. Документы выносить не дают.

Возможно, с автором дневника приключилось нечто подобное. Личные заметки он уничтожить не успел, а показывать не хотел. Боялся за «карму»? Профессиональную репутацию или выходное пособие?

В темном от копоти и пыли блокноте хуже всего сохранились первые страницы, где объяснялось, о чем и для кого дневник. Удалось разобрать нечто среднее между эпиграфом и посвящением:

> *«…Скрам не всегда работает. Это мое предупреждение всем. Он может погубить твою карьеру так же, как растоптал мою. Не знаю, сколько еще осталось. Но буду конспектировать каждый день… [обрывается]*
>
> *…предупредить остальных… [обрывается]… принимай решения и вдумчиво выбирай подходы. Не спусти несколько лет своей карьеры в унитаз, как сделал я! … [обрывается]».*

Полистав, Иван решил законспектировать, но увлекся и переписал дословно все, что сохранило время.

Электронной версии пришлось дать собственное название.
Так родилась «Черная книга Скрам».

Деспотичный гуманизм

Agile манифест — психологическая ловушка для управленца

О том, как мы полюбили agile, который нам ничего не обещал и даже не говорил ничего нового. И перенесли свои ожидания на Скрам. А Скрам, вопреки манифесту, задал жесточайшие правила игры.

Я втянулся в Скрам случайно. Как и тысячи других невинных жертв. Вот что усыпило мою бдительность: гуманистический настрой и «мягкие принципы» agile манифеста. Он был нашими «воротами в Скрам». Сбил с толку. Как я мог быть так наивен!

Я проваливаю самый главный проект в моей жизни. Вернее, проваливаем все мы, но уволят именно меня. Я — управленец, на мне основная ответственность. На рынке сейчас полно менеджеров и нехватка инженеров. Так что меня уволят. Едва станет окончательно понятно, как глубоко мы зарылись и как безнадежно отстаем от срока по крупным релизам.

Буду делать пометки. Напишу сколько успею. Сейчас думаю, главная причина провала — это «бездумный Скрам». Не надо было его применять вообще. Не на нашем проекте.

Попробую вспомнить, как я втянулся.

В памяти довольно отчетливо — через «манифест». Тот самый набор ценностей agile, который может не устроить только сумасшедшего. «Ну, я-то нормальный», решил я и стал думать, как претворить их в жизнь. И, конечно, подвернулся Скрам. Практическая методика (почему-то ее обзывают «фреймворк» — бессмысленное и непереводимое слово), предлагающая конкретный набор ролей и правил. От которых, как выяснится, нельзя отступать (но об этом позже).

Но сперва был agile.

Помню то утро, когда ребята показали и рассказали про манифест. В нем все было стройно. Но уже первая строчка сигнализировала мне «Agile манифест разработки ПРОГРАММНОГО обеспечения». Программного. И ничего другого.

Этот подход придумали для программистов. Причем не для любых, а для особенных (об этом позже). Никто не предполагал в проектах консалтинговой составляющей, объемного внедрения, строительства и проектирования каналов связи, работы с контрактной документацией. Неспроста в команде разработки Скрама останется всего одна роль — «разработчик» (об этом тоже потом). Авторы заботились о программистах. Манифест, как и руководство по Скрам, — абсолютно

программисто-центричен. Остальным (даже заказчику) в нем нет комфортного места.

Но это полбеды. Если вспомнить манифест дословно:

> *«Благодаря проделанной работе, мы смогли осознать, что:*
>
> *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»*.

Я почему-то позволил себе думать, что это признаки исключительно agile. Но это же простой здравый смысл! **Основа любого менеджмента вообще**. Я поработал и на государство, и на финансовые корпорации — видел примерно половину всех возможных «бюрократий». Таких, про которые принято говорить «у-ууу, там водопадный подход» (это не так, по крайней мере, не обязательно так). Но даже там результат ВСЕГДА был важнее документации, а сотрудничество с заказчиком позволяло договориться о любом дополнительном к контракту соглашении в самой сложной ситуации. Ни на одном проекте планы не остаются нерушимы, они меняются, и это норма.

Дальше в манифесте: «**То есть, не отрицая важности того, что справа, мы все-таки больше ценим то, что слева**».

Вот так будет всегда, когда станешь иметь дело с agile. «Мягкие лапы», уход от ответственности, гимны — вместо внятных инструкций. Черт, он же все время был там! Прямо перед моими глазами! Абзац, делающий весь манифест не более чем набором благих пожеланий. Т. е. и «процессы важны», и люди, и их взаимодействие. И продукт, и документация. Т. е. важно вообще все. Но в критичных ситуациях нужно не бодаться, а договариваться, не прятаться за процессами, а идти, общаться и выяснять потребность.

Да кто спорит-то?! **Надо было в тот момент сказать команде: «Я тоже за» — и продолжать работать по PMI**. Мы так тогда делали. Или по Prince2, как было до того. И с заказчиком общались, и продукт рабочий создавали, и людей на первое место ставили… Нет смысла разбирать еще 12 принципов, они также не привносят ничего, с чем не согласился бы любой нормальный управленец. Не важно, проектами он руководит или отделом, и какую методологию (PMI, IPMA, Prince2 или иную) «исповедует». Надо было заявить: «У нас уже agile согласно вашему же манифесту, идите работайте».

Но мы совершили классический второй шаг и, прочитав полдюжины книг, погрузились в Scrum Guide.

Я много повидал фреймворков, стандартов и методологий для управления чем бы то ни было. Но Скрам гайд — самый жесткий из всех, что я видел. Он сразу ставит себя бескомпромиссно и снимает ответственность.

Первое, чем встречает Скрам гайд (вернее, технически, это одна из последних в нем строчек), — невероятная жесткость, прямо противоречащая agile-принципам. Прекрасно помню абзац:

> *«…Детальное описание фреймворка представлено в рамках этого Руководства. Роли, артефакты, правила и события Скрама не подлежат изменению. Хотя использование отдельных элементов данного фреймворка допустимо, но полученный результат не может называться Скрамом. Скрам существует только в качестве цельного фреймворка, но он может вмещать в себя другие техники, методологии и практики».*

Роли, артефакты, правила менять нельзя. Нельзя менять правила! Черт! Иначе это уже не Скрам. А как же «люди и их взаимодействие важнее процессов»? Вот что должно было меня насторожить. Самый популярный управленческий agile-фреймворк прямо так и говорит. Это Скрам. Ничего менять нельзя! Даже не думайте называть это Скрамом в противном случае! И да, это agile.

Куда я смотрел? Как проглядел этот **деспотичный гуманизм**?

Когда позднее мы начали внедрять Скрам и сносить проектное управление, команда тыкала меня носом в этот абзац. «Ничего менять нельзя, — говорили они. — Или ты против agile?!» А как можно быть против agile, ведь в манифесте описан здравый смысл.

Это психологическая западня. Либо ты внедряешь Скрам, либо ты против здравого смысла. Как я вообще в ней оказался? Не понимаю. Надо было опираться на манифест (все же, там — рациональное зерно, а гайд — набор ритуалов, не всегда понятных и ориентированных на удобство программистов, способных к самоорганизации). И применять те практики, которые упрощают создание продукта и взаимодействие с заказчиком. Неэффективные и странные — выкидывать.

Лучше всего — сказать бы, что **у нас не будет совсем уж классического гибкого подхода, вроде Скрам, но «майндсет» у нас agile-ный**, и каждую ценность, каждый принцип я готов поддержать. И действовать согласно agile-майндсету, а не букве «исчерпывающего руководства по Скрам» [строка на титульном листе Scrum Guide]. И все. На этом злоключения и кончились бы. Вернее, они бы не начались. Но сила инерции и боязнь порицания велики. Лучше бы мы не боялись и не придерживались Скрам гайда строго.

Ретроспективное планирование

Сперва работа, потом сроки

О том, как мы сперва не могли назвать заказчику сроки проекта на старте. А потом поняли что никогда не узнаем даже собственной производительности.

Вот первое, на чем мы серьезно обожглись.

На ранних этапах заказчика практически всегда, волнуют сроки. И деньги. Это вопросы к обсуждению. Но Скрам запрещает такую постановку вопроса.

В человеческой жизни не так уж много примеров работы, которую можно делать по частям и сколь угодно долго, лишь бы регулярно поставлять полезные элементы.

Капитальный ремонт квартиры, строительство дома или закупка еды для пикника, учеба в институте. Все это имеет мало смысла по частям. Важно, когда и за сколько в конечном счете в квартиру можно будет въехать или когда основная масса критично-нужной еды для пикника окажется в багажнике, к какому году вообще теоретически возможно закончить институт. Все эти цифры имеют значение. По ним можно понять, стоит ли вообще связываться с пикником, ремонтом и ВУЗом прямо сейчас или лучше отложить. Отучиться два курса и «там сориентироваться» — часто нерациональный подход, особенно для человека, обремененного семьей или тщательно планирующего карьеру.

В бизнесе точно так же. Заказчику надо знать, во что он втягивается и будет ли в этом экономический смысл. Директору на внутренних проектах — цифры на старте, порой, важны не меньше.

Что предлагает заказчику Скрам? Постоянное общение, вовлечение в каждый спринт (что, кстати, далеко не всех обрадует — у заказчика свои дела). Но не оценку сроков.

Современные методологии оценивают менеджера проектов по тому, как он и его команда умеют удовлетворять ожидания заказчика и при этом попадать в первоначально названный срок и бюджет. Но Скрам не проектный, а продуктовый подход, что многократно повторяет Scrum Guide.

Скрам отметает все, кроме удовлетворения ожиданий заказчика. Работа с расписанием для него второстепенна. Скрам **верит в заказчика, которому не важны сроки и деньги в начале**.

Единственный способ оценить продолжительность в Скрам — начать работать. Хотя бы 2–3 спринта (лучше — больше). При двухнедельных спринтах счет идет на месяцы. Тогда становится понятна «velocity» («мощность» команды). То, сколько задач и какой сложности команда может «переварить» за спринт. По этой «мощности» определяют — сколько еще понадобится времени (спринтов), чтобы сделать все оставшиеся в текущем бэклоге задачи (в случае, если он не будет изменяться).

На старте, пока работа не началась, определить сроки невозможно. Скрам это прямо запрещает. Нет velocity — нет оценок. Вот тут надо остановиться и перечитать. Не отработал несколько спринтов — оценок нет.
Некоторые эксперты предлагают специальный «нулевой спринт» вне проекта, но из-за того, что команда врабатывается — он крайне плох в качестве ориентира.

Другими словами, заказчик интересуется: «Ребята, когда закончите?» Ответ: «**Подожди месяцок-другой, мы тебя сориентируем»**.

Серьезно. Чтобы понять абсурдность ответа, достаточно поставить себя на место заказчика ремонта, покупателя дома, абитуриента или поступающего в ВУЗ и слышащего что-то вроде: «Ты сперва поступи, поучись, мы потом скажем, сколько у нас курсов и во сколько тебе диплом выйдет».

«Взамен получишь регулярные занятия, прирост знаний — ты же за ними в первую очередь пришел?» И да, и нет — пришел за знаниями, но и диплом, и звание, которое он дает, имеют значение.

Хотя получить сроки спустя пару спринтов — только половина проблемы. Настоящая беда в том, что сроков вы все равно не получите.

Причина в коллективной ответственности.

Скрам считает продолжительность, оттолкнувшись от «мощности команды». Повторим это еще раз: «Команды».

Velocity не рассчитывают на человека. Она относится к команде и только к команде. Своеобразный «социализм» — одна из ключевых парадигм Скрам.

Проблема в том (черт, я должен был это предвидеть), что люди на проекте меняются. Одни уходят в отпуск, другие пропускают по болезни. Кто-то (да, и такое бывает) — увольняется. В команду вливаются новички. Каждая такая перестановка как-то влияет на velocity. Никто не знает, как. И это официально.

Один человек (пусть временно) оставил команду? Появились новые сотрудники? Velocity изменилась! Какая она? Станет ясно после… да, 2–3 спринтов (месяц–полтора).

Практика (моя практика, и волосы, поседевшие и выпавшие на этом проекте) свидетельствуют, что люди на проекте как раз перемещаются каждые 2 месяца (пресловутые отпуска и болезни). **А значит, мы НИКОГДА не знаем «мощности» команды**. Метрика, единственная, позволяющая прогнозировать сроки в Скрам, не работает в реальной жизни никогда. Ты никогда на будешь знать срок окончания проекта, ибо velocity в очередной раз недавно изменилась, мы ее только нащупываем.

И что дальше? На что похожа красивая диаграмма сгорания проекта? Она выглядит странно, без шансов.

А если не работает velocity, то что дальше? Отказ от оценок? «No estimate?» Где еще это в реальной жизни работает кроме фантазий Scrum-оводов?

Любые попытки приспособить под проблему «костыли» («мы делали похожий проект, похожими людьми, или тут уволившийся стоил двух новичков, так что...») технически является самообманом. И да, Скрамом уже не является (помнишь про «деспотичный гуманизм»?).

Нужно было сразу ставить вопрос ребром. Разобраться, есть ли на проекте реальные ограничения по срокам и деньгам, и насколько они важны заказчику. И если да, — отказываться от фреймворка, либо поднимать вопрос внутри команды (хотите Скрам — давайте вместе выработаем решение «как»). В противном случае честно предупреждать заказчика (или директора, если проект внутренний). Работая по Скрам, мы не только никогда не сможем сориентировать его по срокам до старта проекта, но и по ходу вряд ли хоть когда-нибудь измерим реальные даты завершения. И применять фреймворк только, если тот самый заказчик или директор с этим согласен. Иначе, огромный шанс привести именно к тому финалу, что вырисовывается у нас.

Кросс-функциональность

О взаимозаменяемости

О том, как Скрам предъявлял нереальные требования к нашей (и почти любой другой) команде.

Я видел это в Скрам гайде. Читал своими глазами и не придал значения. Сколько мы спорили потом!

«Скрам-команды являются... кросс-функциональными... Кросс-функциональные команды обладают всеми необходимыми компетенциями для выполнения работы и не зависят от людей, которые не входят в команду».

Эта фраза всем кажется очевидной. У нас в компании из-за нее едва не началась «гражданская война».

Что она значит? Команда кросс-функциональна как единое целое (в ней есть все нужные специалисты — например, программист, разработчик, аналитик, дизайнер)? Или каждый специалист немного мастер на все руки и может все что надо (немного тестировать, немного программировать и т. п.)?

Понимаешь, к чему я клоню? Если речь про второе, — то сразу проблема. Минимум в 90 % компаний каждый продукт делает созвездие специалистов. Люди разных профессий, способные дополнять, но не заменять друг друга. Программист рисовать не умеет, так что он не помощник дизайнеру, а тот (порой, гуманитарий) ничего не понимает в программировании. В Scrum Guide есть довольно пугающая фраза: «Разработчик — единственная роль для членов Команды Разработки, независимо от типа задач, которые он выполняет. Скрам не признает других ролей в Команде Разработки, это правило не имеет исключений». Давай пока поймем ее в переносном смысле, мол, «разработчик от слова работа, в том смысле, что в Скрам — менеджеров нет»). Ведь если нет, то сразу мимо — в моей (возможно, и в твоей компании Scrum просто не подойдет).

«Покрутим» Scrum Guide дальше. У нас кросс-функциональная команда в первом смысле слова (команда умеет все что нужно, но у коллег специализация). Слову «команда» Скрам придает особое значение. Такое, что практически снимает индивидуальную ответственность. Есть спринт и есть коллектив. Важно не чтобы каждый сделал то, за что взялся, а чтобы до конца спринта успел «весь колхоз». Стоит одному споткнуться, как другие бросятся ему помогать, лишь бы уложиться в спринт.

Твой тестировщик не успевает с работой до конца спринта? За пару дней до финиша программисты (если они более-менее закончили свое) бросаются на выручку. Да, они не тестировщики топ-класса, но помощь окажут. Коллектив в целом справится, спринт спасен.

Когда запахло жареным, мы слушали тренинги и читали книги, к нам приходили agile-коучи. Они всегда поясняли на подобном примере. Но почему не наоборот? Что, если разработчик «споткнулся» и помощь нужна ему? Тестировщик и дизайнер интерфейсов предложат ему свою помощь? Какую? Кодить что ли будут? Да и обрадуется ли такой «помощи» программист? Очевидно, не сильно. По той же самой причине, по какой старший программист не радуется стажерам (они не ускоряют, а замедляют его работу), спасти спринт силами неспециалистов не получится.

Так что же, «...не признает других ролей в Команде Разработки, это правило не имеет исключений» — неужели в прямом смысле слова? Погоди. Еще немного боли!

Еще больше крови мы пролили с тем, что в Скрам не принято планировать заранее. Это основная идея «не тратить время на ненужное планирование». Просто есть бэклог, из него берем самые приоритетные «хотелки» (задачки) и реализуем за спринт.

Звучит неплохо.

Но каждую задачку нужно сформулировать. Желательно в том виде, в каком она полезна пользователю. Бывают элементарные (цвет кнопок в интерфейсе), бывают посложнее (печать отчета или справки в установленном виде). А значит, сперва над задачкой должен кто-то поломать голову. Обычно «аналитик» разбирается, что и в каком виде должно попасть в справку (порой для этого приходится не один день побегать между разными пользователями). Все это записывает и передает в разработку, а сам убегает дальше, размышлять над следующей задачкой.

Разработчик реализует, а потом должен кто-то проверить. Приходит тестировщик (возможно, он пришел еще раньше и писал свои тест-кейсы, но по завершении разработки он должен появиться точно).

В нашем с тобой примере уже есть три роли. Над каждой «хотелкой» они работают в определенном порядке (упрощенно: сперва аналитик, потом разработчик, потом тестировщик).

И как это в жизни? К примеру, запускается спринт. В него включены 4 больших «задачки-хотелки» (что вполне по силам команде).

Если подойти к реализации «в лоб», получится, что сперва над каждой должен поработать аналитик (у остальных пока нет работы — сидят, курят). Потом разработчик (как аналитик закончит со всеми тремя задачами, — «курить» уже начнет он, причем до конца спринта он свое дело сделал). В конце подключатся большую часть времени скучавшие тестировщики (и под конец работать будут почти только они).

Мы с командой смекнули, что в таком простое нет смысла. И оттого попытались в следующий раз нагрузить аналитиков работой заранее. Т. е. в рамках спринта они как бы «пилят» постановки на будущее.
Разработчики — работают в своем ритме, а тестировщики идут с запозданием в спринт и проверяют сделанное.

И тут у нас развалился Скрам. Мы обнаружили, что создали **три команды, работающие параллельно** (аналитики, разработчики, тестировщики).
Друг за другом. И постоянно договаривающиеся (кому и что делать).
Причем на деле все еще хуже (у нас было не три роли, а 5 или 6 — там добавляется дизайнер, технический писатель, иногда DevOps и другие).

Кроме того, стоит кому-то в этой цепочке споткнуться (например, аналитики не справились в свой спринт и к началу следующего не нагрузили разработчиков работой), и весь производственный цикл встает. Чтобы этого не происходило, приходится работать с запасом (у аналитиков написанных требований должно быть на 2-3 спринта, чтобы если что…). И вот это уже точно не Скрам.

Во фреймворках есть свои идеи. Главный посыл. «Мякотка», которую если выбросить, останется одна лишь змеиная шкурка (так осталось и у нас — один бэклог да спринты). Скрам на словах, не на деле. А где простое планирование? Где вовлечение заказчика? Где скорость реакции на его предложения (если у нас требования «пилятся» с запасом на 1–1,5 месяца вперед)?

И ключевое: основа Скрам в том, что каждый спринт приносит пользу. Поработали две недели — заказчик сразу получает прирост полезности. У нас не так. Аналитик работает и 2, и 4 недели над тем, что не имеет ценности для заказчика (он не питается его постановками), продукта нет. Огромная часть коллектива трудится не над тем, что единственно и имеет смысл, а над чем-то вспомогательным.

И тут меня ударило. **«Скрам… не признает других ролей в Команде Разработки, это правило не имеет Исключений». Это в прямом смысле слова!**

Помнишь, что agile манифест — для разработки программного обеспечения и только для него? А Скрам — самый популярный фреймоворк для манифеста.

Все в твоей команде должны быть разработчиками. Тогда то, что написано в Scrum Guide, работает. Иначе — не имеет смысла! Не повторяй моих ошибок — обращай внимание на такое сразу.

Надо было здраво оценить количество ролей и удельный вес разработки. Приниматься за Скрам в проекте, где команда является полноценным созвездием из разных ролей и профессий, — только когда вместе с командой, коллективно, будет найден ответ на вопрос: «Как организуем взаимодействие ролей?»

Командная ответственность

Чувствительность к токсичности

О том, как наша Скрам-команда раз за разом оказывалась беспомощна перед проф. непригодными личностями (а также перед лентяями, грубиянами, демагогами и так далее).

Некоторые вещи звучат хорошо. Но работают не всегда. Причем я должен был предвидеть это. На примере отдельных стран и компаний.

В Скраме нет личной ответственности. Только командная.

Вернее, не так. На уровне абстракций «каждый член команды делает свою работу максимально хорошо, ради своей команды и того, чтобы она уложилась в спринт».

При этом никто и никогда не сможет определить производительность отдельного сотрудника. Сравнить с какой скоростью он работал месяц назад и с какой сейчас. Личные метрики невозможны. Только обмен мнениями, субъективные ощущениями на ретроспективах. Да и иначе обязательно начнется менеджмент. Придет руководитель и начнет «ускорять», «корректировать задачки», в общем, вмешиваться всячески. И нарушит основной принцип Скрам — «команда внутри спринта неприкосновенна» (а значит, может сфокусироваться).

Зато есть ответственность коллективная — уложиться всем вместе в спринт, чтобы не подводить товарищей, помогать им, если нужно. Сделать это трудно по определенным причинам (помнишь «кросс-функциональность»). Либо мы все вместе успешны, либо одна большая неудача на всех. Без исключений. Вытягивать же персонально-закрытые «тикеты» из ИТ-систем — означает подрывать атмосферу, в которой поддерживаешь товарища, не думая о том, кто в итоге закроет задачку.

Я должен был лучше понимать свою команду!

У нас были замечательные специалисты. Опытные и неравнодушные. Неопытные и неравнодушные были тоже. Но были и демотивированные. Уставшие от работы.

Принцип командной ответственности не работает в токсичном коллективе.

Я же видел страны, где пытались установить социализм. Но даже там на уровне лозунгов было «от всех по способностям, каждому по труду». На деле коллективная ответственность размывает эту грань.

Одни тянут, других везут. «Ведь мы же не хотим все вместе провалить спринт». Без вариантов.

Скрам предполагает, что команда сама избавится от токсичных персонажей. Черта с два! Это очень трудно даже опытному менеджеру (привыкшему к сложным переговорам, возможному негативу). И вовсе оказалось невозможно для нашего инженерного коллектива. Ребята хотели работать. И не были готовы выяснять отношения. Даже если цена — «притягивание за уши лентяев». Agile-коуч и scrum-мастер даже близко не справились с задачей (грубо говоря, это вообще не их работа — они продвигали, куочили, задавали вопросы и не более).

Справляться с токсичным коллективом сложно. Не каждому менеджеру это под силу. Рядовым (и не токсичным) сотрудникам это, порой, не по плечу. Примеры саморегулирующихся жестких сообществ (армия) это подтверждают. Если твоя команда не похожа на тюрьму или армию со своими «авторитетами» с их опытом в разруливании коллективных проблем — **даже не надейся, что Скрам-команда сама (без менеджерской помощи) решит большую часть коммуникативных проблем.**

Надо было сразу отдавать себе отчет, насколько токсична наша команда, насколько она команда вообще. И, если проблема выражена, — не применять Скрам, не питать иллюзий, что с его помощью со временем исправится любой неидеальный коллектив.

Полезная функциональность

Можно ли строить мост по Скрам

О том, как мы обнаружил, что Скрам не подходит для некоторых

наших продуктов.

Идея Скрам — каждый спринт выдавать заказчику «новую пользу». Возможно, ты помнишь эту глупую картинку, где покупатель сперва получал скейт, потом велосипед, потом мотоцикл, а потом и авто. Вместо того чтобы сразу и долго ждать машину? Мы тоже ее помним. Мы с нее начинали.

Почему глупая? А как можно из скейта сделать велосипед? Только выкинуть наработки и сделать заново, с нуля.

Ты понимаешь? Скрам даже не скрывается! Он сразу показал нам иллюстрацию того, что он не работает на большинстве задач.
Но мы восприняли ее с воодушевлением!

Мне довелось руководить проектом по созданию серверного софта. Он был частью большого прибора и должен был повысить скорость его работы и снизить энергозатраты.

Мы решили работать по Скрам. Сформулировали «хотелки», стали дробить на спринты. И тут проблема.

Наш софт в принципе не имеет «хотелок», которые волновали бы конечного пользователя. Заказчик попросил всего о двух вещах: «чтобы быстро» и «низкий расход энергии». **Наше решение нельзя было проверить, не закончив разработку.**

Примерно как велосипед нельзя испытать на скорость или устойчивость к грязи, пока он не собран и не поставлен на дорогу.

Вот если бы мы делали веб-сайт! Где сотни элементов — кнопочек, фильтров, цветов. И про каждый у заказчика было бы свое мнение! Но нет. У нас скучнейший (для стороннего глаза) «сервак», который мы пишем ради отдаленных выгод. Мы можем погонять нагрузочные тесты на полпути (но заказчик их не понимает). Можем показать ему «админку» сервера (но она простая, да и не нужна — он просил не ее). Мы долго разрабатываем архитектуру и можем спорить об элементах на ретроспективах в конце каждого спринта (мы так и делали), но **это «мертвые» спринты. В них нет Скрама. Заказчик не получил ничего.**

Ни велосипеда. Ни скейта. Ни даже колесика (а если бы и получил — что бы с ним делал, куда ему велосипедное колесо?).

Запомни: скрам работает не везде. У нашего продукта не было большого количества пользовательской функциональности. И мы дробили на спринты задачки, которые волновали только нас (а заказчику на эти промежуточные варианты было плевать). Он ждал сразу автомобиль, потому что единственное что его волновало — не цвет кузова, а скорость на автобане.

Мы навязали пользователю наши подходы, втянули в спринты, не имевшие для него пользы, спрашивали мнение там, где оно не имело значения.

Если не знаешь, что будешь демонстрировать полезное для заказчика по результатам каждого спринта, — забудь про Скрам.

Надо было оценить соотношение пользовательской функциональности (кнопочек, плашечек, сценариев и кейсов) и функциональности прочей, трудоемкой в реализации (архитектурная, аппаратная, серверная часть и другие). Если перевес не в пользу первой, — отказаться от Скрам, в нем не будет смысла.

Помнишь фразу «нельзя построить мост по Скрам?» Так вот: нельзя построить мост по Скрам!

Скрам голоден до времени заказчика

Избыточное вовлечение

О том, что Скрам напрасно гордится сильным вовлечением заказчика (иногда это плохо).

Это не самая большая проблема Скрам. Но одна из них.

Скрам требует, чтобы пользователи приходили на каждый спринт и забирали результат.

Не важно, хотят они этого или нет. Нужны им промежуточные результаты или нет.

Скрам иначе не работает.

Представь, ты делаешь ремонт квартиры. Там сперва все демонтировали (приходи, смотри). Затем побелили потолок и подвесили лампочку в патроне (приходи). Потом сантехник переварил фановую трубу и временно поставил унитаз, а строитель разломал ненужную межкомнатную перегородку (снова смотри).

А если нет? Если я не хочу? Можно я приду 2 раза за полгода?

Нет. Нельзя. Выдели время каждую неделю (или 2, или месяц — смотря какие у тебя спринты).

Заказчика вовлекать надо. Но **не каждый заказчик любит, чтобы его дергали часто.**

Особенно, если он ждет автомобиль, а ему показывают промежуточный скейт. Нет смысла показывать заказчику, ждущему десерт, наполовину прожаренную булочку.

Надо было сразу объяснить заказчику что его ждет и убедиться, что он действительно готов тратить время (и не объяснять его возможное нежелание боязнью перемен).

Скрам и внешний мир

Совместимость с другими подходами

О том, как Скрам хочет иметь дело с владельцем продукта и «брать деньги из тумбочки» и его не заботит работа организации в целом (в лучшем случае он порекомендует всей компании «тоже стать agile»).

Скрам удобен команде. Бесспорно! Люди, которые пришли работать (программировать), хотят именно программировать. А не строить планы. Скрам — это изначально «манифест разработчиков» (а не управленцев или бизнесменов). Скраму, на самом деле, плевать на основную компанию.

Вернее, не так, адепты довольно хитро научились говорить «чтобы в команде работал agile, желательно, чтобы вся компания была agile».

Раньше я тоже стоял под флагом с надписью «гибкость» и повторял этот лозунг. Теперь я смотрю на это иначе. Мне он кажется просто хитрым способом обосновать дорогой консалтинг, чтобы скрыть базовые проблемы подхода (все те, о которых выше).

Компания живет деньгами. И рисками. Она их планирует, предвидит проблемы, раскладывает их по портфелям. Портфель балансируют так, чтобы одни риски (скажем, экспортозависимые продукты) уравновешивались другими (например, не зависящие от экспорта, но реагирующие на курс национальной валюты и локальные законы). Этим занимаются не только топ-менеджеры, но управляющие финансами и даже юристы.

Сложные решения компания превращает в программы — это обычно результат совокупных усилий всех перечисленных. Программа — сложный набор шагов. Например, «наладить производство кирпичей» означает не только построить фабрику (проект), но и закупить и смонтировать оборудование (проект), найти и обучить персонал (проект), наладить заранее рекламу и логистику (проекты). И все ради финансовой стабильности к определенному моменту (до того, как у компании кончатся деньги, а у ее кредиторов — терпение).

Все это мало волнует разработчиков. Они пишут код продукта, полезного заказчику (это сложно, и это единственная их настоящая задача).

В Скраме у них нет менеджера. Нет никого, с кого можно было бы спросить: «Какие гарантии, что уложитесь в такой-то бюджет и срок? Как у вас с рисками? Что будете делать, когда увидите, что проект отклонился от первоначальных прогнозов?»

Ответ простой и в духе: «Это все нас не волнует, у нас Скрам! Мы не ускоряем команду, не корректируем планы, которых нет. Мы просто смотрим на velocity (которую редко удается подсчитать) и ретроспективно сообщаем, к какому моменту, скорее всего, закончим. Хотите — сделаем вам прогноз в деньгах (сообщите нам ставки всех программистов — мы вычислим). У нас нет и не будет ответа на вопрос: „Что делать, если не успеваем?" Ответ один — будем резать бэклог. Вас что-то не устраивает? Это потому, что у вас организация не agile! Надо, чтобы была agile!»

Увы, быть agile для организации не означает перестать нуждаться в финансовом планировании. В балансировке портфеля. Скрам не умеет управляться с ограничениями внутри микрокоманды (5–7 человек). И вместо того, чтобы бороться с этим, предлагает масштабировать это на уровень организации в целом!

Действительно, вопросы к команде отпадут, если все от топа до юриста будут разделять одинаковые ценности — «имеет значение только продукт, хорошо удовлетворяющий потребности клиента...». Стоп! Но как они отпадут? А зарплата? Ее брать из тумбочки? А она там появится?

Зарабатывание денег требует учета сложных workflow. Ночной кошмар каждого топа — «кассовый разрыв», после которого команда, не получавшая два месяца зарплату, встает и уходит в другую компанию, где «нормальный Скрам и нет бардака». **Если фирма — это машина, то финансы — ее топливо. Топливо нельзя залить «когда получится» (иначе авто встанет на хайвее) или заливать по спринтам** (взаиморасчеты, как правило, устроены не по принципу time and materials, но даже если так, — получение денег с заказчиков часто процесс сложный). Никто не знает, какие проекты на самом деле выстрелят на рынке (особенно если у фирмы не один единственный продукт). Значит, нужно делать несколько разных продуктов, чтобы они друг друга уравновесили. И завершать их придется не «когда получится и заказчик доволен», а в определенных пределах. Иначе разваливается вся фирма, все ее стратегическое и финансовое управление.

Agile на уровне организации (Scaled Agile framework и прочие) — это спринты спринтов, продукт овнеры продукт овнеров, которые хорошо смотрятся только на картинках (хотя нет, там тоже плохо). Добавить сюда хитро придуманные «трайбы» и «гильдии» (tribe & guild), которые при внимательном рассмотрении подозрительно напоминают «переливание из пустого в порожнее» и «усложнение простого» под красивым (немного ребячливым) соусом. А в жизни они просто не могут работать так, ибо операционная работа юриста, финансиста и топа очень часто — не итеративна и не такая, как хотелось бы евангелистам Скрам.

Попытка выбросить назойливых управленцев, которые пристают со своими сроками, бюджетами, рисками на помойку или поместить их в несопоставимую с долгосрочным workflow модель итераций — означает озадачить всю

организацию придумывать способ «как им делать то же, что она делала всегда, только чтобы теперь это было похоже на Скрам».

Нет, это не решает проблем. Гибкость всей организации — химера. Организация — не амеба, а, скорее, автомобиль. Потому и ездит по шоссе. В машине гибкий — водитель (он может свернуть, если хочет). А вот в прочности кузова, наличии колес и тормозов, и бензина нет никакой гибкости. Они либо есть, либо нет. Машина либо двигается, либо стоит в гараже, полуразобранная.

Забудь о «гибкости компании в целом» (пока точно не выяснишь, можно ли планировать зарплату всех сотрудников гибко, с горизонтом в один месяц и добывать ее в такие же сроки, а если разберешься с зарплатой — подумай о налогах, закупках, тендерах и подумай еще — я вот не подумал и расплачиваюсь).

При этом Скрам никак не «коннектится» во внешний мир**. Он сидит как сыч и настаивает, что будет играть по своим правилам** (не трогайте нас, пока не кончится спринт, меняйте только бэклог продукта). И отказывается давать оценки заранее. Сообщать о вероятности рисков. Что-то предпринимать, если первоначальные сроки (их в Скраме нет) или сделанные самой командой прогнозы по velocity будут отличаться от первоначальных. **Аргумент тот же — меняйте бэклог и становитесь там уже гибкими**. Наше дело работать в своем режиме (а в случае токсичной команды ее velocity, как правило, еще и уменьшается со временем, ибо «концов» найти все равно невозможно).

Управление проектами, напротив, изначально совместимо с портфельным и программным менеджментом. Оно чувствует себя частью организации и готово принимать и отдавать ей всю информацию (будь то план по закупкам, срокам, рискам, или что-то еще). Скрам требует от всей организации сперва тоже стать Скрам (что невозможно или не полезно), или не приставать с глупыми вопросами вовсе.

Когда пришло осознание, что запуск Скрам требует глобальных подвижек во всей организации, — нужно было идти к высшему руководству и советоваться. Не стремиться «продавить и переубедить упертое начальство», а попробовать осознать, как и почему оно управляет фирмой иначе, не в agile-стиле. И быть готовым признать, что не всякой организации нужно организовать работу посредством Скрама-Скрамов. И если без этого не получается реализовать свой конкретный проект — возможно, рассмотреть другие подходы.

Навязанные консультанты

Скрам продавал нам консалтинг, а мы не замечали

О том, что Скрам спроектирован под консалтинг. В нем есть и «буква» (роль Скрам-мастера) и «дух» (породил явление agile-куочей), убеждающие вас в необходимости купить труд консультантов.

Ко мне приходили agile-коучи. Я даже не заметил, как это началось.

За предыдущие 8 лет в управления проектами не помню ни одного консультанта, который потребовался бы за деньги и со стороны, чтобы подтянуть команду. Я все мог сделать сам. Почитать, выяснить, спросить.

В Scrum Guide в моем распоряжении было 20 страниц. И море спец. литературы. Почему-то во многих книжках, что **попадались, были веселые картинки и мало конкретики.** Той конкретики, что могла бы предотвратить появление моих записок.

Команда объяснила мне: все дело в том, что работаем своим умом, не имея опыта. А нужен специальный человек. Черт побери, почему для управления проектами по PMBoK в 1000 страниц такой человек был не нужен? А для 20-страничного Scrum Guide сразу понадобился.

Тогда я еще не понял, что если принцип описан чересчур просто, то это понижает входной порог, но чудовищно задирает и удорожает применение в дальнейшем. Гайд сам на это туманно намекает фразой о «простом для понимания и сложном для совершенного овладения».

Один мой друг занимается разработкой игр и говорит тоже о гейм-редакторах. Какой-то простенький конструктор позволяет тебе «сразу сесть и начать пилить игру, вообще не зная кода». Но не дай бог возникнет нужда усложнить логику — придется допрограммировать не заточенный под это движок и пользоваться костылями. И тут без советов «бывалых» не обойтись. Забудь про общеизвестные туториалы (их пишут только для тяжелых профи-решений, которые ты не захотел осваивать с самого начала). Теперь твой путь — кочки и самодельные тропинки.

Скрам — простой конструктор, в который легко начать играть. И пока ты хочешь «не эти ваши диаграммы Ганта», а просто «бэклог-спринты-берндаун-чарты и погнали» — все хорошо. Однако реалии заставляют все время подключать эту систему к аппарату «искусственной вентиляции легких». Со временем это требует все больше трудозатрат (как правило, сопоставимых с усилиями

по освоению «тяжелых» подходов”). И команда убеждается, что без помощи со стороны — не обойтись.

Рано или поздно поймешь, что **трудозатраты стали больше, чем если идти по пути освоения специализированного тяжелого подхода с самого начала.**

У нас появился agile-коуч. Его привел Скрам-мастер. Получилось как бы само собой.

В Скрам гайде всего четыре роли. Одна из них (Скрам мастер) — «узаконенный внутренний консультант». Он «...несет ответственность за продвижение и поддержку Скрама...» и «коучит Команду Разработки...». И, в общем, ясно, что не справляется. Да и как ему справится, если этот самый мастер сам из команды разработки? Самый заряженный, но не глубоко-опытный в построении Скрам в отдельно взятой команде.

Появился этот человек. О, какой это был парень!

Модный, уверенный, но не высокомерный. Опытный, но не занудный. Легко сходился с людьми (даже со мной), собирал их, обсуждал проблемы.

Чуть позже я понял, что работа коуча определяется его названием. Это не тренер и не консультант. Он коуч (почти «ментор» или «психоаналитик», но вовсе не «решатель проблем»).

Этот человек не может навести порядок на улице, где бегает толпа сумасшедших с навязчивыми идеями (ловить буйных психов будет врач с милицией), но, удобно расположив пациента на кушетке, окажет помощь тому, кто сам ее попросит. Причем вся помощь — не советы. А вопросы. «*Правильно заданные* вопросы», — обычно поправлял меня наш чудо-парень.

В духе: «Зачем ты делаешь вот так?», «Как ты думаешь, есть еще способ сделать это по-другому?», «Кто может знать ответ на этот вопрос?» — и так далее. Мягко и, зачастую, не вникая в специфику твоей работы, подталкивает тебя самого к поиску правильных решений. Подталкивает тех, кто хочет меняться, но беспомощен перед токсичными и самодостаточными (помнишь главу «Командная ответственность»?).

Не помню, сколько мы ему заплатили. Помню, что зря ждал от него изменения ситуации. Ребятам коуч нравился. Но не мог сделать токсичную команду не токсичной, не мог предложить нам разрабатывать продукт с большим количеством пользовательской функциональности вместо того, который действительно был нужен заказчику. Словом, не мог изменить ничего, что не было в его силах.

Коуч для нас, чаще всего, был человеком с вопросами. И редко — с ответами. И вовсе лишенный ответственности (он не гарантировал счастья и не мерил

собственную эффективность, он говорил). Так в нашей команде вместо «нуля» (как при проектном управлении) появилось сразу два консультанта (Скрам-мастер и agile-коуч), и я даже не заметил, как это произошло. Зато помню, что забыл заложить затраты на коуча в burndown-chart по деньгам. А потому упустил эти затраты из виду.

Не повторяй моих ошибок. Помни: Скрам подталкивает тебя к покупке консалтинга. В противном случае сообщество скажет, что «не надо было пытаться самому», «надо было звать профи» или вообще «построение agile в команде должно начинаться с построения agile во всей организации». Порой наш консультант намекал, что некоторые наши проблемы решаются внедрением scaled agile framework, но, поглядев в глаза, тяжело вздыхал:

«Вы пока к нему не готовы». Мы действительно были не готовы. Ради разработки продукта менять всю организацию и кратно увеличивать расходы на коучей.

Никто в индустрии не одобряет наших долгих попыток справится самим и «внедрять agile без agile всей организации». Но, оглядываясь назад, я понимаю, что и сейчас повторил бы именно эти шаги. Потому что это разумный способ для всякого работоспособного фреймворка. Менеджмент — не rocket science.

Нужно было напомнить себе золотое правило работы с консультантами — критерии эффективности. Зачем мы их зовем и как проверим, что их работа (работа коучей) эффективна. Держаться этих критериев и не соглашаться на «постепенное выращивание и вызревание команд и всей организации чужими руками».

Планирование каждый день — это очень много

Планы. Вот что нас привлекало!

О том, как выяснилось, что планирование в Скраме занимает больше времени, чем в проектном управлении.

«Скрам не тратит времени на планирование», — объясняли нам коллеги, почитавшие о нем. Ведь, пока вы тратите силы на детальный план (техническое задание и графики), вы, во-первых, занимаетесь тем, что не нужно заказчику, а во-вторых, жизнь меняется и уходит вперед. Спустя пару недель ваши планы нужно переделывать, вы не успеваете произвести даже минимально полезную «хотелку» пользователя. И мы соглашались.

Черт, я должен был уже тогда посчитать.

В то время как любой менеджер проекта потратит на первоначальное планирование проекта от 2 до 5 дней (если говорить о софтверной разработке и маленькой команде) и будет эти планы потом постоянно контролировать.

В Скрам того самого «недельного» первоначального этапа нет. Сформировали бэклог и в бой.

Стоп!

Сформировать бэклог — это уже планирование. Собрать «хотелки». И оценить их трудоемкость относительно друг друга в пресловутых story points. И еще приоритизировать (о, волшебный владелец продукта!). Так сколько на это нужно времени? Допустим, поменьше, чем 5 дней.

Но в Скраме нужно еще кое-что.

Примерно 0,5 дня = планирование каждого спринта.

Примерно 0,5 дня = демонстрация и ретроспектива каждого спринта.

Ежедневные стендапы = в идеале 15 минут; на практике до получаса, примерно. За недельный спринт набегает еще 1 час 15 минут минимум (15 % дня).

Поддержка бэклога = 10 % от всего доступного времени Скрам-команды! (о, боже, как хитро спрятана эта фраза про постоянное перепланирование в самую глубину абзаца про бэклог продукта… ее будто нарочно нет в разделе про спринты, обзоры и ретроспективы — мы фокусировались на них и не заметили). Краеугольное словосочетание! Заметь мы его раньше, — я бы развернулся и ушел. Десять

процентов от всего времени на переосмысление и приоритизацию бэклога всей командой! Десять… Прости, сейчас я успокоюсь.

Давай посчитаем.

Если наш спринт равен неделе (5 × 8 = 40 часов), а сумма всех наших потерь по времени такая, как описана выше (0,5+0,5+0,15+40 × 0,1) = 1,5 дня (12 часов), то, значит, потери на планирование каждую неделю составляют 30 %!
Тридцать проклятых процентов! Вот наш Скрам без планирования.

Я в отчаянии. Обнаружив это (где был мой листок-карандаш и сложение столбиком раньше!), мы начали удлинять спринт (да, теперь я знаю, почему никто не делает спринты длинной в неделю = извини, дорогой заказчик, хотел быстрый результат — подождешь, ибо иначе у нас тут затраты на несчастное планирование запредельные, Скрам же!). В двухнедельных спринтах картина чуть лучше, но и затраты на планирование и ретроспективы возрастают (в месячных спринтах, например, демонстрация и ретро могут длится целый день). В процентах это чуть лучше. Мы смотрим в сторону месячных спринтов. Но на нас грозно посматривает заказчик. «Ребята, а как же регулярные поставки, постоянное вовлечение меня, постоянный прирост полезных Результатов?» Да любой менеджер проектов предложит мне ежемесячные встречи и промежуточные демо! Причем тут ваш Скрам? Ради чего я пожертвовал возможностью прогнозировать сроки на старте проекта? Ради чего играю в вашу игру, если встречаемся мы раз в месяц (а в промежутке мне даже видеть Скрам-команду запрещено, не всякий аналитик со мной разговаривает, ибо «дергать команду внутри спринта» этот ваш Скрам прямо запрещает)?

Мне все сложнее отвечать на такие вопросы. У нас Скрам, но ради чего?.. Парни говорили мне, что ради сокращения ненужного планирования. Которого сперва оказалось примерно 30 % каждую гребаную неделю. Каждую! А не 100 % в первую с уточнением планов по необходимости.

Нужно было все внимательно посчитать и просто удалить этот пункт с повестки. Скрам не сокращает время на планирование. Что бы ни говорили обложки книг-бестселлеров от создателей фреймворка.

Не повторяй моих ошибок. Запуская Скрам, не думай, что сможешь не планировать. Ты лишь снимешь с себя ответственность за сроки, а планировать будешь не меньше. И привлекать к этому придется всех. Треть времени команда будет тратить на разговоры, не на код.

Невозможный владелец продукта

Хороший владелец настолько редок, что должен быть занесен в красную книгу

О том, как мы обнаружили, что хороший владелец продукта практически не встречается в дикой природе, а его роль критична. И не смогли найти подходящего (и ты тоже не сможешь).

Скрам настолько программисто-центричен, что отменяет остальные возможные роли (все, кто не в состоянии быть разработчиком, должны искать себе применение вне команды). Еще он низводит постановку задач до одного человека — владельца продукта.

Скрам подчеркнуто не вникает в то, откуда возьмутся требования. Как оценить их значимость и взаимосвязь. Как оценить влияние на бизнес. И как вследствие всего перечисленного приоритизировать.

Отговаривается в духе: «Заведи продукт овнера».

Пусть это будет ваш «главный по требованиям» и плевать, давай лучше снова рассуждать про будни разработчиков и Скрам-мастеров.

Где такого взять?

Использовать оставшегося без работы аналитика? Причем лишив привычных инструментов (мы не дадим ему проводить долгое обследование, что-то моделировать и выяснять). Возможно, аналитик успеет до запуска работ провести какой-то «предпроект» и укомплектовать бэклог первоначальными «хотелками». Но Скрам сконструирован таким образом, что владелец продукта должен «знать» ответы на все вопросы по продукту (а не искать их в ходе мучительных аналитических расследований).

Возможно, переложить на заказчика (ему надо — вот пусть он и направляет команду)? Супернечестный подход. Единственную роль хоть с какой-то ответственностью мы не просто вывели из команды — мы выбросили ее вообще из своей организации. Мы — «гномы, добывающие руду», об остальном пусть думает заказчик. Формулирует-приоритизирует. Все понимают, почему заказчику трудно с нуля написать хорошее ТЗ. Но отчего-то мало кто сомневается, что продукт овнер от заказчика хорошо управится с бэклогом. Не управится.

Продукт овнер в Скрам должен сочетать в себе аналитика, представителя бизнеса и маркетолога (знать рынок и конкурентов, понимать альтернативы, знать «как у них» и «что срабатывает, что нет». Для Скрама нет проблемы

склеить это в роль и поручить одному человеку — ему вообще плевать на не-разработчиков (и не-методологов, вроде Скрам-мастера).

Продукт овнер — роль с огромным входным порогом. Шансы найти его (маркетолога-аналитика-бизнесмена в одном лице) в разы ниже, чем грамотного управленца, который разделит задачу и скоординирует ее выполнение разными людьми. На стороне заказчика такие не водятся — и не надейся. А на рынке ты не найдешь людей, знающих бизнес заказчика изнутри.

Грамотный продукт овнер — это занесенный в красную книгу и почти мифический персонаж, но обязательный для правильной работы фреймворка.

Надо было первым вопросом при внедрении Скрам задавать себе: «Где мы возьмем продукт овнера?» Есть ответ — думаем дальше (см. главы с первой по последнюю), ответа нет — не надо иллюзий, что он «найдется или наймем». Это возможно, но нелегко. И «ахиллесова пята» всего Скрам.

Заключение

На этом рукопись практически обрывается

Несколько страниц отсутствует. А ближе к концу дневника — краткое обращение.

> *«Пусть мой опыт послужит для тебя уроком. Я писал не критику, но предупреждение — Скрам годится не всегда. Он перекручен, переоценен и перепродан.*
>
> *Не применяй Скрам так, как я.*
>
> *Не запускай проект, пока не будешь точно знать, что возразить и противопоставить каждому моему тезису в каждой главе.*
> *Не ломай себе карьеру».*

—

Сайт автора: www.pmlead.ru

Больше об управлении проектами: http://education.pmlead.ru

Больше о сравнении Scrum, Kanban и PMI: http://itmethods.pmlead.ru

Telegram-канал автора: https://t.me/selihovkin