Илан Голдштейн

SCRUM БЕЗ ОШИБОК

Инструменты, техники и советы для тех, кто работает по Agile

Москва
«Манн, Иванов и Фербер»
2020

Информация от издательства

Научный редактор Мария Князева

Издано с разрешения Pearson Education (Addison-Wesley Professional)

На русском языке публикуется впервые

Возрастная маркировка в соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ: 16+

Голдштейн, Илан

Scrum без ошибок. Инструменты, техники и советы для тех, кто работает по Agile / Илан Голдштейн ; пер. с англ. Марии Князевой, Александра Семенова. — М. : Манн, Иванов и Фербер, 2020.

ISBN 978-5-00146-306-1

Широкое распространение Scrum объясняется его кажущейся простотой, однако его внедрение проходит далеко не так гладко, как ожидают многие. Опираясь на свой обширный опыт, сертифицированный scrum-тренер Илан Голдштейн раскрывает фундаментальные механизмы Scrum и его сущность как фреймворка. В этой книге каждый найдет решение своих проблем и конкретных scrum-задач. Даже те, кто разобрался во всех scrum-тонкостях и держит все под контролем, найдут для себя новые инструменты и добавят их в свой scrum-арсенал. В этой книге Илан Голдштейн собрал 30 лайфхаков. Написаны они таким образом, чтобы их можно было читать и использовать автономно, независимо от других частей книги. Обращайтесь к тем из них, которые больше всего отвечают вашим потребностям. И помните, что в этой книге отражена только реальная scrum-практика, протестированная в боевых условиях.

В этой книге Илан Голдштейн представил столь необходимое руководство для scrum-команд. Тот факт, что Scrum является фреймворком, часто используется для оправдания перекройки его фундаментальных механизмов, по итогам которой то, что когда-то было Scrum, меняется с точностью до наоборот и становится гораздо менее эффективным. Голдштейн проводит четкую границу между теми механизмами, которые могут быть адаптированы, и теми, которые должны оставаться неприкосновенными, чтобы вы могли достичь успеха.

Арик Коган,
бизнес-аналитик в Cougar Software

Илан Голдштейн заслужил верных последователей в agile-сообществе за свои более чем толковые советы и практические решения, которые обеспечивают командам реальные результаты. Я в восторге оттого, что он смог перенести свой опыт в книгу, насытил ее знаниями и сделал легкочитаемой. С нетерпением жду, когда начну использовать все его лучшие идеи на практике!

Пит Димер,
CEO Stormglass и автор The Scrum Primer

Хорошие книги предлагают вам советы, которым вы последуете, и «Scrum без ошибок» определенно из таких. Она подойдет как новичкам, так и опытным практикам, проявляющим здоровый интерес к Scrum, потому что знания, содержащиеся в этой книге, могут изменить правила игры. Илан Голдштейн делится своим обширным опытом, позволяя читателям получить хорошо изложенное и невероятно ценное представление об эффективных agile-практиках.

Кевин Остин,
agile-коуч и руководитель трансформации инвестиционного банка Fortune 50

Команда разработчиков программного обеспечения преуспевает, потому что у нее есть «правильные» люди, которым позволено делать лучшее, на что они способны. Понимание паттернов и антипаттернов (мой любимый антипаттерн — «спринт тестирования»), которое вы вынесете из этого руководства, поможет выяснить, кто именно является для вас «правильными» людьми и как помочь им работать эффективно. Эти лайфхаки заточены под людей, и поэтому они работают. Пусть ваша команда (и вся компания) прочитает книгу и обсудит ее не откладывая в долгий ящик.

Лайза Криспин,
соавтор книги «Agile-тестирование. Практическое руководство для тестировщиков ПО и гибких команд»

Эта книга — замечательное руководство для всех, кто использует Scrum при создании программного обеспечения. Большой ли у вас опыт

за спиной, или вы новичок на самом старте карьеры, вы найдете для себя что-то важное, возьмете на заметку и сможете незамедлительно применить. В легкой, непринужденной манере Илан описывает реальные проблемы, с которыми вы можете столкнуться при использовании Scrum, и дает практические советы по их решению.

Райан Доррелл,
технический директор AgileThought

Освежающий взгляд на Scrum. Илан выведет вас за рамки теории и поделится реальным опытом, предложит практические советы для успешного внедрения Scrum в вашей организации и достижения зрелости. Его выводы и философский подход позволят вам продвинуться в этой игре.

Джон Мэдден,
руководитель программы HotelsCombined

Делясь знаниями, которые он обрел за много лет, внедряя Scrum с разными командами, Илан щедро раздает практические советы, чтобы помочь вам поднять свои scrum-команды на новый уровень. С юмором, вниманием к деталям и историями из личного опыта он написал доступную и легко адаптируемую книгу рецептов Scrum, которую scrum-мастер может использовать в любой рабочей среде. Независимо от того, применяете ли вы Scrum несколько месяцев, или у вас многолетний опыт работы, эта книга предложит вам новые идеи, как решать любые задачи, стоящие перед вашей командой.

Лайза Вуд,
scrum-мастер и блогер в Sockets and Lightbulbs

Scrum не является решением. Решение станет ясным только тогда, когда вы пройдете путь инспекции и адаптации. Это путешествие не будет простым, и вы наделаете ошибок, настраивая Scrum в своей компании. Для меня книга Илана — доходчивая и простая инструкция: она дает вам понимание, инструменты, подходы и веру, чтобы поддержать вас.
Я очень ценю открытость Илана и то, как он делится своим опытом. Мне не обязательно соглашаться с каждой его техникой или идеей, да Илан и не пытается давать указания. Он призывает подумать, бросить вызов допущениям и помогает пройти по выбранному пути.

Мартин Кернс,
scrum-тренер и национальный лидер по Agile и инновациям в SMS Management & Technology

Большинство книг о Scrum посвящены теории и повествуют об отдаленных перспективах от третьего лица. Их трудно читать. Илан же создал нечто иное — книгу, которая напоминает разговор. Я кивал в знак согласия со всеми проблемами, которые он выявил, и смеялся над его комментариями, прагматичными, остроумными и мудрыми. Вы с большим сожалением отложите эту книгу, дочитав до конца.

Клинтон Кит,
scrum-тренер и автор Agile Game Development with Scrum

Илан проделал отличную работу. Эта книга дает глубокий анализ всего, что нужно, чтобы стать scrum-мастером, и описывает реальный опыт, который поможет вам в вашем scrum-путешествии. Отчасти это практические советы, отчасти сборник историй, которые автор искусно соединил в книгу, которую читать полезно и приятно. Я наслаждался книгой и с нетерпением жду экземпляра, чтобы поставить на книжную полку. Илан проделал замечательную работу.

Кейн Мар,
scrum-тренер, соучредитель и президент Scrumology.com

Если бы Scrum и Agile были просты, ими бы каждый занимался! Теперь, когда желающих так много, эта книга может стать виртуальным agile-коучем, которого мне так недоставало на ранних этапах своего scrum-путешествия. Илан — коуч мирового уровня, он наполнил свою книгу исчерпывающим набором решений ко всем общим вопросам и проблемам, которые обязательно возникнут, когда вы начнете менять свой мир работы по Scrum.

Крейг Смит,
agile-коуч и редактор InfoQ

Scrum обманчиво прост, а как говорят, сделать средненько проще всего. Я и сам попадал в разные ловушки. В своей книге Илан показывает нам, как добиться успеха в Scrum. Книга устроена так, чтобы вы сразу могли найти самую полезную для себя информацию. Каждая из множества глав наполнена смыслом, и ее приятно читать. К этим рекомендациям нечего добавить.

Йенс Мейдам,
руководитель отдела разработки Zahnärztekasse AG

Ценная тактика, инструменты и советы Илана отражают его собственный опыт, приобретенный в тяжелых условиях. Это не теоретическая книга о Scrum, а скорее ориентированный на практиков гид по работе со Scrum. Захватывающее чтение. Эта книга — необходимое дополнение к основной литературе по Scrum и идеальное продолжение моей книги «Основы Scrum»!

Кенни Рубин,
управляющий директор Innolution, LLC и автор книги «Основы Scrum. Практическое руководство по гибкой разработке ПО»

Книга Илана не обычное руководство о том, как использовать Scrum на рабочем месте, а скорее набор практических советов, которые охватывают все аспекты создания самоорганизующихся высокопродуктивных scrum-команд. Это очень интересное, приятное и содержательное чтение для всех scrum-практиков.

Майкл Рембах,
менеджер по разработке приложений, Transport for New South Wales

Если Scrum Guide — книга правил, то книга Илана Голдштейна — это инструкция от экспертов. Илан раскрывает все, даже мельчайшие, секреты, которые следует знать каждой команде, чтобы эффективно применять Scrum. Начиная от длины спринта до разделения работы и относительной оценки, Илан продирается сквозь «серые зоны» в Scrum, предлагая мудрые советы и рекомендуя всегда действовать по обстоятельствам.

Рене Траутон,
agile-коуч и автор Agile Forest

Мне особенно нравится стиль повествования. Автор советует пропускать те или иные разделы, чтобы найти темы, интересные именно вам. Илан освобождает нас от идей, которыми мы были одержимы раньше, и предлагает свежий взгляд на вещи. Очень ценно!

Рон Джеффриз,
соавтор Agile Manifesto и основатель xprogramming.com

ВСТУПИТЕЛЬНОЕ СЛОВО МАЙКА КОНА

В духе этой книги я сразу перейду к делу: купите ее. Уверяю вас, собранные здесь лайфхаки чрезвычайно полезны.

По собственному опыту мы знаем, как опасно бывает срезать путь. Фильмы ужасов начинаются с того, что группа подростков хочет срезать дорогу и решает пройти через лес темной ночью. Водитель в семейной поездке выбирает, как ему кажется, короткую дорогу, а потом ему годами напоминают о том, как он ошибся. Нам твердят, что «нет никаких коротких путей к успеху», что он зависит от упорства и мастерства.

Да, в жизни многие короткие пути неэффективны. Но лайфхаки этой книги другие. Они работают.

Впервые я познакомился с Иланом Голдштейном в сети, когда интернет-поиск привел меня к его блогу scrum-лайфхаков. К тому времени у Илана их было немного, все они оказались чрезвычайно полезными и к тому же забавными. Каждый лайфхак написан невероятно остроумно.

Не нужно быть гением, чтобы убедиться: Илан знает, о чем говорит, поэтому я спросил, не думал ли он написать книгу лайфхаков. И вот вы держите ее в руках. В этой книге Илан предлагает тридцать советов, охватывающих все аспекты внедрения Scrum. Он предлагает советы по началу работы, требованиям, оценке и планированию. Вы найдете советы о качестве и метриках, участниках команды и ролях, об управлении вашими начальниками и о непрерывном улучшении. Если вы

столкнулись с подобными проблемами, вероятно, у Илана найдется лайфхак, который поможет вам.

Илан был там и сделал это. Его советы основаны на опыте работы scrum-мастером и сертифицированным scrum-тренером. Пути, о которых рассказывает Илан, — это маршруты, по которым он путешествовал. Они сугубо практические, а не теоретические. Мне очень нравится, что он не боится занять определенную позицию. Слишком много книг чересчур часто выдают стандартный ответ консультантов: «Зависит от…» Ничего подобного вы здесь не найдете.

Будь вы месяц, год или десятилетие в Scrum, вы найдете здесь лайфхаки, которые помогут вам совершенствоваться. Желаю вам всего наилучшего в scrum-путешествии. Знаю, вы доберетесь до цели быстрее, следуя указаниям из этой книги.

Майк Кон,
соучредитель Scrum Alliance и Agile Alliance,
автор книги «Scrum. Гибкая разработка ПО»

ПРЕДИСЛОВИЕ

«Ах, так это Дилберт наоборот![1]» — воскликнул мой друг-психиатр, когда я вкратце рассказал ему про Scrum. (Нет, я не посещал его, чтобы справиться со стрессом от написания моей первой книги, в то время как мой первый ребенок не давал мне спать!) Я рассмеялся: мой друг так просто и изящно передал суть Scrum. Кроме того, я только что нашел эпиграф для книги!

Scrum и его agile-собратья составляют следующую эволюцию профессиональной деятельности и корпоративной культуры. Безусловно, не я один пришел к выводу, что это, возможно, величайший сдвиг со времен появления теории управления и рациональной организации труда, также известной как тейлоризм. (Кстати, знаете ли вы, что Генри Гантт[2], известный своей полосатой диаграммой, был учеником Тейлора?)

Scrum отбрасывает диктаторский, заточенный на эго босса подход к управлению, в рамках которого люди уподобляются легко заменяемым винтикам в механизме с четко определенными процессами. Scrum рассматривает команды как группы ответственных, преданных делу и свободомыслящих людей. Людей, которые, если им дать возможность, будут работать оптимальным образом для достижения максимального результата.

Это невероятно волнующе. Приятно идти в авангарде этих перемен вместе с пионерами Scrum, которые возглавляют наступление. Без

сомнения, в будущем нынешнее время будет признано периодом эпохальных перемен в организации труда.

ПОЧЕМУ Я НАПИСАЛ ЭТУ КНИГУ?

Я вспоминаю разговор с Мартином Кернсом — другим австралийским scrum-тренером. Он обратил мое внимание на то, что нравится мне это или нет, но люди прочитают мою книгу (со множеством тактических приемов, инструментов и советов) и воспримут ее как официальное руководство пользователя — то, чему они должны неукоснительно следовать. Это лишь усилило мое беспокойство: могу ли я давать конкретные советы, которые «прорезают» теорию и переходят сразу к сути, избежав при этом директивности? Я нашел для себя ответ: рассказывая о *лайфхаках в этой книге*, я делюсь с вами *одним из подходов*, а не директивно излагаю жесткий регламент внедрения Scrum. «Разве может быть больше одного подхода к Scrum?» — можете удивиться вы.

Этот момент прекрасно объясняет Кеннет Рубин[3] в своей книге Essential Scrum[4]:

Scrum основан на небольшом наборе ценностей, принципов и практик (в совокупности это фреймворк Scrum[5]). Организации, внедряющие Scrum, должны принять его во всей полноте, однако это не означает, что во всех компаниях Scrum будет реализован одинаково. Скорее у каждой фирмы будет своя уникальная реализация фреймворка Scrum, основанная на тех подходах, которые она выберет для реализации scrum-практик… Подход — это и есть конкретная реализация scrum-практик.

Есть много разных подходов со своими приемами, инструментами и советами, которые вы можете и должны изучить. Тем не менее я надеюсь, что лайфхаки из этой книги помогут вам проанализировать ситуацию и предложат проверенные варианты.

Я написал эту книгу, потому что набил много шишек, спотыкаясь о камни преткновения и разбивая голову о кирпичные стены. Честно говоря, внедрять Scrum действительно непросто! Все кажется понятным в теории, но, черт возьми, попробуйте эффективно его внедрить и использовать. Это будет чем угодно, только не тривиальным занятием. Спустя годы и после работы с несколькими командами я наконец начал видеть отдачу. Я создал легко адаптируемую (и эмерджентную) книгу рецептов Scrum, которая работала в самых разных командах и организациях. Я понял, что мои с трудом добытые знания могут выручить других, работающих в схожих условиях.

Беседуя с Мартином, я получил полезный совет. Заметив, что я недавно впервые стал отцом, он спросил меня, считаю ли я, что должен ограждать маленькую Эми от любой ситуации, в которой она могла бы упасть и пораниться. Мое сердце говорило: «Да, именно так! Я

не позволю причинить вред моей малютке». Однако мой мозг осознал, что даже тем, о ком вы заботитесь, нужно позволить споткнуться (иногда), чтобы они на собственном опыте узнали, что работает, а что нет. При этом вы всегда хотите быть рядом с ними, чтобы успокоить их и дать несколько полезных советов на будущее. Итак, эта книга просто доброе напутствие перед ваши scrum-путешествием, чтобы впредь у вас возникало поменьше порезов и ушибов. Полагаю, вы уже дали Scrum шанс и, вероятно, имеете старые раны. И если хоть немного повезет, эта книга защитит вас от следующего раунда ударов.

Если вы новичок в Scrum и ненавидите порезы и ссадины, приступайте к чтению без колебаний. Возможно, некоторые советы помогут вам избегать травм какое-то время. Однако имейте в виду: каждый проект уникален, каждая команда тоже, как и каждая компания. И если вы ожидаете успеха от каждого совета, я хотел бы скорректировать ваши ожидания прямо сейчас — до того, как вы разочаруетесь. Честно говоря, нет волшебного подхода, который бы срабатывал везде и всегда.

Я надеюсь, что те из вас, кто чувствует, что разобрался со всеми тонкостями Scrum и держит все под контролем, найдут новые интересные инструменты и добавят их в свой scrum-арсенал.

ПАРА ГИПОТЕЗ

Большинство уроков этой книги я извлек, работая scrum-мастером в нескольких компаниях, поэтому основной аудиторией книги являются именно они, scrum-мастера. Тем не менее эта книга, безусловно, будет полезна и для других, включая владельцев продуктов, разработчиков и стейкхолдеров. Даже моя жена-адвокат, нисколько не интересующаяся программным обеспечением, сочла ее полезной и любопытной, — вот видите!

Скорее всего, вы не новичок в Scrum и прочитали несколько книг. Возможно, даже посетили курсы и некоторое время пытались работать по Scrum. Если это про вас, моя книга поможет вам перейти на новый уровень эффективности и зрелости в Scrum за счет расширения и развития набора scrum-инструментов.

Если вы новичок в Scrum, ничего не бойтесь! Эта книга содержит много глав (или лайфхаков), которые вы легко одолеете. Тем не менее я рекомендую вам прочитать еще несколько кратких обзоров Scrum:

- *Core Scrum*[6] от Scrum Alliance;
- *The Scrum guide*[7];
- *The Scrum Primer*[8].

Для более глубокого знакомства со Scrum я настоятельно рекомендую приобрести книгу Кеннета Рубина Essential Scrum[9].

КАК ПОЛЬЗОВАТЬСЯ ЭТОЙ КНИГОЙ

Я не писал эту книгу последовательно, главу за главой, поэтому вам не обязательно читать ее с начала и до конца. Вы можете легко перемещаться между разделами, не рискуя ничего упустить.

Лайфхаки написаны таким образом, чтобы они быстро и легко усваивались. Моя цель — изложить их максимально просто и доступно, чтобы даже в пылу сражения они пришли вам на помощь. В мирное время это будет полезным и занимательным чтением (например, пока вы стоите в очереди у офисной микроволновой печки, чтобы разогреть обед).

Вы можете рассматривать «Scrum без ошибок» как книгу рецептов (или книгу заклинаний, если так уж случилось, что вы волшебник или ведьма) — просто пролистайте до следующего лайфхака, решите, какие ингредиенты работают у вас, и не стесняйтесь добавлять специи… на свой страх и риск. Если повезет, у вас появится полезный и практичный подход к решению конкретных scrum-задач.

МОИ ЦЕЛИ

Эта книга призвана не только помочь Scrum работать на вас, благодаря ей вы сумеете поднять scrum-команды на новый уровень эффективности и зрелости. Большую часть того, что я написал, вы не найдете ни в одном scrum-руководстве (даже в вашем учебном пособии scrum-мастера). Зато я описал реальную scrum-практику, протестированную в боевых условиях.

Стоит повторить: я не ожидаю, что вы будете следовать всему написанному в этой книге до последней буквы. Тем не менее я рекомендую вам экспериментировать, проверяя эти приемы, инструменты и советы и адаптируя ваши процессы, чтобы увидеть, приводит ли это к улучшениям. В идеале вы не только выиграете от моего подхода, но и сможете развить его и преподать мне пару уроков!

Глава 1

ЗАПУСК SCRUM

Сделать первый шаг к чему-то новому всегда непросто. Вопросы «с чего начать?», «как начать?» и самый важный «почему нужно начинать?» часто служат своего рода тормозами, отрицательно влияющими на решение компании внедрить такой фреймворк, как Scrum.

Ответить на эти непростые вопросы и отважиться сделать первый шаг помогут три следующих лайфхака.

Лайфхак 1 «Scrum на поле» — это руководство, которое поможет «продать» Scrum тем, кто в вашей организации связан с его внедрением.

Лайфхак 2 «Хрупкий Agile» выделяет типичные ошибки при переходе компании на Scrum и ловушки, на которые следует обратить внимание в первые дни вашего scrum-путешествия.

Наконец, лайфхак 3 «Творческий комфорт» предлагает ряд мер для создания рабочей среды и корпоративной культуры, способствующих появлению здоровой scrum-команды.

ЛАЙФХАК 1. SCRUM НА ПОЛЕ

Scrum действительно легко «продавать», и, должен признать, получить добро на внедрение Scrum проще простого. Ну, может, это и не пара пустяков, но вряд ли кто-то станет оспаривать тезис Кена Швабера, одного из создателей этого подхода: «Scrum, похоже, преодолел пропасть и теперь скорее мейнстрим, чем оригинальная новинка»10. Этот прогресс, безусловно, сделал жизнь нового поколения scrum-евангелистов проще по сравнению с пионерами этого движения. По крайней мере, нам больше не приходится сталкиваться с изумлением аудитории, рассказывая о подходе к разработке программного обеспечения с помощью терминов, заимствованных из регби!

Давайте рассмотрим, что же подразумевается под Scrum. Многие (даже так называемые scrum-мастера) думают, будто это аббревиатура, но на самом деле так называется схватка вокруг мяча в регби (и там все буквы строчные).

Для тех, кто не знаком с этим спортивным приемом, поясним: ватага здоровенных мужиков из противоборствующих команд вцепляются друг в друга руками, как элементы пазла, и изо всех сил стараются отбросить соперников к их воротам (к зоне подсчета очков). Именно эта концепция сплоченной самоорганизующейся совместной работы команды и дала новое agile-наполнение спортивному термину. Впервые, в 1986 году, термин Scrum описали Хиротака Такеучи и Икуджиро Нонака, которые считаются отцами-основателями Scrum11.

Поскольку я родился в стране, где регби очень популярно, то не раз наблюдал за такими соревнованиями. Схватка напоминает атаку древней спартанской фаланги, где каждый воин укрывался под щитом соседа (см. рис. 1.1). Ее не победить, если поддерживается дисциплина, а товарищи по команде действуют сообща.

Рис. 1.1. Если scrum-команда действует дисциплинированно, как спартанская фаланга, она непобедима

ОХОТНИКИ ЗА ОБОРОТНЯМИ

Убеждать стейкхолдеров в эффективности Scrum — мое любимое занятие! Почему? Когда я говорю о прозрачности, быстрой поставке бизнес-ценности, сокращении потерь и снижении рисков, у них загораются глаза. А после того, как я выдвигаю радикальный тезис — изменения должны рассматриваться не как препятствия, а как возможности, — по аудитории проносится вздох облегчения.

Однако следует признать, что мы, scrum-энтузиасты, не охотники за оборотнями с обоймой серебряных пуль. Реальность такова, что, хотя концепция Scrum проста и интуитивно понятна, ее успешная реализация нелегкое дело.

Так что же делает Scrum самым популярным фреймворком Agile? Ответ на этот вопрос зависит от того, к кому вы обращаетесь — к scrum-команде (включая scrum-мастера, владельца продукта и разработчиков) или главным стейкхолдерам (назовем их спонсорами проекта). В оставшейся части этого лайфхака мы сфокусируемся на критических точках, имеющих ключевое значение для двух этих групп.

Майк Кон, один из основателей некоммерческих организаций Scrum Alliance и Agile Alliance, говорит о возможностях Scrum следующее:

Scrum — это гибкий фреймворк, который позволяет сосредоточиться на поставке максимальной бизнес-ценности в кратчайшие сроки[1][2].

Хорошо сказано! Теперь давайте детально разбираться и объяснять обеим нашим группам, что это значит для каждой из них.

SCRUM-КОМАНДА

Начнем с обсуждения ключевых преимуществ, которые мы можем предложить scrum-команде, состоящей из scrum-мастера, владельца продукта и разработчиков.

Меньше переключения контекста

Общепринятое похлопывание по плечу с очередной просьбой поработать над чем-то «более срочным» отныне исключается. Scrum предоставляет концепцию защищенного спринта (которую я люблю называть фиксированной гибкостью). Защищенный спринт позволяет разработчикам полностью сосредоточиться на том, что они обязались выполнить на встрече по планированию спринта (см. лайфхак 8), а также предоставляет владельцу продукта возможность более широко модифицировать бэклог продукта на протяжении всего проекта.

Устойчивый темп

Не стану лукавить и говорить, что, как только вы начнете использовать Scrum, никогда больше не будете засиживаться на работе допоздна. Тем не менее Scrum — это работа в стабильном устойчивом ритме, который позволяет избежать назначаемых в последнюю минуту, наспех организованных и ведущих к ошибкам встреч. Scrum уничтожает традиционную практику героизма поздних рабочих вечеров и выходных, посвященных доказательствам преданности общему делу.

Кеннет Рубин прекрасно это объясняет:

Один из основополагающих принципов Scrum гласит: «Все участники команды должны работать в устойчивом ритме!» (Больше никаких маршей смерти!) При этом они обеспечивают создание продуктов мирового уровня и поддерживают здоровую и приносящую радость атмосферу на работе13.

Больше никакого диктата

Менеджеры проекта с диктаторскими замашками, любящие раздавать указания, больше не должны определять, кто, что и когда делает. Одна из флагманских целей Scrum — создание самоорганизующихся команд, которые и будут определять, как именно выполнять работу, потому что именно они ее и делают!

Нет больше разделения на «мы» и «они»

Хотя Scrum уважает и ценит уникальность любого человека, личные рекорды отступают перед достижениями команды. Уходит в прошлое специфический мониторинг производительности сотрудников (каждого по отдельности), не говоря уже об установках на разделение «мы» и «они»

на различных этапах разработки. Благодаря Scrum каждый в команде максимально концентрируется на одной цели — реализовать то, что они обязались выполнить.

Специально выделенный «щит и бульдозер»

Для сфокусированного разработчика нет ничего хуже, чем необходимость иметь дело с интригами, отвлекаться на перерывы и преодолевать препятствия. Благодаря роли лидера-слуги — scrum-мастера (см. лайфхак 4) — команда разработчиков может сосредоточиться на том, что она делает лучше всего, — разработке отличного программного обеспечения. Scrum-мастер защищает команду от разрушительных внешних воздействий и решает проблемы, которые могут препятствовать прогрессу.

Надеюсь, теперь у вас есть команда, которую вам удалось убедить в преимуществах Scrum и которая готова его использовать.

СПОНСОРЫ ПРОЕКТА

А теперь давайте раскроем ключевые преимущества, которые получают наши главные спонсоры проекта.

Снижение рисков

У традиционного проекта по разработке программного обеспечения риск составляет 100%, а поставленная ценность — 0%, и так будет, пока не настанет день успешного релиза. Длительные 18-месячные циклы выпуска продукта по водопадной модели14 не могут дать осмысленной целостной картины или понимания ценности вплоть до самого релиза (см. рис. 1.2).

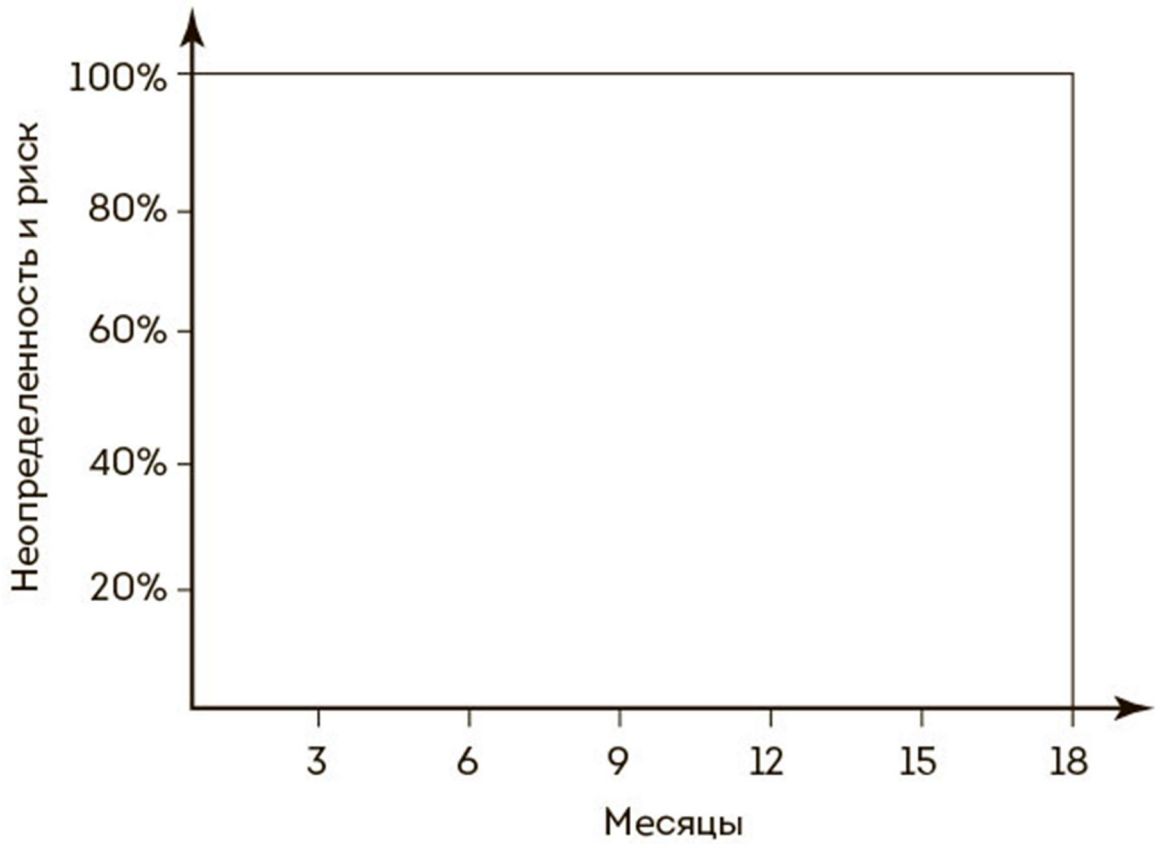

Рис. 1.2. Проекты, разрабатываемые по водопадной модели, сопряжены со 100-процентным риском вплоть до самого конца

Scrum-команда, обеспечивая высокую функциональность, поставляет клиентам истинную бизнес-ценность в течение нескольких недель (или дней), а не месяцев или даже лет. При этом благодаря более быстрым циклам обратной связи существенно снижаются риски.

Прозрачность, открытость и меньше неожиданностей

Прозрачность особенно актуальна для компаний, у которых спонсоры не имеют опыта разработки программного обеспечения. Для них ваша работа — черный ящик. Scrum основан на эмпирическом, наглядном управлении процессами, что делает прозрачность его основополагающим принципом. Это достигается за счет простых для понимания информационных источников (таких как доска задач — (см. лайфхак 21)), а также регулярных обзоров спринта, на которые приглашают всех желающих.

Непрерывное улучшение

Наряду с прозрачностью двумя другими столпами эмпирического управления процессом являются инспекция и адаптация. Эти важные элементы применяются как к продукту, так и к процессу его разработки, чтобы обеспечить его непрерывное улучшение по всем направлениям. «Инспекция и адаптация» — основная мантра Scrum.

Изменения — это возможности

Спонсоры проекта больше не испытывают досаду, приходя с гениальной идеей и желая добавить ее в бэклог продукта в середине проекта. В этом и заключается концепция фиксированной гибкости. Спонсоры проекта, с разрешения (и через) владельца продукта, чувствуют себя вправе добавлять в бэклог продукта то, что считают нужным, на любом этапе в течение всего проекта.

ХОРОШИЕ И НЕ ОЧЕНЬ ХОРОШИЕ НОВОСТИ

Видите, я же говорил, это просто! Хорошая новость заключается в том, что в каждой ключевой группе вряд ли найдется множество людей, которые не заинтересуются тем, что может предложить Scrum.

Не очень хорошая — как бы ни было легко продать Scrum, его реализация — совсем другая история. Чтобы ваша scrum-команда ревела как готовый к старту Scrum Ferrari, а не урчала как старенький Scrum Pinto, потребуются терпение, непредвзятость, шишки, которые вы уже успели или только успеете набить. И конечно, такие полезные книги, как эта.

ЛАЙФХАК 2. ХРУПКИЙ AGILE

Пожалуй, один из самых неприятных комментариев, которые я слышу от молодых scrum-команд: «Мы используем Scrum: работаем в спринтах, проводим ежедневные стендапы, у нас даже есть бэклог продукта». Может быть, вместо этого сказать честно: «Мы не ведем никакой документации, выпускаем релизы как бог на душу положит, планируем все на лету и не задумываемся об ошибках, потому что просто исправим их в следующей итерации»?

Эти люди делают Scrum ужасную антирекламу, а их проекты неизбежно терпят неудачу. Что еще хуже, после всего этого очень сложно вернуть доверие стейкхолдеров, разочарованных искаженной реализацией Scrum.

ЭТО ФРЕЙМВОРК, А НЕ МЕТОД

Часто можно услышать, что Scrum — это метод. Это неправильно. Scrum — это фреймворк практик, связанных небольшим набором четко определенных правил. Между методом и фреймворком есть существенная разница. Метод подразумевает универсальный формульный подход, в то время как фреймворк предлагает более гибкую платформу, на основе которой могут быть получены разные подходы в зависимости от той или иной рабочей среды.

Чтобы правильно внедрить Scrum, важно следовать четким правилам и работать в рамках практик фреймворка. Только так вы окажетесь на верном пути. Кен Швабер в своем блоге пишет: «Scrum — это как шахматы. Вы либо играете по правилам, либо не играете совсем»15. Расширяя эту аналогию, можно сказать, что частичная реализация Scrum похожа на игру с 20 шахматными фигурами вместо стандартных 32. Есть небольшой шанс, что игра так или иначе пойдет, но это будет не стандартная и выверенная игра в шахматы, это будет другая игра с непредсказуемыми последствиями (см. рис. 1.3).

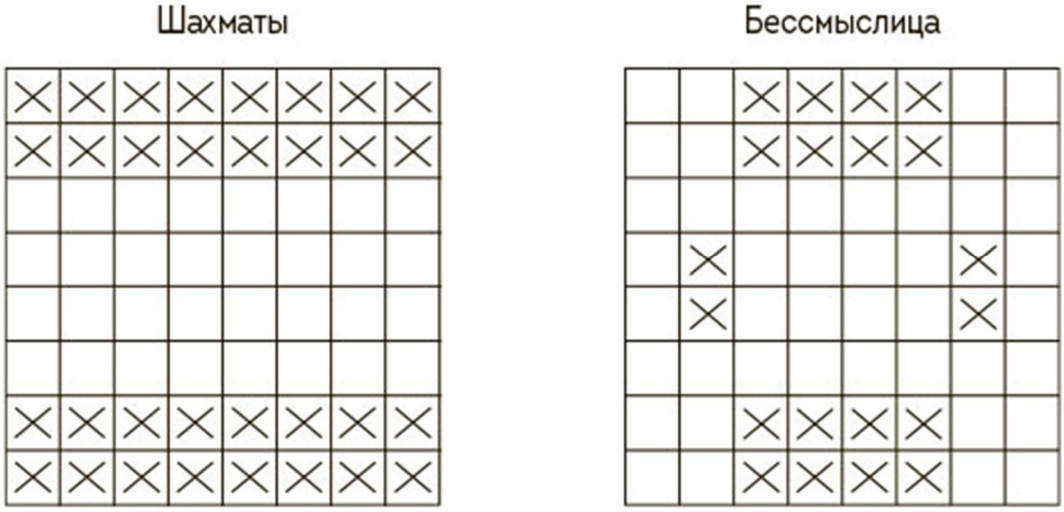

Рис. 1.3. Так же как нельзя менять правила игры в шахматы, не следует менять и правила Scrum

Scrum не содержит лишних правил или практик. Чтобы он работал как задумано, его нужно реализовывать целостно. Частичное применение Scrum равнозначно отказу от него.

КВАЛИФИКАЦИЯ ПРОТИВ КАЧЕСТВА

Сертификация scrum-мастеров, безусловно, полезна, но и здесь присутствует человеческий фактор, снижающий ее значимость. Много лет назад, когда я проходил первый курс обучения на scrum-мастера, в нашей группе был участник, менеджер проекта из банка, который, казалось, видел себя сержантом на курсах молодого бойца. Я тогда подумал:

даже если он несколько лет потратит на обучение, все равно ему не стать scrum-мастером, ведь качества гораздо важнее подтвержденной сертификации (см. лайфхак 4).

ЗЛОУПОТРЕБЛЕНИЕ AGILE-МАНИФЕСТОМ

Те, кто склонен искажать Scrum, обычно цитируют Agile-манифест, чтобы оправдать отсутствие документации и дефицит планирования.

Манифест agile-разработки программного обеспечения

Занимаясь непосредственно разработкой программного обеспечения и помогая в этом другим, мы постоянно открываем для себя более совершенные методы. Благодаря проделанной работе мы осознали следующие ценности:

- **люди и их взаимодействие** важнее процессов и инструментов;
- **работающее программное обеспечение** важнее исчерпывающей документации;
- **сотрудничество с заказчиком** важнее согласования условий контракта;
- **готовность к изменениям** важнее следования плану.

То есть, не отрицая важности того, что справа, мы все-таки больше ценим то, что слева.

Те, кто злоупотребляет Agile-манифестом, либо (а) не прочитали последнюю строку, либо (б) забыли последнюю строку, либо (с) проигнорировали последнюю строку.

И помните: хотя элементы слева ценятся больше, элементы справа также важны и почти всегда пригодятся вам в процессе разработки продукта.

АНТИПАТТЕРНЫ SCRUM

Рассмотрим несколько симптомов того, что Scrum искажен, деформирован и используется неверно. Не путайте их с проблемами «прорезывания зубов», с которыми сталкиваются начинающие (но настоящие) scrum-команды. Например, ежедневный Scrum, который не всегда начинается вовремя, — это неидеально, но при правильной мотивации эту проблему можно решить, и регулярное смещение графика не всегда сигнализирует о том, что команда не принимает эту практику.

Спринты тестирования

Обеспечение качества — неотъемлемая часть процесса разработки. Требование не может считаться выполненным, если оно не удовлетворяет критериям готовности (см. лайфхак 11). Однако иногда это правило нарушается. Обычно это происходит на этапе «функциональных» спринтов, фокусирующихся исключительно на разработке нового кода (чтобы создать впечатление прогресса), дополняемых связкой «догоняющих» спринтов для выявления и исправления багов.

Типичное оправдание такого поведения — команда хочет в первую очередь показать общие функциональные возможности ПО пользователям (чтобы проверить свою работу). В такой ситуации я обычно говорю команде: «Если вы работаете в спринтах, это еще не значит, что вы ушли от водопадной модели». Помните: до тех пор

пока функциональность продукта не будет полностью протестирована и готова к релизу, она не является завершенной (должна считаться невыполненной, то есть бесполезной, и очень рискованной).

Другое проявление этого антипаттерна — программисты и тестировщики работают в разных спринтах. Например, тестировщики отстают на один спринт от программистов16. Такое становится возможным, если практика автоматизированного тестирования все еще недостаточно зрелая и зависимость от ручного регрессионного тестирования велика. Этот ступенчатый сценарий спринта неизбежно приводит к таким же «догоняющим» спринтам, которые обсуждались выше.

Бесконечные нулевые спринты

Нулевой спринт на самом деле не спринт, а искусственный термин, используемый для описания предварительной работы, которую команде необходимо выполнить, прежде чем она будет готова начать первый настоящий спринт (со всеми необходимыми атрибутами).

Эта предварительная работа, как правило, не подразумевает временных рамок и не обладает всеми типичными структурными элементами, которые есть в реальном спринте (бэклог спринта и четко сформулированные требования).

Хотя мне не нравится вводящий в заблуждение термин, концепция предварительного этапа мне понятна и я не имею ничего против. Главная проблема с нулевым спринтом возникает, когда в него входит бесполезная работа, задерживающая начало реальных спринтов. Давайте взглянем на то, что должно и чего не должно быть в нулевом спринте (см. рис. 1.4).

Рис. 1.4. Минимальный список задач для нулевого спринта

Элементы в списке «Не включать» на рис. 1.4 могут показаться туманными, в отличие от конкретных функциональных требований, но это не значит, что их нельзя оценить, спланировать и разбить на задачи и, следовательно, включить в спринт. Ввиду неопределенного характера этой подготовительной работы необходимо окружить ее надлежащими техниками спринта, чтобы обеспечить большую прозрачность и более жесткий контроль.

Изменяющаяся продолжительность спринта

Постоянная продолжительность спринта важна по ряду причин, описанных в лайфхаке 8.

Одно из самых распространенных оправданий, которые я слышал при несоблюдении этого правила: команда хотела взять несколько дополнительных дней и закончить почти готовые требования, чтобы обзор спринта был впечатляющим.

Полагаю, есть только две причины для изменения длины спринта:

1. Когда новая команда экспериментирует на начальном этапе работы.
2. Когда все работы завершаются ранее последнего дня финального спринта (см. рис. 1.5).

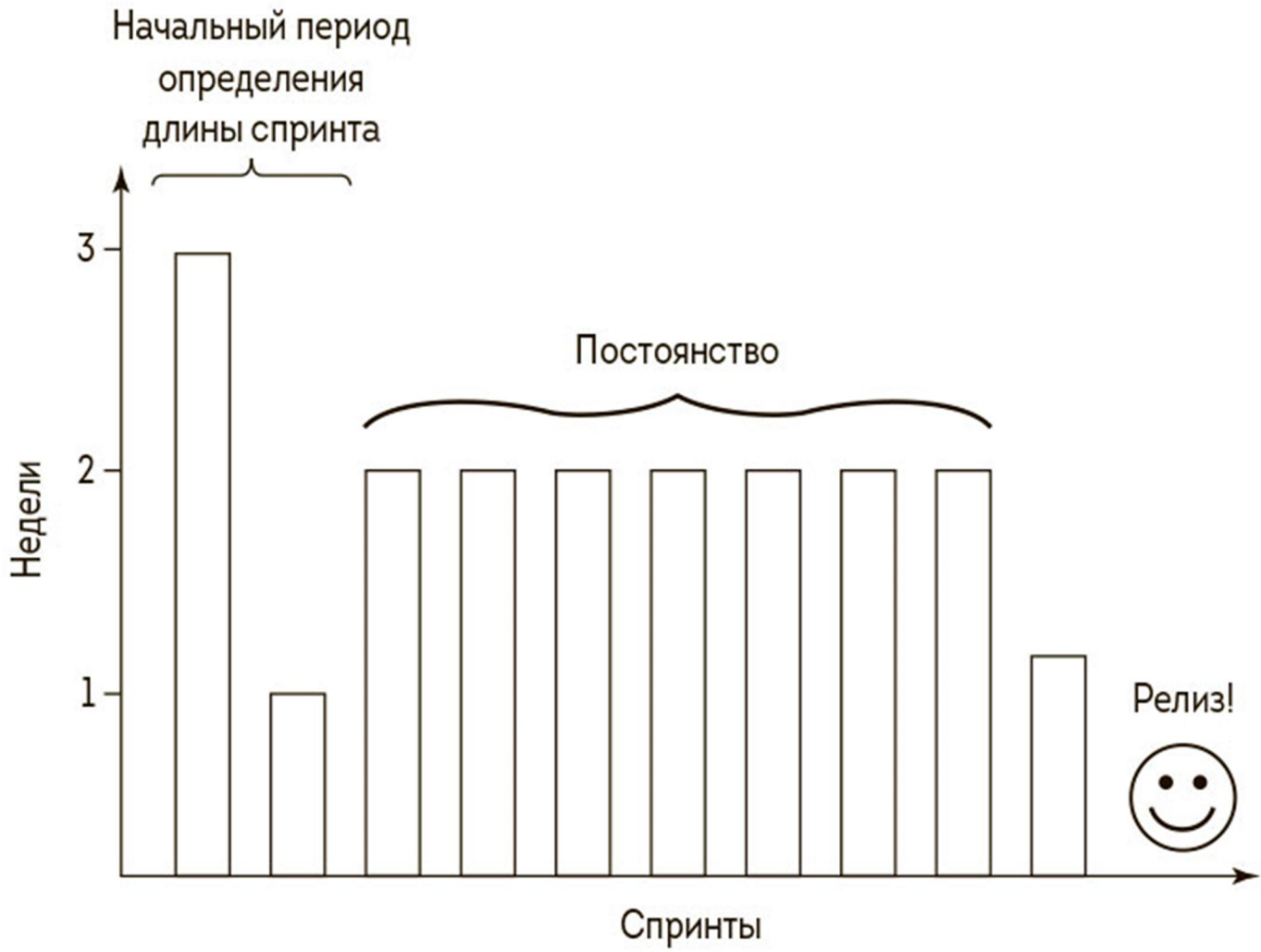

Рис. 1.5. После определения продолжительности спринта поддерживайте ее неизменной

Изолированная оценка

Подобная ситуация характерна для всех процессов вне Scrum, с которыми я сталкивался. Она возникает, когда ведущий разработчик пытается единолично оценить продолжительность всех работ.
Вы можете спросить: а в чем проблема? Разве опытный сотрудник с наивысшей квалификацией не способен дать точную оценку? Тем не менее это действительно одна из проблем. Хотя ведущий разработчик может быть самым опытным, в большинстве случаев сам он не станет выполнять работу. По своим способностям он, несомненно,отличается от участников команды, которым и предстоит выполнить стоящие перед ними задачи. Следовательно, и его оценка будет отличаться от реального положения дел. Работу выполняет не один человек, а вся команда, поэтому она (как коллектив) и должна давать оценку (см. лайфхак 14). Когда это делает один человек, никакой системы сдержек и противовесов для него не существует. А если у него был выходной, он не услышал некоторые важные данные и сделал неправильные выводы? Когда в оценке задействованы все участники

команды, такие промахи гораздо менее вероятны благодаря нескольким слоям и фильтрам при обработке информации.

Доверие к спецификации

Услышав такие комментарии, как «если этого нет в спецификации, это не будет сделано» или «я реализовал именно то, что было в спецификации», можете быть уверены: ваша одежда насквозь промокла от водопада, под которым стоит ваша команда. Это также показывает, что участники команды утратили способность общаться между собой и стали бюрократами. Любая так называемая спецификация должна служить лишь напоминанием или шаблоном. Фактические же требования выполняются коллективно в процессе совместной слаженной работы команды.

Ошибка при выборе scrum-мастера

Если ваш scrum-мастер не поддерживает атрибуты, описанные в лайфхаке 4, и/или не выполняет функции, указанные в лайфхаке 26, то вас, скорее всего, ведет самозванец. Если он единолично принимает продуктовые и технические решения, раздает задачи в режиме микроменеджмента, регулярно заставляет команду работать сверхурочно или ведет себя как тиран, проблемы неизбежны.

СЛУШАЙТЕ РОДИТЕЛЕЙ

Прислушайтесь к совету, который родители наверняка давали вам не раз и не два: если вы собираетесь что-то сделать, делайте это правильно. Мы уже говорили о том, что в Scrum, как и в шахматах, правила менять нельзя. Либо реализуйте Scrum в границах фреймворка, либо не называйте то, что делаете, Scrum.

ЛАЙФХАК 3. ТВОРЧЕСКИЙ КОМФОРТ

Scrum-мастер: Доброе утро, ребята!

Разработчик 1: (*Ворчит.*)

Разработчик 2: (*Едва различимый кивок.*)

Разработчик 3: (*Наушники на голове, никакой реакции.*)

Знакомо?

Давным-давно этот типичный ритуал был для меня одним из самых неприятных моментов начала работы. Обычное утреннее приветствие всегда вызывало у меня раздражение, и это в самом начале рабочего дня. Почему? Да просто потому, что взаимодействия, подобные описанному, могут заставить вас чувствовать, что вы входите в склеп, а не в улей продуктивности.

Некоторые из вас могут решить, что немного грустно говорить «Доброе утро!», особенно холодным, серым утром в понедельник. «Мы ведь инженеры, черт возьми, а не продавцы мыльных пузырей!» Тогда представьте: вы пришли в гости к приятелю, едва слышно что-то буркнули вместо приветствия, плюхнулись на диван и сидите молча. Как чувствовал бы себя ваш друг в такой ситуации? Думаю, он был бы раздражен.

Улыбка и искреннее «Как дела?» помогают почувствовать, что вы среди друзей, способствуют более продуктивному и живому настрою. Такие мелочи, как дружеское приветствие утром, могут серьезно влиять на то, как команда будет работать, а ее участники — взаимодействовать друг с другом.

Советы этого лайфхака подойдут любой команде. Но вероятно, scrum-команды — наиболее сплоченные группы, с которыми вам доведется работать, поэтому очень важно в самом начале дня убедиться, что все ваши коллеги полны энергии и воодушевлены предстоящей работой.

Зачастую мы проводим на работе большую часть жизни и с коллегами общаемся чаще, чем с родителями, супругами и детьми. Тем из вас, кто, прочитав это, тяжело вздохнул, нужно продолжать читать дальше, потому что приход на работу не должен быть похож на вход в забой.

Хорошая новость: существует несколько простых (и недорогих) способов поддерживать удовлетворенность команды.

ЛИЧНАЯ БЛАГОДАРНОСТЬ

После всего сказанного выше о главенствующей роли scrum-команды вы могли подумать, что признания индивидуальных заслуг здесь просто нет. Это не так. Scrum, безусловно, больше ориентирован на командный результат, чем на индивидуальные рекорды и достижения, но это не означает, что люди становятся винтиками в системе. Команды состоят из отдельных личностей, которым необходимы признание и высокая оценка их тяжелой работы.

Я был scrum-мастером в молодой scrum-команде, которая так хорошо проявила себя на очень важном проекте, что мы получили общекорпоративную премию как лучшая команда года. Это признание было невероятно важно для команды в целом, но, когда я подходил к каждому и искренне благодарил за хорошо выполненную работу, признавая его вклад, это было вдвойне приятно всем.

Вот что говорит Тони Шварц, президент и CEO Energy Project, автор блога в Harvard Business Review, о том, почему индивидуальная оценка имеет большое значение:

Искренняя признательность их заслуг окрыляет и вдохновляет людей. На самом базовом уровне это дает ощущение безопасности и позволяет лучше делать свою работу. Это заряжает энергией. Когда нашу работу

оценивают неадекватно — а такое часто случается, — нас охватывает мучительное беспокойство, это истощает силы и отвлекает от создания ценности17.

Разумеется, я рекомендую не отказываться от благодарности всей команде, а вслед за Дейлом Карнеги, автором бестселлера «Как завоевывать друзей и оказывать влияние на людей», «оставлять дружеский след маленьких искр благодарности в ваших ежедневных путешествиях». Так вы поможете людям творить чудеса.

РАБОЧАЯ СРЕДА

Помню, как несколько лет назад мое электронное письмо с описанием необычной обстановки офисов Google в Цюрихе стало вирусным («Работа в Google» 2005). Многие мои друзья (никак не связанные с программным обеспечением) искренне удивлялись расходам на офисную мебель и оборудование, которые, по их словам, больше подходили детской игровой площадке, чем офису солидной компании. «Люди действительно там работают или просто дурачатся весь день?» — возмущался мой друг адвокат.

Наша отрасль как свет в конце тоннеля корпоративного мира мрачных офисов, больше похожих на фабрики индустриальной эпохи. В отличие от моего друга юриста многие технологические компании понимают, что выполнять сложную работу, наслаждаясь рабочей средой, действительно возможно!

Я не говорю, что вам нужно накупить цветастых кресел-мешков, лавовых ламп и поставить горки. Однако стремление превратить рабочую среду в нечто похожее на уютный дом не должно рассматриваться как каприз или причуда. Это осмысленная цель!

Scrum в значительной степени ориентирован на интерактивную среду, которая способствует такой scrum-ценности, как открытость, поэтому рассматривайте приведенный ниже перечень как фундамент благоприятной scrum-среды:

- большая белая доска и свободные стены, где можно разместить доски задач и связанные с ними артефакты;
- много света (хотя мне доводилось работать с парой странноватых разработчиков, у которых была вампирская тяга к темноте);
- открытое пространство для каждой команды с небольшими перегородками, отделяющими зону одной команды от другой;
- достаточно широкие проходы между стульями для комфортного проведения промежуточного контроля (см. лайфхак 12);
- небольшой круглый стол для обсуждений в рабочей зоне команды;
- легкодоступные вместительные переговорные с проекторами и белыми досками для таких scrum-встреч, как планирование спринтов, обзоры спринтов и ретроспективы;

- зоны «Не беспокоить», где участники команды могли бы уединиться и спокойно поразмышлять; там должно быть достаточно места, чтобы ходить, а также должен стоять стол;
- изолированные зоны для личных телефонных разговоров;
- буферная зона, защищающая от самых шумных подразделений компании — отдела продаж и отдела по работе с клиентами, где сотрудники говорят по телефону большую часть времени.

ОТЛИЧНОЕ ОБОРУДОВАНИЕ

Предложение разработчикам внедрить новейшие и лучшие технологии не должно рассматриваться как жест великодушия. Как вы думаете, считают ли плотники острую стамеску роскошью? Нет, это для них необходимый рабочий инструмент.

Самое интересное: разработчики, которым предлагают новейшее и максимально производительное оборудование, видят в этом огромное преимущество для себя. Как отмечает обозреватель программного обеспечения Джоэл Спольски18, программистов легко подкупить, предоставив им самое крутое и самое современное «железо». Это гораздо более дешевый способ заставить их работать на вас, чем конкурентоспособная зарплата!

Я уверен, Джоэл не призывает платить меньше и не агитирует за экономию на окладах и бонусах. Дело в другом: просто предоставляя хорошие инструменты вместо старого подержанного оборудования (ради экономии пары долларов), вы не только делаете разработчиков счастливыми, но и улучшаете общую производительность!

ИДЕНТИЧНОСТЬ

Люди по своей природе общинные, племенные создания, нам нравится чувствовать себя частью особой группы. Scrum — идеальный инструмент для подобного социального контекста, ведь он поощряет размещать команду отдельно от других сотрудников. Как только этот первый шаг будет сделан, наступает время строить действительно прочные связи. Я призываю (но не заставляю) scrum-команды придумать себе название, выбрать свой цвет, как это делают спортивные команды, и украсить свою территорию соответствующими плакатами, логотипами и баннерами. Как отмечают Том Демарко и Тимоти Листер в своей эпохальной книге Peopleware: «Участники хорошей команды испытывают ощущение элитарности. Они чувствуют, что составляют нечто уникальное, что они выше всякой заурядности»19.

В одной компании, которую я познакомил со Scrum, образовались команды «Вулкан», «Громовые коты» и даже «Потрясающая команда», которые вели дружескую борьбу, постоянно подтрунивая друг над другом, например, за то, что сломали сборку или проводили слишком

долгие ежедневные стендапы. Я искренне верю, что эта невинная конкуренция повысила не только производительность, но и чувство гордости за качество проделанной работы.

Хотя индивидуальность команды — жизненно важный аспект, необходимо, чтобы она идентифицировала себя с продуктом, который разрабатывает, и чувствовала себя вовлеченной в него. По словам Майка Кона, один из лучших способов добавить энергии — разжечь страсть. Чем больше увлеченных людей, тем выше вероятность их ежедневного максимального погружения в проекты. В данном случае ключевую роль играет владелец продукта. Он должен иметь четкое представление о том, что хочет получить в итоге, убедительно изложить свои ожидания и передать свое видение команде, чтобы та работала над проектом с энтузиазмом.

Я нашел хороший способ пробуждать такую страсть и чувство сопричастности. Следует приглашать разработчиков на первые встречи, посвященные пользовательским историям. Это поможет scrum-команде не только почувствовать себя вовлеченной в концепцию продукта, но и с самого начала понимать, какого результата от нее будут ждать и почему.

СИЯЮЩИЕ, СЧАСТЛИВЫЕ ЛЮДИ

Превращая рабочую среду в домашнюю или, еще лучше, в парк захватывающих аттракционов, включая местных джокеров и клоунов (в моих командах их было много), мы возвращаемся в наш самый беззаботный период жизни — детство, когда мы были максимально креативны и безмерно счастливы.

Думать, что такие широкие жесты, как выплата бонусов, или раздача красиво звучащих должностей, напечатанных на визитках с восхитительными водяными знаками, или такие инициативы, как знаменитые 20% времени Google[20], позволят сотрудникам оставаться постоянно мотивированными, — это иллюзия. Я не считаю, что люди должны постоянно испытывать чувство «мастерства, целеустремленности и автономии». В книге «Драйв. Что на самом деле нас мотивирует» Дэниел Пинк отмечает, что теплое приветствие утром, искренняя похвала за хорошо проделанную работу и ощущение, что вы часть уникальной команды, — часто этого достаточно, чтобы поддерживать улыбки на лицах ваших коллег[21].

ЗАКЛЮЧЕНИЕ

В трех лайфхаках этой главы мы сосредоточились на выборе тактических приемов, инструментов и советов, которые помогут вам и вашей компании освоить Scrum. Давайте вспомним, что мы обсуждали.

Лайфхак 1. Scrum на поле

- краткая презентация Scrum;
- концепция фиксированной гибкости, используемая для минимизации переключений контекста и одновременно предлагающая гибкость по отношению к скоупу работ;
- ключевые преимущества для scrum-команды и спонсоров проекта.

Лайфхак 2. Хрупкий Agile

- как отличить фреймворк от метода;
- почему важнее фокусироваться на личных, а не на профессиональных качествах;
- на какие антипаттерны Scrum стоит обратить особое внимание.

Лайфхак 3. Творческий комфорт

- как развить и укрепить командный дух, фокусируясь на благодарностях всей команде и каждому ее участнику;
- как улучшить рабочую среду, чтобы обеспечить продуктивное пространство взаимодействия;
- как помочь вашей команде прочувствовать свою идентичность и проникнуться общей целью.

Глава 2

ОТНОШЕНИЕ И СПОСОБНОСТИ

Если посмотреть на типичный список вакансий, можно увидеть, что большинство работодателей делают акцент на описании функциональных задач и обязанностей. Однако для успеха Scrum необходимо использовать совершенно другой подход и больше углубляться во внутренние фундаментальные установки и компетенции, необходимые участнику scrum-команды. Следующие три лайфхака помогут погрузиться глубже в роли участников scrum-команды и осознать, что им необходимо, чтобы отлично справляться со своими обязанностями.

Лайфхак 4 «Искусный scrum-мастер» опишет семь ключевых навыков, которыми должен обладать великий scrum-мастер.

Лайфхак 5 «Рок-звезды или студийные музыканты?» посвящен разбору фундаментальных компетенций, которые необходимы участникам scrum-команды.

Наконец, лайфхак 6 «Выстраиваем команду» предложит инструкцию, как собрать эффективно работающую scrum-команду.

ЛАЙФХАК 4. ИСКУСНЫЙ SCRUM-МАСТЕР

Я не верю, что звание scrum-мастера может быть присвоено кому-то только потому, что он или она знает правила и практики Scrum вдоль и поперек. Этого звания достойны только те, кто понимает подлинную суть этой роли и руководствуется ею в жизни. Scrum-мастер должен осознавать, что означает быть лидером-слугой. Роберт Гринлиф, основатель современного движения «Лидер-слуга», так описывает эту, казалось бы, противоречивую роль:

Все начинается с искреннего чувства, что ты хочешь служить, в первую очередь служить. И это осознание заставляет стремиться к лидерству. Такой человек резко отличается от того, кто хочет в первую очередь быть лидером из-за жажды власти или материальных благ22.

Итак, что значит быть лидером-слугой в контексте Scrum? Давайте посмотрим на некоторые фундаментальные компетенции и навыки, которыми должен обладать настоящий scrum-мастер.

ЛИДЕРСТВО БЕЗ ПОЛНОМОЧИЙ

Умение руководить без получения управленческих полномочий — главный вызов для нового scrum-мастера. Присоединиться к группе — сложно. Руководить группой — еще сложнее. Вести за собой, не обладая явными управленческими полномочиями, — сложнее всего (см. рис. 2.1).

Рис. 2.1. Лидерство без полномочий требует настоящего лидера-слуги

Руководить, используя безграничную власть и чрезвычайные полномочия, на первый взгляд кажется простым делом. Однако любой свергнутый диктатор согласится с тем, что такой подход не работает в долгосрочной перспективе. На протяжении какого-то периода диктатор может выжимать результаты из подчиненных, не испытывающих

никакого уважения к руководителю. Но лишь вопрос времени, как скоро так называемое лидерство рухнет и воцарится хаос — вне зависимости от того, сколько сил было потрачено ради сохранения власти. Подлинному лидеру не требуются ни власть, ни сверхполномочия, ни, разумеется, сила. Люди хотят следовать за таким человеком и вдохновляются более деликатным стилем лидерства.

Я верю, что способность быть лидером, не имея власти, врожденная. Вероятно, ее можно развить, но это очень сложно. Для тех, у кого ее нет, но кому очень хочется ее развить, приведу небольшой список того, с чего стоит начать:

- отбросить свое эго;
- искренне заботиться как о команде, так и о продукте;
- действовать честно и вести себя одинаково по отношению ко всем участникам команды;
- излучать уверенность и скромность одновременно;
- быть максимально доступным в любое время (перерыв на душ может быть исключением).

ВНЕДРЯЙТЕ ИЗМЕНЕНИЯ БЕЗ СТРАХА

Как говорил Вудро Вильсон, 28-й президент США: «Хотите нажить себе врагов — попробуйте что-то изменить». Изменения пугают большинство людей, потому что вынуждают выйти из зоны комфорта в новый, неизвестный мир — мир, в котором сложившийся статус и профессионализм подвергнутся проверке на прочность.

Проблема для молодого и полного энтузиазма scrum-мастера заключается в том, что использование Scrum полностью изменит мир команды, в которую он пришел.

Если не внедрять изменения обстоятельно и благоразумно, то любые из них, даже конструктивные, могут быть восприняты участниками команды негативно.

Присоединяясь к новой scrum-команде, вы не должны поспешно все менять. Будьте терпеливы; наблюдайте за средой, текущими процессами, людьми, командой, технологиями и тем, что происходит в организации в целом. Будьте «мухой на каждой стене» и поговорите с максимальным количеством людей. Даже если у вас приказ сверху немедленно реорганизовать работу команды и «скраммифицировать» все вокруг, сначала оцените ее готовность к этому. У вас не будет второго шанса произвести первое впечатление: если вы начнете переход на Scrum раньше времени, внедрить изменения станет гораздо тяжелее.

Сразу найдите союзников и укрепляйте отношения с ними. Люди из вашего окружения, готовые стать ранними последователями, — те, кто рад позитивным изменениям, — окажут неоценимую помощь при внедрении новых практик.

Когда будете готовы, не медлите. Сначала реализуйте одну или две инициативы (например, начните проводить ежедневный scrum и вести единый бэклог продукта). Добейтесь нескольких небольших, но убедительных побед, расскажите о полученных выгодах и отталкивайтесь уже от этого. После того как завоюете доверие, окружающие будут гораздо лучше воспринимать ваши стремительно разворачивающиеся в команде инициативы.

БУДЬТЕ ДИПЛОМАТОМ, НЕ БУДУЧИ ДЕМАГОГОМ

Scrum-мастер — это ступица, соединяющая спицы в колесе — представителей разных отделов, которых нужно объединить для создания эффективно работающей scrum-команды.

Чаще всего вы будете сталкиваться с глубоко укоренившейся привычкой каждого отдела (например, IT-отдела и отдела маркетинга) жить в своей отдельно стоящей от всех остальных подразделений башне (см. рис. 2.2). И что еще хуже, не просто в башне, а в целой крепости с отлично выстроенной системой обороны, призванной держать чужаков из других подразделений как можно дальше. Нужны навыки искусного дипломата, чтобы одолеть эту привычку делить всех на своих и чужих. Нужно уметь донести выгоды от работы команды и научить уважению ко всем ролям, которые необходимы для выполнения работы и создания ценности для пользователя. Scrum-мастер не должен принимать чью-либо сторону или втягиваться в интриги внутри компании. Заботы scrum-мастера — эффективность и поддержание здоровой рабочей атмосферы в команде. Но ни первое, ни второе несовместимо с демагогией.

Рис. 2.2. Настоящий scrum-мастер построит мост через пропасть между отделами

Как пишет Кеннет Рубин: «Scrum-мастер должен быть открыт и честен в общении. При работе с участниками команды нет места для тайн и скрытой информации; то, что вы видите и слышите от scrum-мастера, должно быть тем, что вы получаете»23.

ВЕДИТЕ СЕБЯ САМООТВЕРЖЕННО, НО НЕ ОБЕСЦЕНИВАЙТЕ СВОЮ РОЛЬ

Я помню, как смотрел «Тур де Франс» (ежегодную изнурительную велогонку во Франции) и был поражен быстрым коротким финишем одного из велосипедистов на особенно сложном и изматывающем участке. После того как все участники гонки набрали скорость, возник настоящий хаос из велосипедистов, толпой летящих к линии финиша. Всего за несколько миль до финиша два гонщика из одной команды оторвались от пелотона. Один ехал в более комфортной позиции, пристроившись за задним колесом второго — ведущего, который проторял дорогу. Проделав всю тяжелую работу, ведущий велосипедист притормозил и встал обратно в пелотон. Он потратил почти всю энергию, но вытащил товарища по команде подальше от основной группы велогонщиков и открыл ему путь к победе. Эта самоотверженная роль в велогонках выполняется ведущим. Обязанность ведущего — использовать всю свою выносливость и тактическое мышление для защиты товарища по команде ради его победы, не претендуя на личную славу.

Scrum-мастер должен выступать как этот ведущий велосипедист; признание успехов команды должно быть для него выше признания его собственных заслуг. Но такое самоотверженное поведение не означает, что роль ведущего должна быть недооценена. Да, ведущий не будет стоять высоко на пьедестале, но роль, которую он играет, должна признаваться жизненно необходимой для успеха команды.

ЗАЩИЩАЙТЕ, НО БЕЗ ФАНАТИЗМА

При описании роли scrum-мастера обычно используют метафору овчарки, которая ведет стадо через опасную местность и защищает от волков. Мне нравится это сравнение, но я предостерегаю от того, чтобы воспринимать его буквально. Хорошему scrum-мастеру не следует излишне опекать свою команду — так же как маме не нужно становиться «матерью-наседкой», которая постоянно квохчет вокруг своих детей (конечно же, с самыми лучшими намерениями) и не оставляет ребенку ни единого шанса на самостоятельность.

Scrum-мастер должен знать, когда ему нужно бросать все и спасать команду, а когда лучше позволить команде решать проблемы самостоятельно, чтобы дать ее участникам возможность расти как в личностном плане, так и профессионально.

ПОДДЕРЖИВАЙТЕ ТЕХНИЧЕСКИЕ ЗНАНИЯ, НЕ БУДУЧИ ЭКСПЕРТОМ

Технические специалисты могут стать великими scrum-мастерами, но я часто сталкиваюсь с двумя большими проблемами, когда scrum-мастером является либо технический специалист, либо эксперт в какой-то узкой области:

- Когда участник команды столкнулся со сложностями при решении задачи, непреодолимое желание помочь ему или ей возникает слишком рано, не дав возможности разобраться самостоятельно.
- Желание поучаствовать в обсуждении технических или функциональных деталей на планировании спринта будет слишком велико. А принимая активное участие в подобном обсуждении, scrum-мастер отвлекается от своих основных обязанностей — обязанностей фасилитатора.

СПОКОЙНО ОТНОСИТЕСЬ К ТОМУ, ЧТО РАБОТА НИКОГДА НЕ БУДЕТ ЗАКОНЧЕНА

Scrum-мастер может дожить до того счастливого дня, когда, посмотрев на команду, он поддастся искушению и скажет: «Мы сделали это! Я больше ничего не могу улучшить!» Однако независимо от того, как обстоят дела, помните: нет предела совершенству и улучшения возможны всегда. Более того, со временем команда будет изменяться — за счет выгорания кого-то из участников, продвижения по службе, а в некоторых (печальных) обстоятельствах — увольнения. После таких изменений снова предстоит много работы и дальнейших улучшений.

ЛИДЕРСТВО СЛЕДУЮЩЕГО ПОКОЛЕНИЯ

Настоящие scrum-мастера — часть нового поколения «просвещенных профессионалов». Роль scrum-мастера глубже и сложнее, она не сводится к перечню обязанностей из должностной инструкции. Чтобы осознать ее суть, придется смотреть в самый корень.

Вот почему я настаиваю, что при поиске новых scrum-мастеров нужно видеть картину целиком: scrum-мастером может оказаться человек с любым бэкграундом — главное, чтобы он успешно демонстрировал способности, описанные в предыдущих лайфхаках. Не каждый может стать scrum-мастером, но каждый может им оказаться.

ЛАЙФХАК 5. РОК-ЗВЕЗДЫ ИЛИ СТУДИЙНЫЕ МУЗЫКАНТЫ?

Пока никто не смотрит, я бы хотел тайком вставить одну дополнительную строку в Agile-манифест:

Отношение важнее профессионализма

Несмотря на то что профессионализм — это ценно, еще больше мы ценим отношение к работе и коллегам. Не поймите меня неправильно, профессиональные компетенции, безусловно, важны, но, если бы мне пришлось выбирать между опытным разработчиком с

суперответственным отношением к работе и гениальным, но вечно угрюмым и неприветливым разработчиком, я выбрал бы первого.

РОК-ЗВЕЗДЫ

Среди IT-специалистов популярен тренд набирать в команду ярких «рок-звезд». У меня всегда была проблема с этой метафорой, ведь она запутывает всех. Давайте на минуту задумаемся, какими качествами обладают настоящие рок-звезды. Уверен, вы согласитесь, они представляются нам харизматичными, творческими личностями с яркой индивидуальностью. Это ведь хорошие черты, верно? Но давайте взглянем на другую сторону медали — взрывной характер, постоянное желание привлечь к себе внимание, высокомерие и подход «есть мое мнение и неправильное». Разве это похоже на хорошего командного игрока? Думаю, нет.

СТУДИЙНЫЕ МУЗЫКАНТЫ

Теперь давайте посмотрим на студийных музыкантов. Они легко обходятся без света софитов, вместо стремления к славе поддерживают вокалиста группы в создании отличного альбома. Ветеран музыкальной индустрии Бобби Овсински пишет в справочнике студийного музыканта:

Все считают, что студийные музыканты — это творческие универсальные музыканты с невероятно широким набором разных навыков… На самом деле секрет в том, что если вы тесно работаете с другими музыкантами, звукорежиссерами, продюсерами, исполнителями, музыкальными лейблами и агентами (и кто знает, с кем еще) и с вами работать легко, значит, шансы, что вас позовут снова, весьма велики.

Работа в студии имеет свои особенности, и она отличается от игры вживую. Существует протокол студийного музыканта, и вы должны будете его соблюдать. Достаточно сказать, что, если вам нравится быть в центре внимания, студийная работа не для вас[24].

Мой вывод: я бы предпочел команду студийных музыкантов команде рок-звезд.

ЦЕННОСТИ SCRUM

Как набрать команду студийных музыкантов, чтобы быть уверенным, что ваш Scrum не рухнет под грузом огромного эго участников команды и постоянных конфликтов? Стоит начать с основных ценностей Scrum. Ценности Scrum должны разделяться всеми участниками команды и лежать в основе их профессиональных качеств. Эти ценности показаны на рисунке 2.3.

Рис. 2.3. Основные ценности Scrum25 должны быть приняты всеми участниками команды

В дополнение к ценностям Scrum я всегда ищу в участниках scrum-команды следующие качества: энергия, эмпатия, любознательность и дружелюбие. Давайте расскажу, что я имею в виду, более подробно.

Энергия

Я работал с несколькими умными, профессиональными разработчиками. Это были дружелюбные и добродушные люди. Похоже на портрет идеального кандидата в scrum-команду, верно? Однако те же самые разработчики обладали особыми способностями, сродни тем, что были у высасывающих души дементоров из «Гарри Поттера» (подробнее о дементорах можно прочесть в энциклопедии Гарри Поттера). Как настоящие энергетические вампиры, они могли иссушить всю положительную энергию в комнате. Особенно это проявлялось во время ежедневных scrum (стендапов), основная цель которых — дать команде энергетический заряд на весь день. Если кто-то из участников команды пополняет свой запас энергии, высасывая ее из остальной команды, посмотрите, что беспокоит этого участника и чем вы можете ему помочь.

Эмпатия

Работа в тесном коллективе требует терпения и понимания. Каждый участник команды полагается на всех остальных при достижении командных целей. Однако реальность такова, что у всех у нас случаются непродуктивные дни. Проколотая шина, опоздавшая няня, тяжелые личные обстоятельства или просто плохое самочувствие — огромное количество вещей может помешать нам достичь прогресса. Когда такие ситуации возникают (а они неизбежны), другие участники команды должны помочь и взять на себя любую дополнительную нагрузку — как, например, солдат вытаскивает раненого товарища с поля боя.

Любознательность

Команды разработки кросс-функциональны. Из лайфхака 6 вы узнаете, что в идеале они состоят из людей, обладающих «Т-образными» компетенциями. Иными словами, участники команды должны уметь грамотно действовать за пределами своих компетенций, чтобы преодолеть бутылочное горлышко узкопрофильности. Это требует

желания и готовности расширять свои навыки и умения, используя любую возможность узнать больше о задачах и функционале, в которых они не являются экспертами.

Дружелюбие

Помню, как работал с асоциальным, но умным и профессиональным разработчиком. После его особенно критического выпада против нового владельца продукта я решил поговорить с ним. Наш разговор прошел примерно так:

Я: Независимо от того, что ты думал о его идеях, есть способы и средства донести свое мнение без атомной бомбардировки словами.

Он: Мне платят не за то, чтобы я заводил здесь друзей, а за работу.

Я: И да и нет, приятель. Ты прав, ты здесь, чтобы выполнять свою работу, но тебе платят еще и за то, чтобы ты работал в команде в атмосфере сотрудничества. Чем более дружелюбным ты будешь, тем более эффективным станешь.

Он: (*Тишина.*)

Подбирая человека в команду, я ищу не просто вежливого человека — я стараюсь идентифицировать того, кто действительно дружелюбен. Гораздо легче спешить на помощь другу, чем незнакомцу (или, что еще хуже, кому-то, кого вы не любите). И само собой разумеется, работать с друзьями намного веселее (это идет только на пользу). Как пишет Юрген Аппело, автор книги «Management 3.0»: «Это не значит, что вы должны со всеми стать близкими друзьями. Это невозможно. Но узнать немного больше о жизни коллег, об их семьях, доме и увлечениях (а им знать больше о вас) было бы отличным началом»26.

Уважение

Уважение является одной из основных ценностей Scrum. В отличие от других ценностей Scrum, именно эту ценность порой воспринимают неоднозначно, поэтому я считаю важным более подробно объяснить, что я подразумеваю под уважением.

Давайте посмотрим правде в глаза: люди (даже очень умные) иногда не слишком удачно формулируют свои мысли. Возможно, они неправильно истолковали какую-то важную исходную информацию, а может быть, просто переволновались и выдали то, что было на уме, не отфильтровав сначала. К сожалению, я работал в нескольких компаниях, где мозговой штурм больше походил на схватку дзюдоистов, которые только и ждут ошибки друг друга, чтобы бросить «противника» через комнату, парализовав его при помощи гиперкритики.

Враждебность — последнее, что вы хотите получить, когда ваша команда креативит. Надо создать чувство безопасности, когда каждый

знает: товарищи по команде будут уважительны, даже если они не в восторге от ваших идей или мнений. Как пишет Дейл Карнеги, проявляйте уважение к мнению другого человека. Никогда не говорите, что он не прав. Когда люди слышат «вы не правы» слишком много раз, их идеи (включая хорошие), скорее всего, вскоре иссякнут (см. рис. 2.4). Есть возможности не соглашаться без демонстрации неуважения.

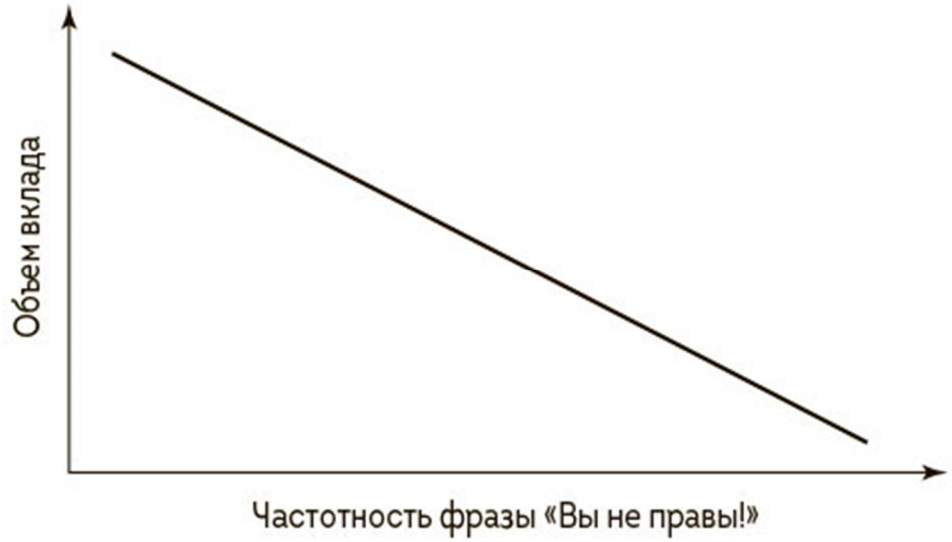

Рис. 2.4. Усилия, которые вы готовы приложить, уменьшаются тем больше, чем чаще вам говорят: «Вы не правы!»

ВРЕМЯ ДЕЛАТЬ МУЗЫКУ

Я считаю, что успех Scrum основан на командах — людях с одинаковым позитивным настроем, готовых к коллаборации. Группа ярких, но эгоистичных людей никогда не будет работать так же хорошо, как группа сплоченных товарищей по команде, готовых к сотрудничеству.

Помните, что Scrum — это про команды, а не индивидуальности. Это не означает, что люди здесь не признаются уникальными личностями. Это значит, что их личные цели должны уступить место целям команды. Бобби Овсински пишет:

Каждый здесь затем, чтобы сыграть свою роль настолько хорошо, насколько это возможно. Когда лампочка звукозаписи не горит, люди здесь настолько же разнообразны и разобщены, как и везде, но, когда приходит время создавать музыку, каждый на 100% сфокусирован только на музыке.

Однозначно дайте мне студийных музыкантов вместо рок-звезд, большое спасибо![27]

ЛАЙФХАК 6. ВЫСТРАИВАЕМ КОМАНДУ

Спортивные тренеры в течение многих месяцев обдумывают, кого включить в команду. У вас не всегда будет богатый выбор кандидатов,

однако так же, как при формировании спортивной команды, вам потребуется уделить много внимания подбору людей в scrum-команду. Это особенно актуально, если вы запускаете пилотный проект, где будущее Scrum для вашей организации находится в руках этой команды.

Вы должны учитывать множество факторов, подбирая scrum-команду. Это отношение к работе, совместимость друг с другом, набор профессиональных компетенций и личных качеств, размер команды, доля узкопрофильных специалистов в команде, совместно используемые ресурсы и оборудование рабочего пространства — и это только часть. Самое важное — набрать такую команду, которой будет легко проститься с предыдущей иерархической моделью управления и таким привычным взглядом на мир, как разделение на своих и чужих.

КАЖДЫЙ БУДЕТ РАЗРАБОТЧИКОМ!

В успешной самоорганизующейся scrum-команде нет места разногласиям, клановости и разделению на функциональные (например, программисты и тестировщики) и иерархические (техлиды и все остальные) подкоманды.

Scrum решает эту проблему очень просто: все участники команды разработки рассматриваются на равных и имеют одно общее название — «разработчик». Являетесь ли вы программистом, тестировщиком, UX-дизайнером или техническим писателем, в Scrum вы разработчик. С философской точки зрения мне очень нравится этот подход — он устанавливает единые правила игры и отражает тот факт, что в разработке программного обеспечения должны быть задействованы все (это больше не уникальная вотчина программиста, ранее монополизировавшего звание разработчика).

На практике внедрение этого подхода может быть сопряжено с трудностями, поэтому мне нравится объяснять его немного по-другому. Я привожу команде аналогию с медицинским специалистом: независимо от специальности все они врачи. Точно так же, как доктор может специализироваться на неврологии или педиатрии, разработчик может заниматься программированием или тестированием. Для простоты и ясности я, как правило, использую более конкретные обозначения (программист, тестировщик и т. д.) при обращении к ним. Однако, безусловно, выступаю за более широкое понимание профессии разработчика.

РАЗМЕР SCRUM-КОМАНДЫ

Я сделал этот раздел коротким, потому что в сообществе тех, кто использует Scrum, нет никаких разногласий по этому пункту. Как утверждает Майк Кон в своей последней книге «Scrum. Гибкая разработка ПО»: «Общепринятый совет: идеальный размер scrum-

команды — от пяти до девяти человек»[28]. Добавлю, что отдаю предпочтение нижнему порогу — от пяти до семи.

СООТНОШЕНИЕ ГРУПП РАЗРАБОТЧИКОВ

Когда дело доходит до состава команды разработки, не существует единого правила для всех, потому что каждый проект и команда разные. Однако, если вы не знаете, с чего начать, предлагаю соотношение, следуя которому я работал успешно (хотя я настоятельно рекомендую вам проверить его эффективность на практике и адаптировать под свою команду).

3 программиста / 1 тестировщик / 0,5 узкопрофильного специалиста

На заметку:

- в одной scrum-команде может работать несколько узкопрофильных специалистов. Я имею в виду тех, кто сосредоточен исключительно на узких областях, — администраторы баз данных, UX-дизайнеры и отраслевые эксперты;
- 0,5 совсем не означает, что я люблю работать с карликами, просто эти разработчики будут делить свое время между двумя проектами. Мы обсудим эту достаточно спорную идею позднее;
- предложенное соотношение подразумевает высокий уровень автоматизации тестирования, который позволит тестировщику сфокусироваться на задачах, более подробно описанных в лайфхаке 18.

В этом описании идеального соотношения участников scrum-команды я говорю об узкопрофильных специалистах. Вы можете включать таких специалистов в ваши scrum-команды, но при этом важно помнить, что Scrum нацелен на завершение начатой работы как можно раньше. Чтобы использовать такой тип потока задач, необходимо придерживаться концепции «роения». То есть вместо того, чтобы несколько разработчиков одновременно трудились над разными элементами бэклога продукта, предпочтительнее ограничить количество задач и нацелить максимум разработчиков на завершение меньшего количества элементов бэклога. Ограниченная пропускная способность узкопрофильных специалистов создает бутылочное горлышко, и тогда у команды возникает проблема. Чтобы брать лучшее от обоих подходов — работу с универсальными специалистами-разработчиками и работу с узкопрофильными специалистами, — я призываю поощрять разработчиков развивать Т-образные навыки (см. рис. 2.5). Как объясняет Кеннет Рубин[29], разработчик с Т-образными навыками обладает:

1) глубокими знаниями в своей предметной области (например, для UX-дизайна) — вертикальная палочка буквы Т;

2) широкими, но необязательно глубокими навыками в других областях (таких, как тестирование и составление технической документации) — горизонтальная палочка буквы Т.

Рис. 2.5. Поощряйте разработчиков развиваться в своей области и осваивать дополнительные компетенции в смежных областях

Участники команды с развитыми Т-образными навыками гораздо лучше создают «роение».

Майк Кон рекомендует подход, который я использую, чтобы подталкивать команды в правильном направлении:

На следующем планировании спринта договоритесь, что один из участников команды на протяжении всего спринта не будет работать по своей специальности. Он может помогать советами тем, кто выполняет эту работу, но не должен делать ее сам30.

Через быстро обучающихся участников команды знания будут быстро распространяться, и, хотя может показаться, что первая пара спринтов идет медленно, вы начнете видеть отдачу от этих вложений очень скоро.

ЧАСТИЧНАЯ ЗАНЯТОСТЬ

В предыдущем разделе я упомянул о выделении узкопрофильными специалистами 50% своего времени для работы в команде. Это «частичное назначение» не слишком популярно в сообществе Scrum, и по уважительной причине. Джеймс Шор и Шейн Уорден, авторы *The Art Of Agile Development*, сформулировали это так:

Работники, присоединяющиеся к команде только на часть своего рабочего времени, не связаны со своими командами, их часто нет рядом, чтобы участвовать в обсуждениях и отвечать на вопросы, и они должны

переключаться между задачами, что вызывает значительные, но неочевидные на первый взгляд проблемы.

Я полностью согласен с тем, что в идеале все участники команды должны посвящать 100% времени своей команде. При этом я часто нахожу ненужным (с точки зрения требований) и нереалистичным (с точки зрения бюджета) держать узкопрофильных специалистов, таких как администраторы баз данных, посвящающих полный рабочий день одной команде разработчиков. Это не значит, что я не согласен с Шором и Уорденом, и это не значит, что не бывает случаев, когда вам в штате нужны узкопрофильные специалисты. Это значит, что нам часто приходится использовать доступные возможности по максимуму, но во многих ситуациях мы не можем позволить себе роскошь включить узкопрофильных специалистов в команду на полный день.

В качестве утешительного приза для пуристов, читающих эту книгу, отмечу: я не против распределения разработчиков по двум проектам, я категорически против их распыления на три проекта или более — подобный уровень переключения контекста контрпродуктивен.

Еще один вариант, который вы могли бы рассмотреть, особенно если узкопрофильному специалисту приходится работать над более чем двумя проектами одновременно: воспринимайте их как партнеров и тренеров для остальной команды. Вместо того чтобы включать их в состав scrum-команды, вы распределяете их между несколькими командами, и они помогают в конкретных задачах, одновременно обучая других разработчиков.

МОЖЕТ ЛИ SCRUM-МАСТЕР РАБОТАТЬ С НЕСКОЛЬКИМИ КОМАНДАМИ?

Этот вопрос постоянно обсуждают на форумах, посвященных Scrum. Прежде чем изложить свою точку зрения, поделюсь некоторыми статистическими данными, представленными на глобальной scrum-конференции (первой международной конференции по Scrum) scrum-тренером Полом Годдардом:

- 75% scrum-мастеров менее половины своего времени действуют как scrum-мастер в своей нынешней команде;
- 45% scrum-мастеров поддерживают две или более зрелые scrum-команды;
- 88% scrum-мастеров берут на себя дополнительные обязанности за пределами их ответственности как scrum-мастера.

Многие в scrum-сообществе будут доказывать, что роль scrum-мастера достаточно значима, чтобы обеспечить тождество «одна команда = один scrum-мастер». Я поддерживаю этот тезис, а в лайфхаке 26 привожу множество убедительных аргументов, почему scrum-мастер должен все

свое время посвящать команде. И все-таки вопрос остается открытым. Я однажды работал scrum-мастером в трех командах одновременно. Это, возможно, не идеально, но позволяет действовать эффективно, если предположить, что эти команды находятся на пути к зрелости, то есть к высокой самоорганизации. Рассмотрим подобное жонглирование командами чуть позже.

Молодые scrum-команды требуют одного scrum-мастера на полный рабочий день — это очевидно. Только возникшим командам необходим высокий уровень обучения, наставничества, сопровождения и больше всего инспекции и адаптации. Брать на себя больше одной молодой scrum-команды нереалистично.

Взрослеющие команды, которые успешно используют Scrum, не имеют системных проблем (таких как интриги и тому подобное) и идут по пути непрерывных улучшений, могут потенциально работать со scrum-мастером, ведущим две разные scrum-команды.

Наконец, «элитные», зрелые scrum-команды, которые умеют самоорганизоваться и совершенствование которых связано с точечной настройкой, могут, пожалуй, обойтись и scrum-мастером, работающим в трех похожих scrum-командах (см. рис. 2.6).

Рис. 2.6. Сколько команд может вести один scrum-мастер в зависимости от уровня развития команд

ОТНОШЕНИЕ ВАЖНЕЕ ПРОФЕССИОНАЛИЗМА

Отношение настолько важно, что я посвятил ему целый лайфхак, поэтому рекомендую вам не изобретать велосипед, а внимательно прочитать лайфхак 5.

НЕ БОЙТЕСЬ СБОРНОЙ СОЛЯНКИ (НО БУДЬТЕ ОСТОРОЖНЫ)

Имея возможность активно путешествовать и жить на нескольких континентах, я могу понять и оценить преимущества неоднородности. Это особенно очевидно в командах разработки. Поскольку считается, что разработка программного обеспечения является отраслью с международно принятыми стандартами, многие из нас работают в условиях, напоминающих Генеральную Ассамблею ООН.

Команды, состоящие из представителей разных культур, возрастов, опыта и знаний, обычно гораздо интереснее и в конечном итоге формируют питательную среду для создания прорывных решений.

Но будьте осторожны! В нескольких командах, с которыми я работал, мне пришлось деликатно разбивать подгруппы, которые сформировались на базе общего географического и культурного бэкграунда. Складывание подобных подгрупп приводило к ситуациям, в которых в открытой рабочей зоне говорили на разных языках (и я не про языки программирования).

По своей сути люди тяготеют к тем, у кого с ними много общего. Но я выступаю активно против подгрупп внутри scrum-команды. Это вызывает чувство отчуждения, не говоря уже о потенциальной потере знаний из-за того, что другие не понимают их язык.

Кроме того, я обнаружил, что участникам таких разнородных scrum-команд следует быть осторожными и выбирать выражения, чтобы шутки не показались грубыми, а дружеское подтрунивание оскорбительным.

ПРАВИЛА СОВМЕСТНОЙ РАБОТЫ

Лисса Адкинс, автор «Коучинга agile-команд», рекомендует формировать то, что она называет «правилами совместной работы», для разрешения ситуаций, подобных описанным выше.

Например, в одной из команд, с которой я работал, у нас был набор «домашних правил» (см. рис. 2.7), которым мы все следовали.

Рис. 2.7. Визуализация правил совместной работы напоминает участникам команды, как они должны относиться друг к другу

ВСЕ ЗА ОДНОГО И ОДИН ЗА ВСЕХ!

Я всегда подмечаю тот волшебный момент, когда понимаю, что я больше не среди группы индивидуальностей, а в единой, сплоченной команде. Момент, когда этот «пазл» складывается, порой возникает во время интенсивной, но дружелюбной дискуссии. Однако чаще это случается, когда все веселятся, наслаждаясь шуткой или двумя. Видеть, как пазл складывается, — мой любимый момент. Именно тогда возникает братство мушкетеров и все участники команды чувствуют, что они в одной лодке, знают и верят, что победят или проиграют вместе как команда.

ЗАКЛЮЧЕНИЕ

Три лайфхака этой главы посвящены выбору тактики, инструментам и советам, которые помогут вам и вашей организации оценить базовые компетенции и способности, необходимые команде для эффективной работы даже в экстремальных ситуациях. Давайте вспомним, какие вопросы мы рассмотрели.

Лайфхак 4. Искусный scrum-мастер

- что значит быть настоящим лидером-слугой;
- ключевые навыки, необходимые выдающемуся scrum-мастеру;

- врожденные установки, которые демонстрирует настоящий scrum-мастер.

Лайфхак 5. Рок-звезды или студийные музыканты?

- потенциальные проблемы ярких разработчиков-одиночек (рок-звезд);
- почему мы хотим, чтобы наши разработчики думали и вели себя как студийные музыканты;
- набор ценностей, которые должны сформировать профессиональный тип вашей команды.

Лайфхак 6. Выстраиваем команду

- рекомендуемый размер команды и соотношение групп разработчиков в ней;
- проблема частичной занятости и как ею управлять;
- факторы, которые важно учитывать, принимая решение, может ли scrum-мастер работать более чем с одной командой.

Глава 3

ЗАЩИЩАЕМ КОМАНДУ И ПЛАНИРУЕМ СПРИНТ

Итак, ваша компания готова применить Scrum и вы собрали команду, обладающую всеми необходимыми компетенциями и демонстрирующую приверженность работе. Настало время двигаться дальше.

Три следующих лайфхака не только помогут задать команде правильный курс, но и дадут несколько советов и рекомендаций, как удержать проект на плаву.

Лайфхак 7 «Закладываем фундамент для Scrum» содержит несколько рекомендаций, как заложить прочный фундамент, который обеспечит эффективность вашей scrum-команды.

Лайфхак 8 «Планируем спринт и двигаемся по плану» дает конкретные практические советы, как эффективно спланировать спринт.

В лайфхаке 9 «Разоблачаем препятствия» я привожу рекомендации, с помощью которых вы сможете контролировать те трудности, которые возникают во время спринта и мешают достичь целей.

ЛАЙФХАК 7. ЗАКЛАДЫВАЕМ ФУНДАМЕНТ ДЛЯ SCRUM

Работа со scrum-командами чем-то похожа на химию: так же как в научной лаборатории, успешные «химические реакции» намного легче запустить, если у вас есть все необходимые реактивы и условия для работы. Как проницательно отмечает Майк Кон: «Чтобы пожинать плоды Agile, нужны глубокие изменения. Эти изменения требуют многого не только от разработчиков, но и от всей компании».

Давайте рассмотрим некоторые ключевые организационные пред-посылки и особенности рабочей среды, которые станут частью вашего плана по применению Scrum.

ПОДДЕРЖИВАЙТЕ СТАБИЛЬНОСТЬ КОМАНДЫ

Том Демарко и Тимоти Листер заклинают любую организацию, которая хочет создавать сплоченные команды: «Оберегайте и защищайте успешные команды».

Признаю, я работал со scrum-командами, проекты которых считаю не слишком успешными. Я легко могу назвать причину неудовлетворительного результата: это моя неспособность как scrum-мастера поддерживать сплоченность команды в течение всего проекта. Такая проблема часто возникает, когда ключевых разработчиков перебрасывают с текущего проекта на более срочный (см. рис. 3.1). Иногда эту ситуацию может усугублять затянувшаяся реорганизация компании (которую все чаще и чаще можно наблюдать во время мирового финансового кризиса). Совокупность этих двух факторов становится настоящим вызовом для тех, кто призван оберегать и сохранять сильные команды.

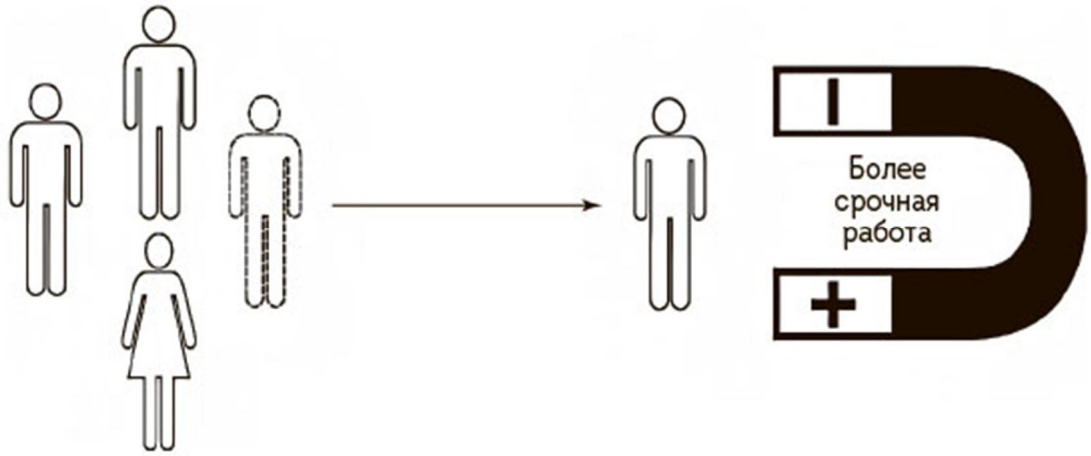

Рис. 3.1. Остерегайтесь «более срочных» проектов, которые пытаются утащить участников вашей команды

Демарко и Листер так оценивают потери от ротации сотрудников: «Объективно в нового участника команды вы будете вкладываться

приблизительно три месяца, которые вы потратите на обучение и адаптацию новичка». И эта оценка не учитывает такие мало поддающиеся измерению издержки, как снижение скорости работы, демотивация участников команды или потеря ценных, но не задокументированных знаний.

ОБУСТРАИВАЙТЕ РАБОЧУЮ СРЕДУ

Без сомнения, многими моими самыми успешными scrum-проектами становились те, в которых я смог физически отделить scrum-команды от остальной организации.

Демарко и Листер разъясняют, почему это правильно:

Почти всегда имеет смысл вывести команду проекта… из общего корпоративного пространства. Работа в своем особом пространстве позволяет сохранить больше энергии и, соответственно, демонстрировать более высокие результаты. Люди меньше страдают от шума, меньше отвлекаются и меньше страдают от фрустраций.

Я работал со scrum-командами, которые были вынуждены взаимодействовать в большом open space — рядом с отделом продаж, сотрудники которого висели на телефонах целый день, — и так каждый день. У меня также были команды, пострадавшие от отдела снабжения, который не позволил им перенести несчастный маленький круглый стол для встреч к себе. Я мог бы и далее приводить примеры, но суть в том, что отделение и независимость своего рабочего пространства внутри офиса — это святой Грааль.

Независимо от того, сможете ли вы достичь этой высокой цели, вы должны сделать все от вас зависящее, чтобы scrum-команда находилась в одном помещении и все ее участники сидели рядом. Scrum, безусловно, может работать и для распределенных команд, которые не собрать в едином пространстве, однако это не оптимальный вариант.

Помимо очевидных преимуществ работы в общей среде, когда вы находитесь в непосредственной близости друг от друга, Джеймс Шор и Шейн Уорден приводят еще более вескую причину, почему команда должна сидеть вместе:

Сидеть вместе — самый эффективный способ развить эмпатию (способность сопереживать), который я знаю. Каждый член команды видит, что остальные работают так же усердно[31].

В лайфхаке 3 более детально рассмотрены другие ключевые моменты организации пространства, благоприятного для Scrum.

ОЦЕНКА — ЭТО НЕ ГАРАНТИЯ

Как выглядит неудачный сценарий? Заказчик проекта подходит к случайно выбранному участнику команды и спрашивает, сколько времени займет разработка фичи XYZ. Тот снимает наушники; отрывается от того, над чем работает; поднимает взгляд к потолку и выдает очень приблизительную оценку — чтобы хоть как-то успокоить заказчика. И — подумать только! — оценка оказывается неточной. А заказчик начинает давить на всю команду, чтобы та — в соответствии с этой оценкой — выполнила взятые на себя «обязательства». Но пошло оно к чертям, если цена таких «обязательств» — переработки и пропущенный выпускной вашего ребенка!

Экстренное сообщение: оценка ничего не гарантирует. В противном случае не было бы необходимости в самом слове «оценка». Оценка — это только прогноз, основанный на доступной информации и вводных данных в заданный момент времени. Это должно быть четко понято стейкхолдерами проекта, прежде чем он будет запущен.

РАБОТА ПО ЛЮБВИ

Мэри и Том Поппендик, авторы Leading Lean Software Development, выделяют два типа компаний: компании, где работают за деньги, и компании, где работают по любви (по взаимности).

В компаниях, где работают исключительно за материальное вознаграждение, сотрудники считают так: «Я прихожу на работу, и вы мне платите за мое время. Если вы хотите от меня большего, платите больше». В компаниях, где работают «по любви», совершенно другие стандарты: «Я буду относиться к вам так же, как вы ко мне. Я ожидаю справедливой компенсации за свою работу. Однако, если вы хотите от меня внимания и лояльности, тогда и компании следует проявлять заботу и поддержку по отношению ко мне, помогая полностью раскрыть мой потенциал32.

Лучшие scrum-команды состоят из преданных и внимательных людей. Очевидно, что компании, использующие модель «работы по любви», в отличие от тех, где люди работают исключительно за деньги, более успешны при использовании Scrum, особенно в долгосрочной перспективе.

ПОДДЕРЖИВАЙТЕ УСТОЙЧИВОЕ РАЗВИТИЕ

В лайфхаке 1 я рассказывал, что один из ведущих принципов Scrum — работа в устойчивом темпе.

Это же отмечает Роман Пихлер в книге «Управление продуктом в Scrum»:

Разработка продукта похожа на марафон. Если вы хотите прийти к финишу, вы должны набрать устойчивый темп. Многие владельцы

продукта совершают ошибку и давят на команду, чтобы она взяла больше работы[33].

Любая компания, которая поддерживает культуру сверхурочной работы и не просто поощряет ее, а еще и оплачивает в смехотворном размере, нарушает один из принципов Scrum (или, если уж на то пошло, любого другого agile-фреймворка). Сверхурочная работа должна быть исключением, а не правилом, и, как признает Кент Бек в книге «Экстремальное программирование», это «симптом серьезной проблемы в проекте»[34], а не обычное дело.

ЗАПУСКАЕМ ПИЛОТНЫЙ ПРОЕКТ

Я не сторонник резкого внедрения Scrum (одним махом), хотя и у него есть свои преимущества. Вместо этого я предпочитаю запустить пилотный проект. Рекомендую такой подход даже в тех случаях, когда руководство рвется в бой и хочет сразу перевести всю компанию на работу по Scrum. Почему я советую инвестировать дополнительное время в пилотный проект? Потому что это отличный способ подтвердить, что Scrum работает, и убедить руководство «купить» эту идею. Майк Кон прекрасно объясняет:

Пилотный проект запускается для того, чтобы «проторить путь» следующим проектам; он становится проводником изменений и показывает, как делать что-то новое… В рамках отрасли в целом у нас достаточно доказательств, что Scrum работает; но что действительно нужно узнать компаниям — как Scrum будет работать именно у них[35].

Я всегда запускаю пилотные проекты, прежде чем переходить к работе с бо́льшим количеством команд. Собственно говоря, без свободы экспериментировать, которую открывает пилотный проект, я не уверен, что эта книга вышла бы в свет.

Заманчиво выбрать в качестве пилотного проект, который не имеет большой ценности для компании и, следовательно, не несет с собой практически никаких рисков. Но это иллюзия. Шор и Уорден не советуют так делать:

Избегайте делать пилотным проект, имеющий низкое значение для компании. Вам будет сложно привлекать заказчиков и эффективно настраивать процессы. И даже если вы с методологической точки зрения достигнете успеха, компания может посчитать проект провальным.

Возвращаясь к обсуждению стабильности команд и неуспешных проектов, в которых я участвовал, могу сказать, что причину, по которой я не смог сплотить команды, следует искать именно в пилотных проектах. Те пилотные проекты, над которыми мы работали, не имели ни высокого приоритета, ни большой ценности для компании. Когда наступил критически важный момент распределения ресурсов

и пришлось выбирать между пилотным и «более важными» проектами, догадайтесь, кто вынужден был уступить.

Как долго должен продолжаться пилотный проект? Если вы внимательно читаете эту главу, то уже знаете, что пилотный проект не должен отличаться от любого другого значимого проекта. Как утверждает Роман Пихлер: «В Scrum нет правила, которое определяло бы длительность проекта. Но, как правило, он длится от 3 до 6 месяцев»[36].

СФОРМИРУЙТЕ РЕАЛИСТИЧНЫЕ ОЖИДАНИЯ

Изменения требуют времени, и часто приходится делать шаг назад, чтобы потом продвинуться на два вперед. Применение Scrum требует значительных изменений в мышлении и отказа от давно укоренившихся привычек. А это не происходит в одночасье.

Должно пройти время, прежде чем разработчики начнут чувствовать себя комфортно в кросс-функциональных командах, а старый командно-административный подход исчезнет. Вот почему компаниям следует отбросить наивность и не рассчитывать, что в ближайшем будущем они увидят невероятные результаты.

Быть терпеливыми и пестовать ваши изменения и команду — таковы правила игры. И те компании, которые умеют оказывать поддержку и создавать благоприятную среду, пожнут плоды своих усилий в ближайшем будущем.

ЛАЙФХАК 8. ПЛАНИРУЕМ СПРИНТ И ДВИГАЕМСЯ ПО ПЛАНУ

Вспоминая аналогию с химической лабораторией, Демарко и Листер рекомендуют следующий «химический элемент» для создания процветающей организации — чувство выполненной работы. Я полностью согласен с их советом и предлагаю еще один, дополняющий элемент — старт с чистого листа.

К счастью, формат спринта позволяет как перезагружаться и каждый раз делать новый старт, так и чувствовать удовлетворение от завершения работы в конце спринта. В этом лайфхаке мы рассмотрим один из процессов Scrum, который позволяет делать такие регулярные перезагрузки, — планирование спринта.

Благодаря совместному перепланированию целей для нового спринта каждые несколько недель команда может каждый раз начинать заново — вместо того, чтобы застревать в кажущейся бесконечной и однообразной текущей работе. Более того, отсутствие регулярных и ожидаемых встреч по планированию спринта приводит к полной неразберихе — участники команды начинают собираться спонтанно для планирования и проектирования.

УТОЧНЕНИЕ БЭКЛОГА ПРОДУКТА

Прежде чем команда соберется для планирования, рекомендую убедиться, что бэклог продукта должным образом уточнен. Во-первых, удостоверьтесь, что владелец продукта (с соответствующей помощью) не только определил приоритетные задачи для предстоящего спринта, но и расписал их достаточно подробно. Настолько подробно, чтобы разработчики могли сразу приступить к их реализации. Например, включил более подробные критерии приемки, а также макеты и схемы, если это в принципе возможно (см. лайфхак 11). Во-вторых, мотивируйте владельца продукта вместе с тестировщиком написать несколько начальных тест-кейсов (основанных на критериях приемки). Они помогут наиболее полно описать основную работу требуемого функционала.

ЦЕЛИ — ЭТО ХОРОШО

Большинству из нас нравится работать, имея четкие цели, вот почему так важно определить основную цель спринта, которая будет лейтмотивом проходить через весь спринт. Она задает тему текущего спринта. Например, цель спринта — улучшить механизм обмена сообщениями. Это не значит, что задачи, не связанные с этой целью, не включаются в спринт. Это значит, что бо́льшая часть работы должна быть направлена на улучшение механизма обмена сообщениями. Цель спринта также помогает в принятии решений, ведь каждый участник спринта остается сфокусированным на цели, не отвлекается и не отклоняется от нее.

СКОЛЬКО ДОЛЖЕН ДЛИТЬСЯ СПРИНТ?

Раньше считалось, что продолжительность спринта должна быть 30 дней — не больше и не меньше. Теперь все намного гибче: почти везде признают, что длина спринта может варьироваться от команды к команде. Поговорите с любой scrum-командой, и увидите, что подавляющее большинство спринтов длятся от одной до четырех недель. Я пробовал все четыре варианта. По моему мнению, неделя — это слишком мало, а четыре недели — чересчур много, так что я выбираю между двумя и тремя неделями. Для нового проекта я принимаю решение на основе двух факторов:

- **предпочтения команды**. Некоторые команды предпочитают более долгий спринт, чтобы успеть набрать темп, в то время как другие выбирают более простое планирование, которое как раз достижимо при более коротком спринте;
- **изменчивость требований**. Если владелец продукта достаточно часто меняет требования из-за специфики продукта или рыночных условий, я бы рекомендовал более короткий спринт (см. рис. 3.2).

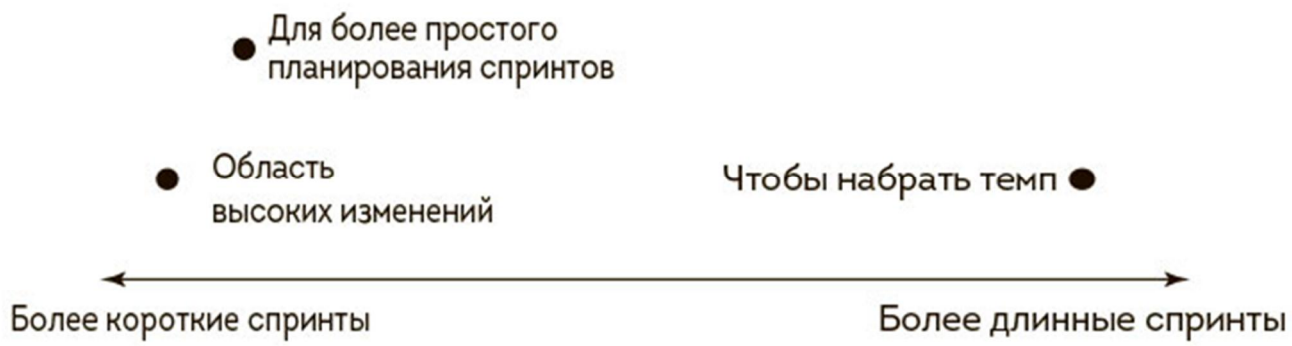

Рис. 3.2. Факторы, которые нужно учитывать при определении длины спринта

Очень важный момент: когда предпочтительная длина спринта выбрана (поначалу придется немного поэкспериментировать), ее нужно зафиксировать и больше не менять (за редким исключением). Есть несколько причин избегать эпизодического изменения длины спринта:

- Определенный ритм помогает команде лучше понимать, как самостоятельно набрать необходимую скорость для фокусировки на цели.
- Метрику скорости (см. лайфхак 13) работы команды можно рассчитать только при одинаковой длительности спринта. В противном случае эта метрика становится менее показательной и ее труднее вычислить.
- Если вы меняете продолжительность спринтов, то обзор спринта, ретроспектива и планирование каждый раз будут приходиться на разные дни недели. Такая нерегулярность может стать головной болью для команды, особенно если вы делите переговорные с другими отделами компании.

ПЛАНИРУЕМ ЕМКОСТЬ СПРИНТА

Перед тем как погрузиться в процесс планирования, команда должна определить емкость спринта. Помните: не все участники команды смогут работать на полную каждый спринт. Некоторые вынуждены одновременно трудиться над несколькими проектами — это неидеальная ситуация, но она встречается достаточно часто (см. лайфхак 6). Если это ваш случай, убедитесь, что разработчики, задействованные в нескольких проектах, не перегружены. И не забудьте учесть государственные праздники, обучение и отпуска сотрудников.

Не попадитесь в ловушку иллюзии, будто те, кто работает только над вашим проектом, потратят весь рабочий день на задачи спринта.

Например, в команде, с которой я недавно работал (с использованием двухнедельных спринтов), емкость спринта на одного штатного разработчика рассчитывалась так: 9 дней × 6 часов в день = 54 часа.

Мы отталкивались от девяти дней, потому что время, затрачиваемое на планирование спринта, обзор спринта и ретроспективу, суммарно

составляет один рабочий день. Цифра 6 часов появилась, поскольку в течение 8-часового рабочего дня сотрудник сфокусированно работает над задачами спринта только 6 часов. Оставшееся время, как правило, уходит на другую активность, не связанную с текущим спринтом, — уточнение бэклога продукта (для подготовки к следующему спринту) или более общие задачи, связанные с работой в компании (например, ответы на электронную почту или помощь коллегам не из scrum-команды). Обратите, пожалуйста, внимание, что емкость спринта на одного разработчика в день будет варьироваться в зависимости от команды и процессов в организации. Иными словами, не следует считать емкость 6 часов в день универсальным стандартом. Чтобы определить, какая именно емкость спринта подходит вашей команде, я рекомендую использовать статистику прошлых спринтов (см. лайфхак 19).

Теперь давайте рассмотрим непосредственно сам процесс проведения встречи по планированию спринта. Я бы разделил его на две части: «что» и «как».

ЧАСТЬ 1. ЧТО

Эта часть включает все, что связано с презентацией владельцем продукта наиболее приоритетных элементов бэклога и ответом на любые уточняющие вопросы команды. Презентация выполняется для каждого из элементов бэклога, которые необходимо завершить в предстоящем спринте. Я рекомендую использовать метрику скорости работы команды (см. лайфхак 13) для того, чтобы определить, сколько элементов бэклога владелец продукта должен подготовить для рассмотрения на планировании.

ЧАСТЬ 2. КАК

Затем наступает черед команды разработчиков разделить элементы бэклога на детализированные технические задачи и оценить каждую с точностью до часа. Хотя эти оценки иногда могут быть не точны, они помогают команде принимать более обоснованные решения и обретать бо*л*ьшую уверенность в том, что будет сделано к концу спринта. Как поясняет Майк Кон:

Задача не в том, чтобы определить часы, однако именно часы часто становятся гарантией того, что мы обсудили все технические и продуктовые детали задач. Обсудили на таком уровне, который позволит начать спринт с ощущением, что мы сможем закончить всю запланированную работу37.

Я не ожидаю, что владельцы продукта будут присутствовать на второй части (если только сами не захотят). Как я говорил ранее, я не считаю необходимым, чтобы владельцы продукта обязательно находились вместе с командой, однако они всегда должны быть на связи —

на случай, если команде потребуется уточнить какие-то детали. Ничто так не стопорит планирование спринта, как недоступность владельца продукта.

Хотя мне нравится использовать планирование на основе метрики скорости работы команды для части 1, также я ценю *планирование на основе обязательств* — чтобы определить количество задач, которые могут быть включены в бэклог спринта. Команде разработки предстоит выполнить следующие шаги:

1. Начните с максимально приоритетных элементов бэклога.
2. Разбейте элементы бэклога на задачи с оценкой в часах.
3. Выявите зависимости в задачах.
4. Продолжайте этот цикл, пока совокупная оценка задач не достигнет размера емкости спринта команды.
5. Если результат планирования, полученный при использовании подхода на основе скорости команды, не совпал с результатом планирования на основе обязательств, позовите обратно владельца продукта (если он уже покинул помещение) для того, чтобы:
    - добавить дополнительный элемент бэклога, если в спринте еще осталась свободная емкость;
    - объяснить владельцу продукта, почему в бэклог спринта попало меньше элементов, чем изначально планировалось, если емкость спринта заполнилась быстрее, чем изначально рассчитывалось.

ЧТО ТАКОЕ ЗАДАЧИ?

Обычно я использую несколько критериев для создания задач:

- Каждая отдельно взятая задача должна быть небольшой, самостоятельно тестируемой частью элемента бэклога (см. лайфхак 10). И каждая задача должна пройти через все этапы, чтобы получить статус критерия готовности — и соответствовать всем критериям, согласно которым задача будет считаться закрытой.
- Каждая задача должна занимать не более 8 часов (чем короче, тем лучше).
- Хотя несколько разработчиков могут работать над одним и тем же элементом бэклога, только один разработчик или пара38 могут работать над одной задачей (см. рис. 3.3).
- Учитывайте задачи при подготовке обзора спринта (см. лайфхак 22), например при сборе информации для демо.

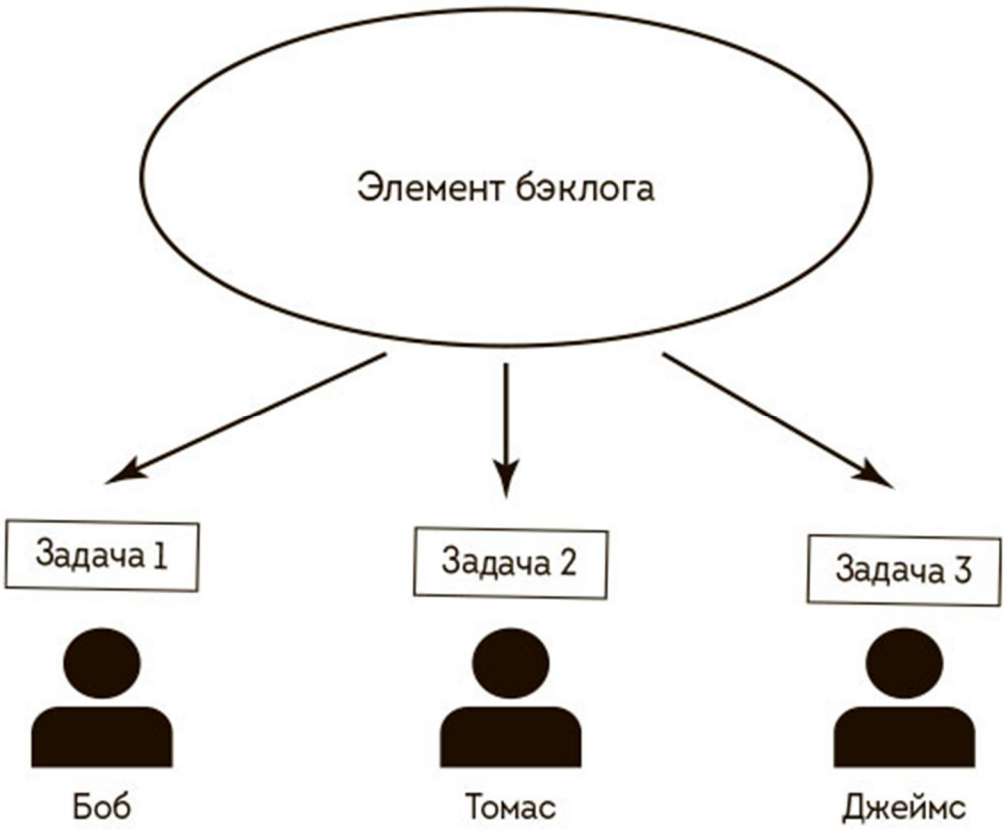

Рис. 3.3. Хотя над одним элементом бэклога могут работать разные разработчики, над одной задачей может работать только один разработчик

Теперь, после планирования, у вас сформирован бэклог спринта, включающий задачи и их оценки. Суммировав все оценки задач, вы получите оставшееся время в спринте. Это время может пересчитываться каждый день, исходя из завершенных задач, и отслеживаться на диаграмме сгорания задач (см. лайфхак 19). Ежедневно перед уходом домой каждый в команде разработки должен обновлять время, исходя из завершенных и оставшихся задач, чтобы поддерживать диаграмму сгорания в актуальном состоянии.

ПРАВИЛЬНОЕ КОЛИЧЕСТВО ТРЕБОВАНИЙ

В идеальном мире после планирования спринта — когда все детали уже обговорены — у вас есть набор ясных и понятных элементов бэклога, которые ожидаемо будут выполнены командой в текущем спринте. Периодически вы будете обнаруживать, что после планирования осталась небольшая незаполненная емкость спринта, которой недостаточно, чтобы выполнить полноценный элемент бэклога. Это нормально. Просто дайте знать команде, что в оставшееся время можно будет начать следующий по приоритетности элемент бэклога. Однако не ожидайте, что он будет завершен в этом спринте. Я предпочитаю именно такой подход — вместо того, чтобы искать маленький элемент бэклога, который займет ровно столько времени, сколько осталось в емкости спринта, и обнаружится при этом где-нибудь в самом низу бэклога

продукта. Я считаю, важнее фокусироваться на самых приоритетных элементах бэклога с наибольшей ценностью для бизнеса.

5 «П»

Как говорят в британской армии, «**п**равильное **п**ланирование и **п**одготовка **п**редотвращают **п**оражение». Тщательное и хорошо проведенное планирование спринта важно для того, чтобы спрогнозировать работы настолько точно, насколько это возможно.

Спринт не всегда идет по плану, и порой приходится вносить коррективы. Однако, если планирование спринта проведено грамотно, цели команды будут понятны всем, и это упростит координацию и повысит точность результатов.

ЛАЙФХАК 9. РАЗОБЛАЧАЕМ ПРЕПЯТСТВИЯ

Вы обучили новых scrum-бойцов, и они готовы к своей первой миссии. Команда рвется в бой, проект запущен. Дела идут хорошо. Вы проводите ежедневный scrum, сервер непрерывной интеграции гудит, задачи перемещаются по доске — жизнь прекрасна! Затем, практически из ниоткуда, начинают лететь пули, взрываться мины и ваши бойцы перестают двигаться вперед. Настало твое время, scrum-мастер! Пора выходить!

Ладно, на самом деле вашей команде угрожает совсем не шквал вражеского огня. Это могут быть постоянно падающая сборка39, чинящий препятствия заказчик проекта или, возможно, потеря ключевого участника команды. Все, что препятствует прогрессу команды и с чем нужно разобраться, и есть приоритет для scrum-мастера.

ВЫЯВЛЯЕМ ПРЕПЯТСТВИЯ

Давайте начнем с определения, что такое *препятствие.* Я выбрал вот такое:

Препятствие — это любое событие, которое осложняет разработчику работу и уменьшает его ожидаемую емкость спринта.

Вспомните лайфхак 8: неразумно планировать разработчику, занятому только на вашем проекте, емкость спринта 8 рабочих часов (для 8-часового рабочего дня). «Почему нет?» — спросите вы. Что ж, будем реалистами: сотрудники не тратят каждую секунду рабочего времени на выполнение операционных задач текущего спринта. Мы должны брать в расчет совещания; затянувшуюся совместную работу и взаимопомощь; незапланированные события в компании; перерывы, чтобы «проветрить голову», — и это только часть. Не поймите меня неправильно: в определенные дни некоторые участники команды смогут

сфокусироваться на задачах спринта и максимально приблизятся или даже превысят показатель 8 часов работы. Однако в другие дни постоянные перерывы не позволят даже пару часов сосредоточенно работать над целями спринта.

РАЗНООБРАЗИЕ ФОРМ И РАЗМЕРОВ

Препятствия могут быть абсолютно любых форм и размеров. Вот небольшой список типичных препятствий (и операционных, и системных), которые нужно внимательно отслеживать.

Раздутые встречи. Проекты, в которых применяют Scrum, обычно не нуждаются в этих излишне растянутых мероприятиях; как правило, их инициируют другие отделы или же они возникают из-за непредвиденных обстоятельств.

Болезнь. Недуг настигает без предупреждения. Вы мало что можете с этим поделать и я настоятельно не рекомендую переносить болезнь на ногах. Это глупо. Качество работы страдает, а зараза быстро распространяется по офису, на что досадуют все вокруг.

Падающие сборки. Без «здоровой» сборки разработка не может вестись непрерывно. Поправить падающую сборку — приоритет № 1 для любого участника команды.

Проблемы с оборудованием. Что бы это ни было — отказ оборудования, проблемы с программным обеспечением, сбои в сети, — любые подобные проблемы могут серьезно затруднить прогресс и вызвать глубокое разочарование у команды.

Ненадежный поставщик. Это, возможно, один из самых сильных демотиваторов, ведь это препятствие не находится под контролем scrum-мастера и команды. Плохо поддерживаемые компоненты или расширения могут высасывать время из ваших спринтов.

Неупорядоченный бэклог продукта. Спринт никогда не должен начинаться без того, чтобы владелец продукта четко понимал, какие требования должны попасть в бэклог спринта. Более того, эти требования должны быть достаточно подробно описаны, чтобы команда разработки могла сразу приступить к работе над ними. Если эти требования не готовы перед планированием, спринт точно не начнется гладко (см. лайфхак 11).

Владелец продукта тянет с принятием ключевых решений (из-за отсутствия на работе или отсутствия полномочий). Владелец продукта должен быть на связи с командой в течение всего спринта, чтобы отвечать на вопросы, связанные с бэклогом спринта. Если его часто нет или он должен постоянно согласовывать свои решения с кем-то еще, работа scrum-команды может быть парализована неопределенностью.

Индивидуальная мотивация. Многие компании практикуют аттестацию своих сотрудников и связанные с этим индивидуальные схемы вознаграждения, основанные исключительно на личной эффективности. Надеюсь, вы уже усвоили, что в scrum-команде нет никакого «я». Поэтому до тех пор, пока в аттестациях сотрудников фокус не будет смещен на командную работу, компания будет посылать противоречивые сигналы участникам scrum-команды.

КОНТРОЛИРУЕМ ПРЕПЯТСТВИЯ

Мне нравится использовать пятиступенчатый подход при работе с препятствиями: подтвердить, приоритизировать, устранить, маркировать, сделать выводы.

Подтвердить. Очевидно, вначале нужно выяснить, что за препятствие перед вами. Как правило, о них говорят на ежедневном scrum, но безотлагательные препятствия нужно выявлять, не дожидаясь ежедневного scrum. Ретроспектива спринта может вскрыть будущие препятствия, которые ускользнули от внимания в текущем спринте. Все препятствия необходимо отслеживать и мониторить до тех пор, пока они не будут устранены.

Приоритизировать. Если перед вами возникло несколько препятствий одновременно, вы сможете разобраться только с одним или двумя за раз (вы ведь не супермен). Оценив важность и срочность возникших препятствий, вы сможете понять, с чего именно начать.

Устранить. В идеальном мире scrum-команда может устранять любые препятствия, возникающие у нее на пути, но, к сожалению, так не всегда получается в реальности. Чтобы избежать заминок, важно понимать, когда обращаться за помощью к другим командам, чтобы вернуться в штатный режим работы.

Маркировать. И scrum-команда, и стейкхолдеры должны быть проинформированы о любых возникающих препятствиях. Сюрпризов не должно быть ни для владельцев продукта, ни для заказчиков проекта, ведь речь идет о сорванных планах — в их распоряжении должно быть как можно больше времени, чтобы смягчить цепную реакцию последствий.

Сделать выводы. Ретроспектива спринта (см. лайфхак 23) — основная встреча, на которой анализируются препятствия, возникавшие у команды в ходе спринта. Очень важно извлечь уроки, чтобы избежать их повторения и/или зафиксировать способы их разрешения. Это поможет снизить негативное влияние от аналогичных препятствий, если они появятся вновь.

БЛОКЕР ИЛИ ПРЕПЯТСТВИЕ?

Многие команды используют термины *блокер* и *препятствие* как синонимы, но я бы их различал (см. рис. 3.4). Это помогает отделить помеху, которая *остановила* прогресс по конкретной задаче, но не обязательно замедлила общее движение команды (блокер), от помехи, которая *замедлила* прогресс команды в спринте (препятствие).

Рис. 3.4. Блокер влияет только на одну задачу, а препятствие действует как парашют, замедляя общий прогресс команды

Типичный блокер возникает, когда задача имеет какую-то зависимость от другой задачи, чье исполнение затянулось. Короткий, временный блокер встречается достаточно часто и не должен вызывать особого беспокойства. В большинстве случаев, пока решается вопрос с зависимостью, можно работать над другими задачами спринта. Тем не менее у вас должна быть четкая картина всех задач с блокерами, неважно, насколько короткими являются эти помехи. Я отслеживаю все задачи с блокерами следующим образом: приклеиваю стикер с такой задачей под углом 45 градусов так, чтобы он выглядел как ромб и выделялся на доске. Это четкий сигнал: нужно немедленно изучить блокер и убедиться, что он будет устранен в кратчайшие сроки.

ОЦЕНИВАЕМ МЕСТНОСТЬ

Если вы похожи на меня, то будете ошеломлены, когда обернетесь и вспомните все препятствия, возникавшие в течение проекта. Зачастую, только проанализировав список всех препятствий, вы сможете оценить, насколько сложную местность пришлось преодолеть вашим scrum-бойцам.

Этот перечень может оказаться неоценимым подспорьем, если вам придется защищать команду от претензий из-за задержек, повлиявших на даты релизов. Однако я убежден, если вы начинаете устранять препятствия, как только те поднимают свои уродливые головы, таких неприятных ситуаций будет немного и между ними будет большой лаг.

ЗАКЛЮЧЕНИЕ

В лайфхаках этой главы мы рассмотрели вопросы тактики и выбора инструментов, а также обсудили советы, как задать scrum-команде направление и не позволить с него сойти. Давайте вспомним, о чем мы говорили.

Лайфхак 7. Закладываем фундамент для Scrum

- размещение участников команды в одном пространстве (когда это возможно);
- важность корпоративной культуры для процветания Scrum;
- как и почему пилотные scrum-проекты могут помогать в масштабировании Scrum на всю компанию.

Лайфхак 8. Планируем спринт и двигаемся по плану

- факторы, которые необходимо учитывать при выборе длины и определении цели спринта;
- как определить реалистичную емкость спринта;
- способы структурировать планирование.

Лайфхак 9. Разоблачаем препятствия

- разница между блокерами и препятствиями;
- типы препятствий, которые нужно отслеживать;
- зачем нужен список препятствий.

Глава 4

УТОЧНЕНИЕ ТРЕБОВАНИЙ

К счастью, нам больше не нужно гадать на кофейной гуще, чтобы подготовить предварительное техническое задание. Но, как уже говорилось, доведение требований до исполнителей и проверка на корректность по-прежнему вызывают затруднения.

Три лайфхака этой главы станут для вашей команды руководством, как формировать и управлять требованиями и связанными с ними критериями готовности.

Лайфхак 10 «Структурируем пользовательские истории» дает рекомендации, как командам разбивать требования, подготовленные к спринту, на задачи размера «бери и делай».

Лайфхак 11 «Определяем критерии готовности» предлагает подумать о том, как помочь scrum-команде устанавливать, видоизменять и развивать критерии готовности.

Лайфхак 12 «Прогрессирующее откровение»40 посвящен тому, как команде избежать потерь с помощью промежуточного контроля.

ЛАЙФХАК 10. СТРУКТУРИРУЕМ ПОЛЬЗОВАТЕЛЬСКИЕ ИСТОРИИ

Scrum не предписывает использовать какой-то определенный формат пользовательских требований, которые затем становятся элементами бэклога. Однако можно с уверенностью утверждать, что де-факто стандартом пользовательских историй стал формат, изложенный в основополагающей книге Майка Кона «Пользовательские истории»41.

Рискну предположить, что все, кто сейчас держит в руках эту книгу, читали «Пользовательские истории» и/или работали с теперь уже повсеместно распространенным форматом:

«Как <роль / персонаж пользователя> … я хочу <что-то получить> … для того, чтобы <с такой-то целью>…»

Я не собираюсь торить эту лыжню заново и обсуждать преимущества пользовательских историй или способы разбиения больших историй на меньшие: об этом прекрасно пишет Майк Кон. Вам стоит ознакомиться с его книгой, если вы этого пока не сделали.

В этом же лайфхаке я бы хотел глубже исследовать те аспекты пользовательских историй, которые не так хорошо разъяснены. Например, взаимосвязи историй и задач или способы обработки «технических историй».

ДЕКОМПОЗИРУЕМ ИСТОРИИ

Помимо обычных, готовых к спринту пользовательских историй (см. лайфхак 11) часто используются два других типа, которые помогают собрать все требования: эпики и задачи.

Майк Кон под эпиком понимает «пользовательскую историю, на разработку и тестирование которой вы потратите более одного или двух спринтов».

Когда вы только начинаете формировать продуктовый бэклог, большинство требований могут быть по своим характеристикам больше похожи на эпики. Помните, что в Scrum требования эмерджентны, поэтому нет необходимости детализировать все пользовательские истории. Подобные пользовательские истории должны быть только у самых приоритетных элементов бэклога, над которыми вы будете работать ближайшие один-два спринта.

Чтобы разбить эпики на готовые для включения в спринт пользовательские истории (наш следующий уровень), я советую обратиться к книге Майка Кона. Там вы прочтете про несколько вариантов разделения, включая разделение по функциональности, по дополнительным ролям пользователей, по набору данных и операционных границ (помимо прочих вариантов).

Задачи составляют третий, наиболее детализированный уровень. И после того, как они идентифицированы и определены, команда готова приступить к созданию действительно отличного программного обеспечения.

НАРЕЗАЕМ ЗАДАЧИ

Формулирование удобоисполнимых, готовых ко взятию в спринт пользовательских историй — только часть проблемы. Следующий этап — рассмотрение этих высокоприоритетных историй на планировании спринта (см. лайфхак 8), где они станут элементами бэклога спринта.

Бэклог спринта, как правило, формируется не только из подобных пользовательских историй, но и из их «дочек» (появившихся в процессе планирования), которые обычно можно отнести к категории задач. Это конкретизированные и детально описанные части истории, над которыми команда работает в спринте и которые отслеживает с помощью стикеров на доске с задачами (см. лайфхак 21). Я склоняюсь к тому, что на эти задачи должно уходить от 2 до 8 часов. Чуть дольше — и они становятся трудноуправляемыми.

Если разделение эпиков на пользовательские истории можно назвать искусством, то декомпозиция историй на детализированные задачи потребует от вас таланта суши-шефа.

Распространенная декомпозиция задач, которую я наблюдал (и использовал в прошлом), выглядит примерно так:

Пользовательская история

«Как новый пользователь, я бы хотел зарегистрироваться на сайте XYZ, чтобы начать использовать его потрясающие возможности».

Задача 1. Дизайн функциональных тестов.

Задача 2. Создание тестовых данных.

Задача 3. Разработка базы данных.

Задача 4. Разработка бизнес-логики.

Задача 5. Разработка пользовательского интерфейса.

Задача 6. Разработка функционального автотестирования.

Эта декомпозиция выглядит логичной и убедительной и даже может отлично сработать. Но знаете, что она мне напоминает? Да, вы догадались — мини-водопад! Хотя эта мини-версия не так опасна, как водопадный подход на продуктовом уровне, она может доставить немало проблем.

Возьмем для примера задачи 3, 4 и 5. Очевидно, что владельцу продукта будет сложно оценить правильность исполнения требований, пока все три задачи не будут завершены. Просьба к владельцу продукта проверить изменения в схеме базы данных и связанных с ними хранимых процедурах (задача 3) не гарантирует, что мы убедимся в том, что разработка идет в верном направлении, — до тех пор, пока не будет затронут функционал пользователя.

Почему бы вместо этого не применить к уровню задач принцип «вертикальных срезов», также используемый на уровне пользовательских историй? Благодаря ему можно будет начать принимать работу уже через несколько часов. Звучит классно, правда?

Давайте посмотрим, как будет выглядеть способ «вертикальных срезов», пригодный для разбиения историй на самостоятельные задачи.

Пользовательская история

«Как новый пользователь, я бы хотел зарегистрироваться на сайте XYZ, чтобы начать использовать его потрясающие возможности».

Задача 1. Разработать функционал ввода логина и пароля (включая тест-дизайн и автоматизацию тестирования).

Задача 2. Разработать механизм аутентификации по электронной почте (включая тест-дизайн и автоматизацию тестирования).

Задача 3. Разработать посадочную страницу (включая тест-дизайн и автоматизацию тестирования).

Что я здесь сделал? Разбил историю на несколько относительно самостоятельных функций для конечного пользователя. Каждая из них включала небольшую часть работ с базой данных, бизнес-логику и пользовательский интерфейс (см. рис. 4.1). Так что мини-водопад стал безопасной струйкой, а циклы обратной связи теперь можно было измерять в часах, а не в днях!

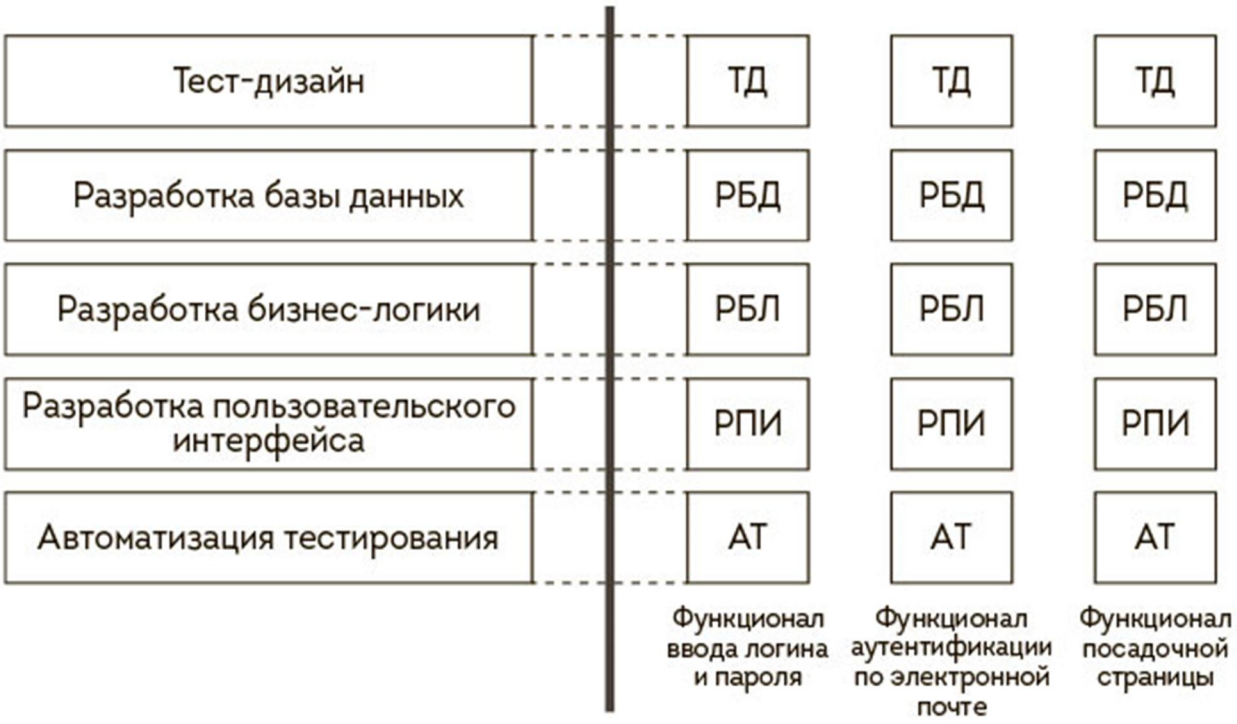

Рис. 4.1. Вместо обособленных слоев каждая задача декомпозирована вертикально, чтобы сократить циклы обратной связи

Уверен, один или двое из вас, дочитав до этого места, подумают: «Минуточку! Если вам удалось разбить историю на небольшие функциональные части, почему бы тогда не выделить эти части в отдельные истории, а не в задачи?» Хороший вопрос! И к счастью, у меня есть ответ на него.

Во-первых, вспомните совет Кона: «Истории должны нести ценность для пользователя». Вернемся к приведенному выше примеру. Хотя регистрация на сайте — это действительно то, что пользователь может оценить, аутентификация по e-mail не обязательно имеет для него ценность.

Во-вторых, важна независимость историй. Если вы можете декомпозировать историю на несколько более мелких и затем независимо их приоритизировать, логично рассматривать их как самостоятельные истории, а не задачи. Возвращаясь к примеру выше, поясню: ни одну из тех задач нельзя приоритизировать отдельно, потому что функциональность, которую сможет оценить пользователь, будет неполной и обрывочной.

Вместо того чтобы есть торт слоями, давайте лакомиться вкусными кусочками — чтобы насладиться всем, что есть в торте, за один укус.

Технические истории и баги

Мне нравится простое и понятное определение технической истории из книги Хенрика Книберга Lean from the Trenches[42]:

Технические истории — это те вещи, которые необходимо сделать, но при этом они неинтересны пользователю (например, обновление баз данных, удаление неиспользуемого кода, рефакторинг запутанной архитектуры, дописывание автотестов для старого функционала).

Когда заходит речь о технических историях, я часто слышу два вопроса:

1. Как мы можем создавать технические истории, когда предполагается, что во главе угла пользовательских историй должен стоять пользователь?
2. Как мы можем использовать стандартный формат пользовательских историй для технических историй?

Мой ответ на первый вопрос: там, где это возможно, вместо создания отдельной технической истории попробуйте включить технические требования в качестве задач одной из функциональных пользовательских историй, к которой эта техническая работа может быть отнесена. Непременно убедитесь, что в случаях, когда техническая работа относится к нескольким функциональным историям, вы учитываете технические задачи только один раз, без дублирования, и только в истории с наивысшим приоритетом.

Мне нравится возможность включать техническую работу в функциональные истории (а не выделять их в отдельные технические истории), ведь так я могу быть уверен, что владелец продукта не проигнорирует ее просто потому, что она неинтересна пользователю. Благодаря включению технической работы в функциональные истории такая работа заведомо не будет проигнорирована, а владелец продукта начнет осознавать техническую сложность, присущую таким историям.

Отвечая на вопрос, как формулировать технические истории, используя стандартный формат пользовательских историй, скажу, что вам не нужно этого делать. Мне нравится формат пользовательских историй для функциональных историй. Однако для технических историй я нахожу его не самым подходящим. Просто используйте любой формат, который несет смысл и легко передает информацию. Убедитесь, что вы выбрали верный формат, максимально подходящий для технических типов историй. Этот же совет применим и для задач по исправлению багов в продуктовом бэклоге. Хотя баги можно фиксировать, используя формат пользовательских историй, я еще раз подчеркну, что с таким подходом можно перемудрить и внести путаницу (представьте, будто вы пытаетесь вбить прямоугольный колышек в круглое отверстие).

КТО ПОСТОЯНЕН, ТОТ ВСЕГДА ПОЧТЕН И СЛАВЕН

В конечном счете формат, который вы выберете, и то, как вы будете декомпозировать ваши истории, зависит только от вас (к слову сказать, некоторые команды даже не доходят до этого уровня). В Scrum нет

упоминаний об этом и, соответственно, нет предписаний, которым вы должны следовать.

Однако, если вам нужно применить только одно правило, пусть им будет единство подхода. Это не значит, что вы связаны с ним навечно. Как верные последователи Scrum, вы, без сомнения, будете постоянно анализировать и дорабатывать ваши процессы. Но, однажды решив дать шанс тому или иному подходу, убедитесь, что команда понимает его и что он применяется для всех пользовательских требований. Постоянство дисциплинирует. Кроме того, это станет вашей страховкой на те (надеюсь, редкие) случаи, когда у владельца продукта не будет возможности продолжать диалог.

ЛАЙФХАК 11. ОПРЕДЕЛЯЕМ КРИТЕРИИ ГОТОВНОСТИ

Я недавно разобрался с проблемой, которую считаю одной из самых сложных в практике Scrum, и эта проблема никак не связана с моей профессией. Я наконец убедил мою жену Кармен, которая совсем не технарь, попробовать Scrum дома! С рождением нашей дочки Эми мы перестали успевать с работой по дому, поэтому я не преминул воспользоваться представившейся возможностью, и теперь наш дом украшает немного эклектичное произведение искусства из стикеров.

Я не мастер на все руки, тем не менее справился с одним из моих домашних scrum-заданий (починить стол) и очень гордился собой. Я перенес свой стикер в колонку «Сделано» прямо на глазах у Кармен — красота! Не моргнув глазом она поспешила вернуть мой стикер обратно в столбец «В работе», сопроводив комментарием: «Починить стол — хорошая работа. Но твоя задача не соответствует моим критериям готовности — твои инструменты все еще лежат на столе». В этот момент я не только очень гордился собой (потому что жена, похоже, слушала мою непрекращающуюся болтовню про Scrum), но и укрепился в важном выводе. Когда взаимодействуют двое или больше людей, самое важное — согласовать ожидания. Scrum учитывает критическую важность этой максимы и предлагает в помощь жизненно важный концепт — критерии готовности.

НЕЧЕТКИЕ КРИТЕРИИ

Хотя дискуссии на холиварные темы могут быть забавными, я обычно нахожу их пустой тратой времени, особенно на рабочем месте. Вы ходите по кругу, результата нет, собеседники разочарованы и раздражены. Не могу сказать, сколько раз в прошлом я становился свидетелем словесных перепалок между программистами и тестировщиками, обсуждающими качество. Программисты с пеной у рта доказывали, что их код прекрасен и готов к вводу в эксплуатацию, в

то время как тестировщики бились головой об стену, утверждая обратное. Так кто же был прав? И те и другие, и никто из них. В чем заключалась проблема? Не было четко определено и/или не было доведено до сведения всей команды, что считать удовлетворительным качеством.

Критерии готовности, разрабатываемые командно, предотвращают подобные споры, случающиеся с завидной регулярностью (см. рис. 4.2). Критерии готовности становятся базовым соглашением, четко определяющим, какие условия должны быть соблюдены, чтобы эту часть работы признать выполненной.

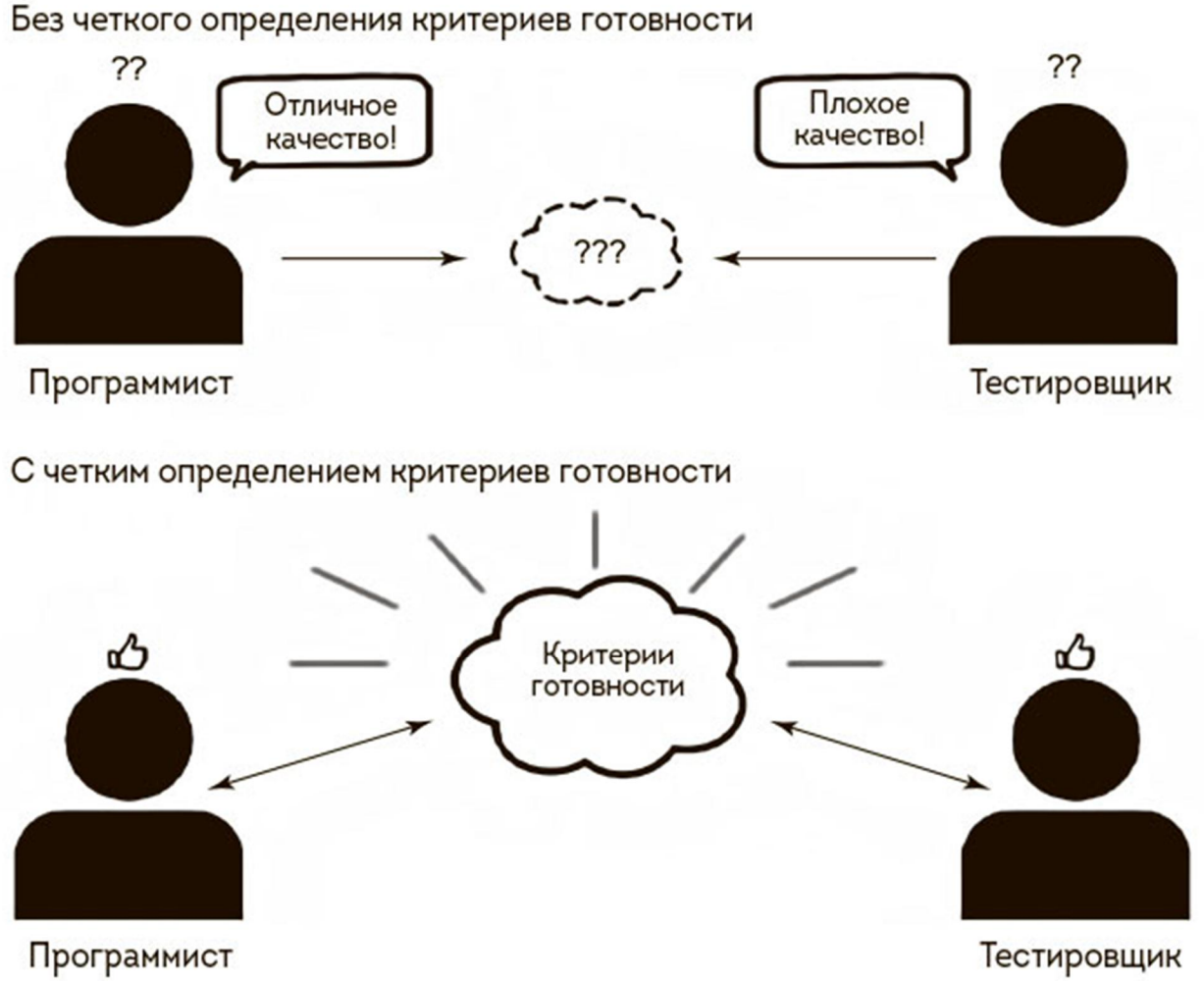

Рис. 4.2. Выравнивание ожиданий с четким определением критериев готовности минимизирует двусмысленность

С ЧЕГО НАЧАТЬ

Самое важное, что нужно осознать, формулируя первые критерии готовности, — они не высечены в камне. Вам не нужно целую вечность обдумывать, какими они должны быть: критерии готовности могут эволюционировать со временем. Как и в остальных случаях, мы должны анализировать и корректировать критерии готовности. Как пишет Клинтон Кит43, «команды расширяют стандартные критерии готовности,

улучшая свои практики. Это позволяет им постоянно повышать свою эффективность».

Советую проявлять гиперреализм или даже консерватизм, формируя первоначальные критерии готовности. Вы не можете погуглить и найти критерии готовности. Ваши должны быть сформулированы под вас — с учетом особенностей продукта, а также возможностей и ожиданий команды (и помните: все они со временем поменяются). Убедитесь, что вся команда, включая разработчиков, владельца продукта и scrum-мастера, вовлечена в формулирование критериев готовности.

РАЗНЫЕ УРОВНИ

Критерии готовности могут применяться на нескольких уровнях. Задачи, пользовательские истории (см. лайфхак 10) и релизы могут иметь собственные критерии готовности (см. рис. 4.3).

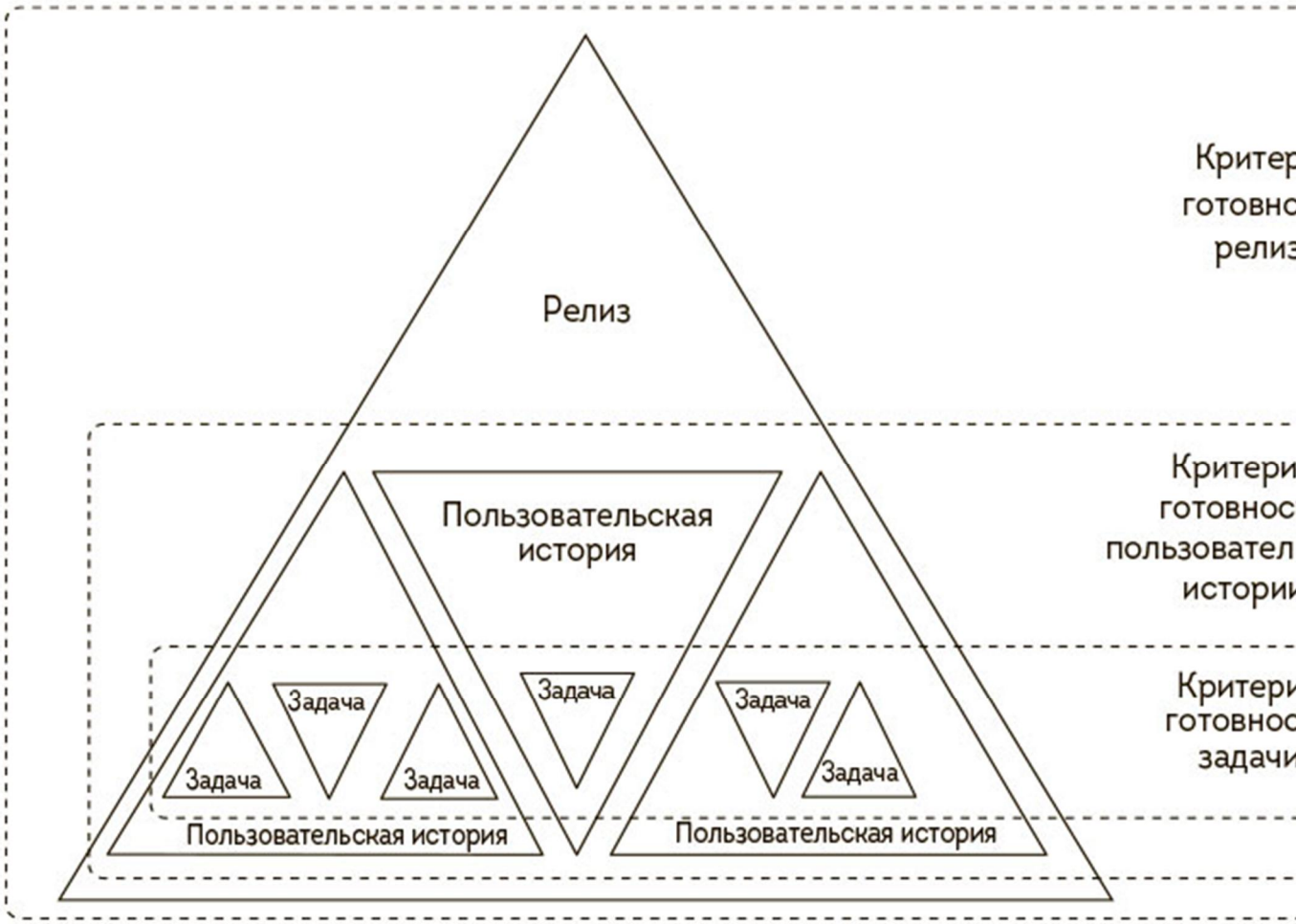

Рис. 4.3. Критерии готовности могут отличаться на разных уровнях

Давайте рассмотрим несколько примеров. Однако помните, что универсальных критериев готовности нет. Вот некоторые ориентировочные критерии, с которыми я работал и которые могут стать полезной подсказкой при обсуждении внутри команды.

Уровень задач (в качестве примера задача на программирование)

Критерии готовности на уровне задач включают следующие элементы:

- код прошел юнит-тесты;
- код прошел ревью кого-то из коллег (если не использовалось постоянное парное программирование), чтобы обеспечить соответствие стандартам написания кода;
- код прошел контроль в системе управления версиями с четкими комментариями, которые можно отслеживать;
- проверенный на входе код не ломает сборку (см. лайфхак 18);
- доска задач обновлена, и время, оставшееся на выполнение задачи, равно 0 (см. лайфхак 21).

Уровень пользовательской истории

В этой истории можно выделить две части.

Первая часть — критерии готовности *описания требований* пользовательской истории. Соблюдение этих критериев позволяет включить истории в спринт.

Вторая и более очевидная часть — критерии готовности, определяющие, когда история готова для релиза.

Критерии готовности описания требований пользовательской истории (см. рис. 4.4):

- пользовательская история была оценена (см. лайфхак 14);
- критерии приемки четко определены;
- пользовательская история была однозначно приоритизирована в продуктовом бэклоге;
- вся необходимая и пригодная к использованию документация (например, схемы и макеты в случае необходимости) доступна;
- исходя из первоначальной оценки становится очевидно, что пользовательская история спокойно поместится в один спринт.

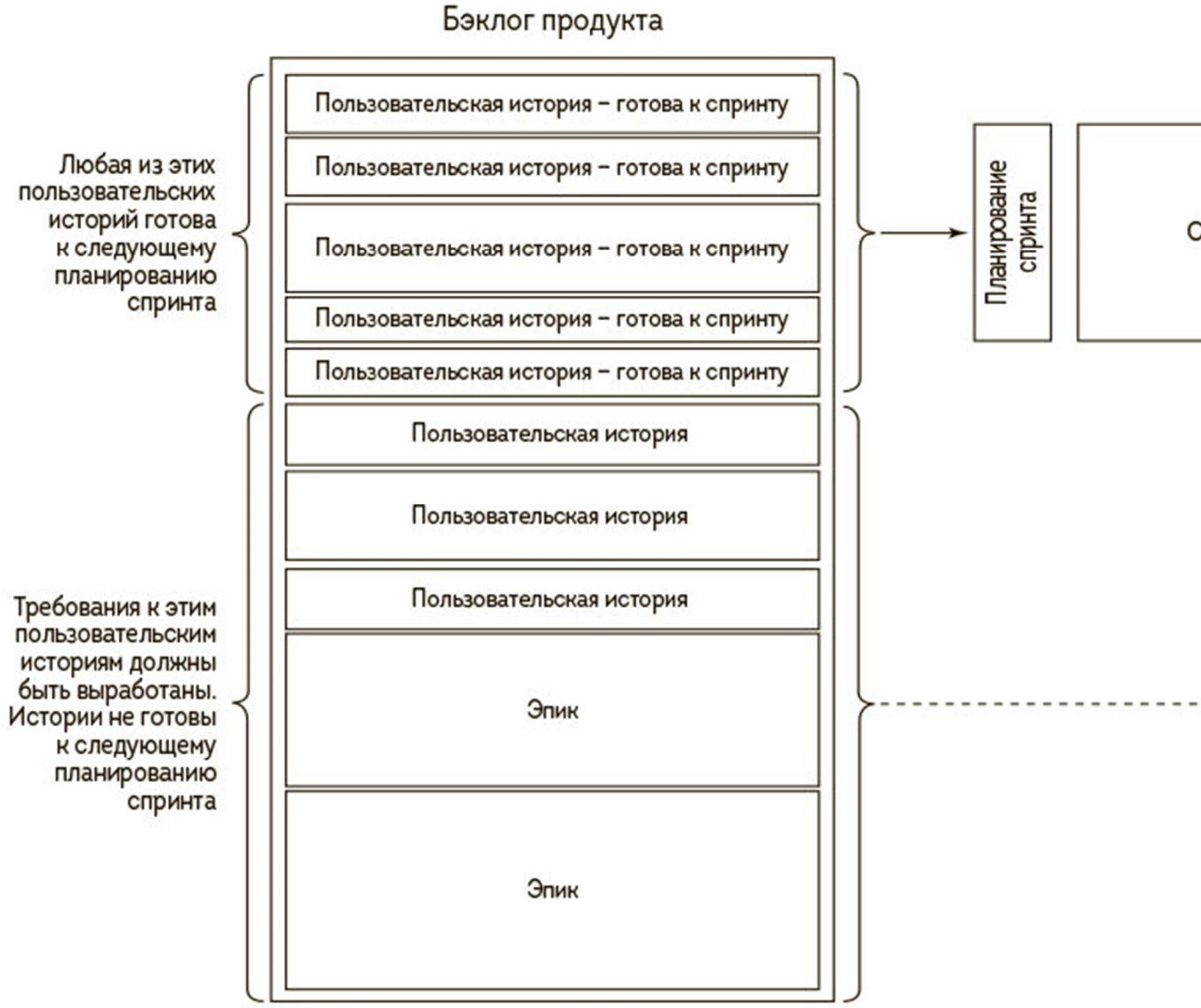

Рис. 4.4. Когда пользовательская история готова, она может быть рассмотрена на планировании

Критерии готовности истории для релиза:

- автоматическое функциональное тестирование подтверждает, что новый функционал работает так, как и ожидалось, от начала и до конца;
- регрессионное тестирование подтвердило успешную интеграцию с другими функциями;
- все соответствующие сценарии сборки / развертывания были модифицированы и протестированы;
- окончательный работающий функционал был проверен и принят владельцем продукта;
- вся соответствующая документация для пользователей написана и проверена;
- если это необходимо, все переводы и другие вещи, связанные с местонахождением, интегрированы и проверены;

- пользовательскую историю команда показала всем стейкхолдерам на обзоре спринта.

Уровень релиза

Критерии готовности на уровне релиза включают следующие элементы:

- весь код, относящийся к релизу, был успешно загружен на «боевые» сервера[44];
- релиз прошел smoke-тесты[45] (как ручные, так и автоматические) на «боевых», а не только на тестовых данных;
- сервисная служба и служба маркетинга потренировались работать с новым функционалом;
- финальный релиз был проверен и принят владельцем продукта.

СИСТЕМНЫЕ ТРЕБОВАНИЯ

Наряду с перечнями, приведенными выше, часто в критериях готовности отражается необходимость соответствия системным требованиям (это также называется нефункциональными требованиями). Они, как правило, заканчиваются на «-ость» и включают такие аспекты, как масштабируемость, межплатформенная совместимость, технологичность, безопасность, расширяемость, совместимость с другими продуктами. Эти требования должны быть включены во все уровни разработки продукта, начиная от задач и заканчивая релизом.

Вот несколько примеров этих требований.

Масштабируемость. Возможность масштабирования до 20 000 пользователей, использующих продукт одновременно.

Полная операционная совместимость. Любая используемая сторонняя технология должна быть кросс-платформенной.

Технологичность. Во всех компонентах должен использоваться простой модульный дизайн.

Безопасность. Необходимо пройти специальные тесты на безопасность и внешнее проникновение.

Расширяемость. Необходимо убедиться, что модуль доступа к данным может подключаться ко всем основным коммерческим реляционным базам данных.

Совместимость. Должна присутствовать возможность синхронизации данных со всеми продуктами в одной группе.

КРИТЕРИИ ПРИЕМКИ ИЛИ КРИТЕРИИ ГОТОВНОСТИ?

Как только ваша команда познакомится с критериями готовности и форматом пользовательских историй, время от времени вы будете

задаваться интересным вопросом: «Должно ли требование XYZ быть частью критериев приемки либо критериев готовности?» Ответ зависит от того, является ли это требование требованием для всех пользовательских историй либо лишь для небольшой подгруппы. Рассмотрим обратную совместимость. Если каждый новый функционал в продукте должен быть полностью совместим с предыдущей версией, тогда подобное нефункциональное требование должно стать частью критериев готовности. С другой стороны, если вы пришли к тому, что обратная совместимость необходима только для части функционала, тогда это требование должно быть добавлено в критерии приемки исключительно у этого функционала.

ТАК ЖЕ ПРОСТО, КАК ГОТОВИТЬ!

Так же как и в лайфхаке 10, при определении способа декомпозиции пользовательских историй на задачи, при определении критериев готовности очень важно проявлять последовательность. Ваши критерии готовности будут эволюционировать по мере изменения требований и способностей команды. Можно начать с детализированного сложного списка критериев, который покажется очень впечатляющим на первый взгляд. Но скоро выполнять их станет невозможно, и кредит доверия и моральный дух команды быстро улетучатся. Поэтому я призываю: будьте реалистичны и оставьте минимально приемлемые критерии готовности, которые каждому по плечу. Помните: они эволюционируют вместе с командой.

Это как добавлять соль во время готовки — вы всегда можете досолить еду, но, если вы пересолили, исправить это будет очень сложно.

ЛАЙФХАК 12. ПРОГРЕССИРУЮЩЕЕ ОТКРОВЕНИЕ

Когда мы стареем, наше тело начинает меняться: еще одна морщинка здесь, еще одна складочка жира там — и так далее и тому подобное. К счастью для тех, кто не хочет стареть красиво, есть мощный инструмент, который мы ежедневно можем использовать для борьбы с этими изменениями, — это чудесный кусочек отражающего стекла, называемый зеркалом! Каждый день вглядываясь в свое отражение, мы можем подмечать малейшие изменения и быстро адаптироваться к ним (если хотим, конечно). Новая морщинка — нет проблем, нанесем побольше крема для лица; намечается излишняя округлость — ок, выложимся в зале сверх обычного, и это поможет решить проблему.

Теперь представьте, что вы целый год не смотрелись в зеркало. Скорее всего, вы столкнетесь с двумя последствиями. Во-первых, без сомнения, вас немного удивит непривычное отражение. Во-вторых, выявленная «деградация», усугублявшаяся весь год, потребует сложных и

трудоемких работ по «исправлению». А может быть, проблема окажется настолько запущенной, что справиться с ней не удастся (вопреки тому, что утверждает индустрия красоты!).

Agile-коуч Майк Дуайер точно подметил в своем блоге: «Scrum не серебряная пуля. Scrum — серебряное зеркало!» Я нахожу эту метафору прекрасной. Действительно, Scrum не серебряная пуля, с помощью которой можно решить проблемы в проектах, — это серебряное зеркало, позволяющее как можно раньше выявлять области, где требуются улучшения. В прошлом, при водопадной модели, разрабатываемый продукт или процессы не анализировались до завершения проекта. Scrum дает командам возможность часто смотреться в зеркало с тем, чтобы обнаружить намечающиеся «морщинки» и своевременно принять меры (пока проблема не усугубилась).

Вы можете подумать, что этот лайфхак посвящен таким важным событиям, как ретроспектива спринта или обзор спринта, но нет. Здесь мы фокусируемся на неформальной активности внутри спринта, которую многие команды еще называют промежуточным контролем. Его цель — анализировать и корректировать работу над пользовательскими историями каждый день в течение всего спринта. «А как же тогда обзор спринта?» — спросите вы. Обзор спринта действительно позволяет регулярно смотреться в зеркало. Однако более частые проверки помогают избежать дополнительных потерь благодаря сокращению циклов обратной связи, а также позволяют поддерживать непрерывную разработку и поставку программного обеспечения (см. лайфхак 18).

Я по-прежнему считаю, что обзоры очень важны, но вижу в них скорее возможность показать и обсудить результат спринта со всеми стейкхолдерами (см. лайфхак 22).

ПРОВЕРКА И ВАЛИДАЦИЯ

Цель промежуточного контроля — убедиться, что вся работа, которая была выполнена (или будет выполнена), соответствует ожиданиям команды. Промежуточный контроль может быть проведен по инициативе разработчика, который хочет получить от владельца продукта подтверждение, что задача понята правильно. Или же, наоборот, владелец продукта хочет удостовериться в правильности проектных решений, пока на их реализацию затрачено еще не слишком много времени и сил.

Как часто в проектах, работающих по водопадной модели, возникала ситуация, когда менеджер по продукту (если использовать старую терминологию), тараща глаза, испуганно восклицал в конце разработки: «Это не то, что я имел в виду! Я думал, это будет работать вот так… бла-бла-бла…» На что разработчики гневно отвечали: «Тогда почему на странице 47 версии 3.6.4 спецификации, которую подписала половина компании, четко и ясно утверждается прямо противоположное?!»

Я не говорю, что о проблемах с коммуникацией можно будет позабыть благодаря Scrum. На самом деле чем чаще вы будете открыто обсуждать проект, тем больше разногласий обнаружится. Однако чем чаще вы взаимодействуете друг с другом, тем проще вам будет сглаживать любые разногласия и двигаться вместе в одном направлении.

КТО ГДЕ КОГДА

Промежуточный контроль можно проводить тогда, когда это требуется. Ряд команд, с которыми я работал, предпочитали запускать его дважды в течение дня (как правило, на час сразу после ежедневного scrum и на час во второй половине дня). Это не значит, что все участники команды должны приходить на встречи по промежуточному контролю на 2 часа в день. Но они знают, что именно в этот период их могут отвлечь. Вот почему им не стоит печалиться, если в это время их похлопают по плечу и попросят помочь с промежуточным контролем.

Поскольку промежуточный контроль обычно представляет собой практическую демонстрацию работы, требующей проверки и валидации, обычно его проводят за столом того разработчика, чью работу и проверяют. Нет смысла тратить время и силы на формальности и отправлять всем приглашения на встречу или бронировать переговорную. Однако, если вы лишены роскоши сидеть вместе всей командой, конечно, придется совершить несколько дополнительных действий для организации встречи.

На встречи по промежуточному контролю я рекомендую всегда звать владельца продукта, соответствующего разработчика (разработчиков) и тестировщика (тестировщиков). Уделяя больше внимания живому общению, а не спецификации, важно убедиться, что все находятся на одной волне и хорошо понимают, о чем идет речь.

ПРОБЛЕМЫ И КОРРЕКТИРОВКИ В РАБОТЕ

Встречи по промежуточному контролю должны быть посвящены элементам бэклога текущего спринта, а не обсуждению будущих историй — это оставьте для планирования спринта (см. лайфхак 8). Результаты подобных встреч, как правило, касаются следующих категорий:

- **Проблемы.** Это проблемы, возникшие в процессе разработки: неработающий функционал, который нужно исправить (см. лайфхак 9).
- **Корректировки.** Это небольшие изменения в конструктивных решениях. Их не следует путать с крупными изменениями, раздувающими пользовательскую историю.

- **Одобрение и улыбки.** Вот что вы получите, если промежуточный контроль подтвердит, что работа идет в правильном направлении!

ОСТЕРЕГАЙТЕСЬ РАЗДУВАНИЯ СКОУПА РАБОТ В СПРИНТЕ

Что произойдет, если владелец продукта после того, как «попробует блюдо на вкус» во время промежуточного контроля, решит добавить специй (и я не имею в виду щепотку соли)? Эта ситуация возникает часто и каждый раз демотивирует. Однако она не станет проблемой, если правильно ею управлять.

Давайте вспомним лайфхак 1: выбрав скоуп работ на спринт, его нужно «оставить в покое» и защищать от изменений. Это даст разработчикам уверенность, что фокус их работы останется неизменным на протяжении спринта. Хотя следует ожидать небольших корректировок и вносить их в спринт, необходимо избегать значительного изменения требований пользовательских историй в середине спринта (см. рис. 4.5). Что же делать, если это произошло? Предположим, владелец продукта решил, что у корзины товаров должно быть больше способов оплаты, чем это было согласовано ранее. Без проблем. Просто создайте еще одну пользовательскую историю, которая будет посвящена исключительно новым способам оплаты, и добавьте в продуктовый бэклог. Если этой истории присвоят очень высокий приоритет, рассмотрите ее на планировании и возьмите в следующий спринт, но не изменяйте скоуп работ в этом.

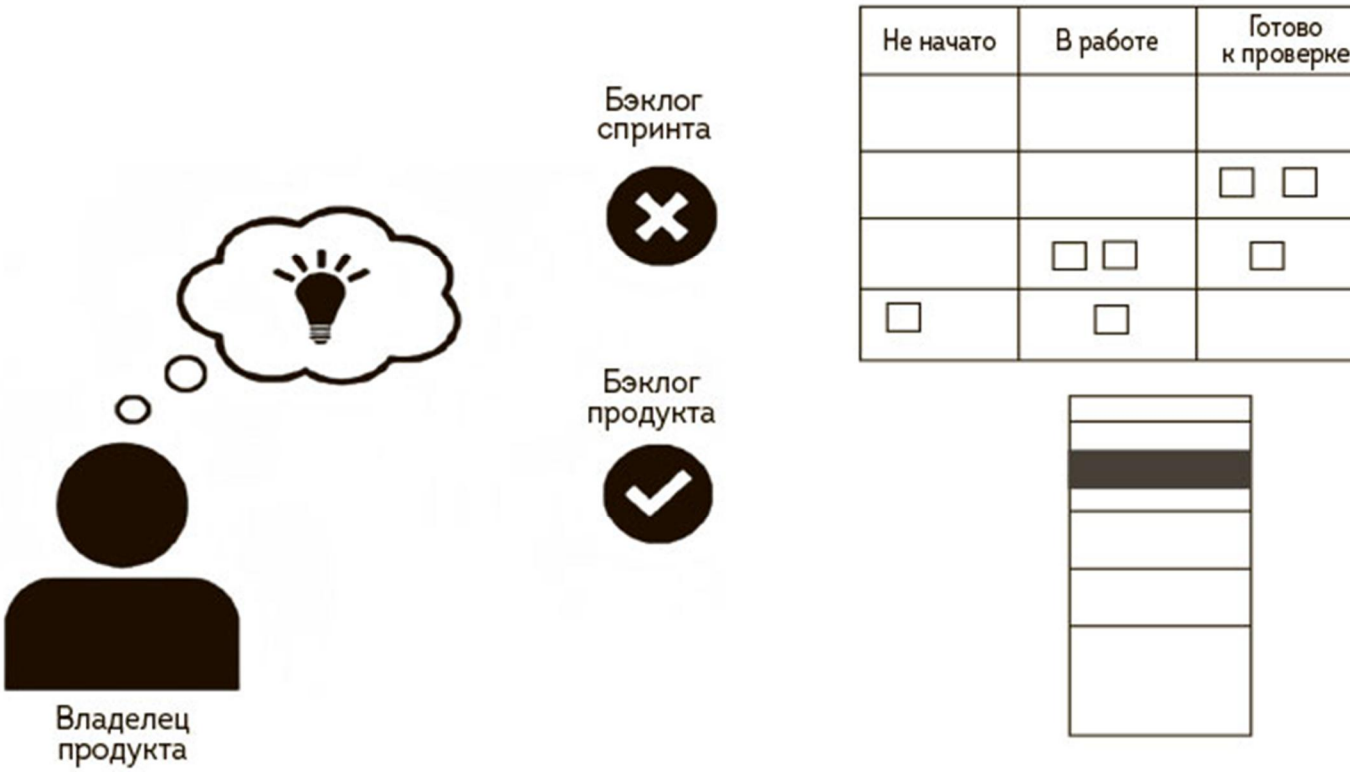

Рис. 4.5. Принимайте небольшие изменения в текущем спринте, а существенные оставляйте для бэклога продукта и будущих спринтов

ЗАФИКСИРУЙТЕ РЕЗУЛЬТАТ

Хотя команды призывают решать проблемы и вносить корректировки прямо на этапе промежуточного контроля (если изменение относительно простое), не всегда это возможно. В этом случае важно, чтобы о выявленной проблеме не забыли, когда придет время привести все дела в порядок. Рекомендую не тратить слишком много времени на документирование. Вместо этого:

1. Добавьте пунктирную линию под уже имеющимся списком критериев приемки пользовательской истории, чтобы отделить новые требования от исходных.
2. Добавьте список участников встречи по промежуточному контролю, а также ее дату и время, чтобы команде было проще вспомнить, когда именно обсуждались изменения (если вдруг итоги встречи подзабудутся).
3. Формулируйте требования короткими четкими тезисами (см. рис. 4.6).
4. Новые изменения могут породить новые задачи, увеличить длительность уже существующих или и то и другое вместе. Это нормально, только убедитесь, что время, оставшееся на текущие задачи, соответствующим образом скорректировано. Я также советую добавлять новые задачи на доску с помощью стикеров других цветов — отличных от тех, что использовались для исходных задач (см. лайфхак 21).

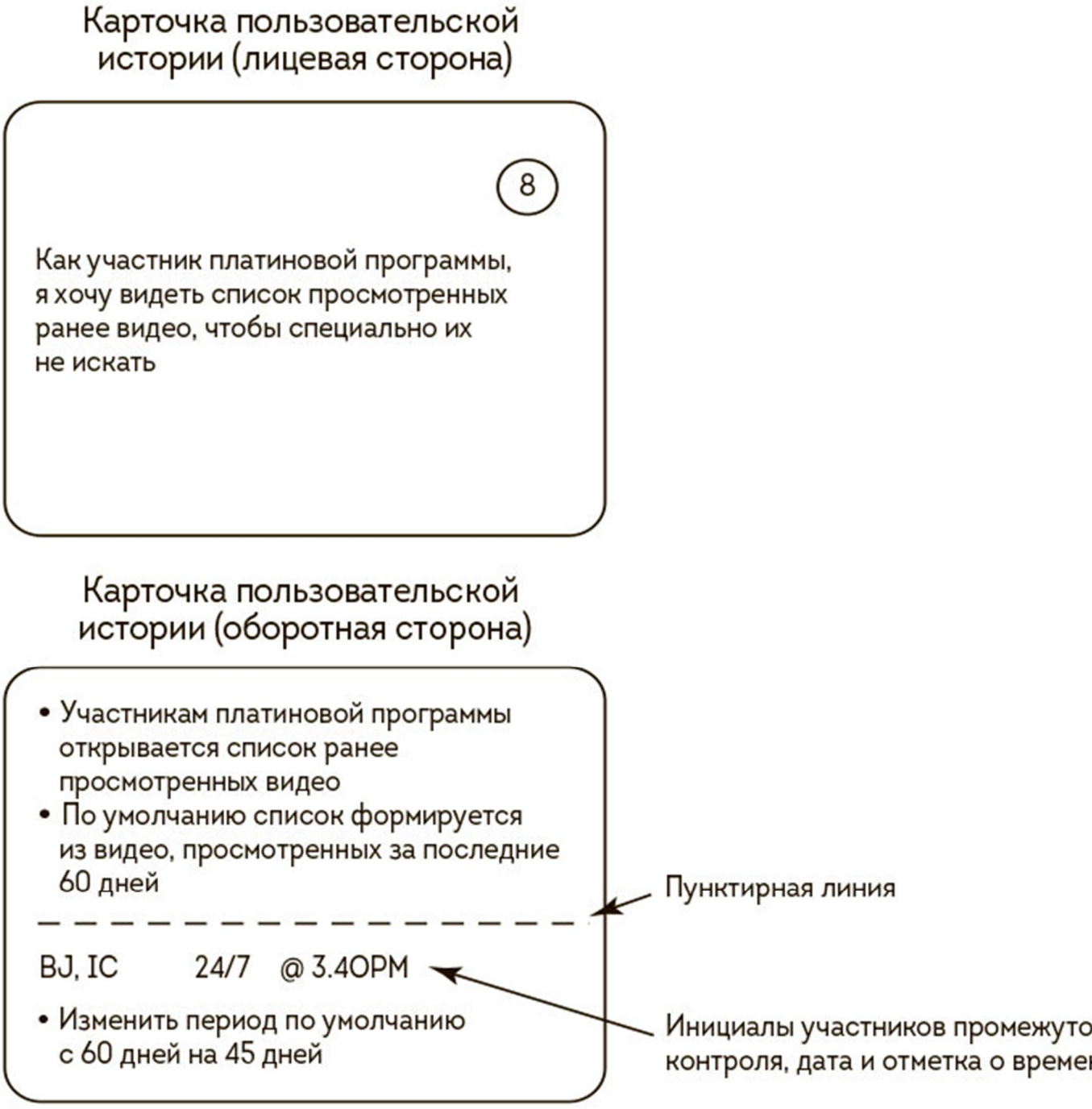

Рис. 4.6. Пример того, как небольшие изменения можно отразить на обороте карточки пользовательской истории

НЕ ПЕРЕБОРЩИТЕ

В Брисбене, где я вырос, в центре города есть несколько открытых склонов для скалолазания, где жители могут побыть на природе. Хотя скалолазание не спорт моей мечты, я периодически пускался в эту авантюру, чтобы полюбоваться бесстрашием и гибкостью «людей-пауков». Я вспоминаю самых отважных (точнее, сумасшедших) спортсменов, которые занимались свободным лазанием (free climbing) — поднимались по скале только с помощью силы рук и ног, при этом не используя верхнюю страховку, а лишь скальные крючья. Чем более

редкими и разнесенными были зацепы, тем больше был риск соскользнуть и дольше времени требовалось на то, чтобы вернуться на эту точку. Однако альпинисты, которые как одержимые слишком часто закрепляли крючья, неизбежно теряли энергию и время, а также ритм и фокусировку, необходимые, чтобы достичь вершины.

Встречи по промежуточному контролю внутри спринта похожи на закрепление якорей при свободном лазании. Их нужно проводить настолько часто, насколько это необходимо, чтобы обеспечить безопасность. Но они не должны превращаться во встречи ради встреч — вы же не хотите понапрасну тратить драгоценное время и энергию команды.

ЗАКЛЮЧЕНИЕ

Три лайфхака этой главы посвящены тактике, инструментам и подсказкам, которые помогут вашей команде определить и развить свои требования и критерии готовности. Давайте вспомним, какие вопросы мы рассмотрели.

Лайфхак 10. Структурируем пользовательские истории

- обзор иерархии пользовательских историй;
- подходы к разбиению готовых ко взятию в спринт пользовательских историй на задачи;
- варианты включения технических требований в бэклог спринта.

Лайфхак 11. Определяем критерии готовности

- отправные точки для определения «готовности»;
- способы формирования нескольких уровней критериев готовности;
- различия между критериями готовности и критериями приемки.

Лайфхак 12. Прогрессирующее откровение

- польза от последовательных встреч по промежуточному контролю внутри спринта;
- организация промежуточного контроля — кто где когда;
- отличия между раздуванием спринта и приемлемыми изменениями в середине спринта.

Глава 5

НАЛАЖИВАЕМ ПРОЦЕСС ОЦЕНКИ

Нравится вам или нет, необходимость давать оценки в проектах по разработке ПО никуда не исчезнет. К счастью, это тяжкое бремя, которое приводит в отчаяние большинство команд, можно облегчить, если они решат принять фактический стандарт для оценки проектов в Scrum — использовать *относительные оценки.*

Три следующих лайфхака познакомят вас с теорией относительных оценок и послужат руководством по переходу от оценок по времени к относительным.

Лайфхак 13 «Относительные оценки» познакомит вас с удивительно простым способом выставлять оценки в относительных единицах.

Лайфхак 14 «Покер планирования в темпе» предложит целый набор подсказок и советов для того, чтобы провести эффективную сессию покера планирования.

Наконец, лайфхак 15 «Переходим к относительным оценкам» даст несколько советов, как помочь команде перейти от оценок по времени к относительным оценкам.

ЛАЙФХАК 13. ОТНОСИТЕЛЬНЫЕ ОЦЕНКИ

И многие из тех, кто читает эту главу, и я бесконечно долго наблюдали, как время утекает, словно песок сквозь пальцы, на затянувшихся встречах по оценке — в попытках скрупулезно декомпозировать расплывчатые требования до детально описанных задач (на очень длинных и очень полосатых диаграммах Ганта). Но еще больше времени мы тратили на практически ежедневное обновление этих диаграмм — из-за неизбежных изменений в скоупе работ, не говоря уже о корректировке самих оценок.

Незадолго до того я понял, что единственная польза от всего этого — интересные полосатые обои для офиса!

Так начались мои героические поиски. Я стремился найти эффективный способ овладеть темным искусством оценок. После долгих поисков я остановился на подходе, который считаю наиболее результативным для оценки эмерджентных требований, — с помощью относительных единиц. Элегантная простота этого подхода окончательно убедила меня: вот он, свет в конце длинного и темного тоннеля оценок.

БОЛЬ ОЦЕНКИ

Прежде чем перейти ко всем тонкостям относительной оценки, давайте вернемся к основам и разберемся, почему оценка настолько трудна и болезненна (особенно в нашем мире разработки программного обеспечения).

Во-первых, мы, люди, изначально не являемся талантливыми оценщиками. Мы склонны быть оптимистами или пессимистами, и очень редко реалистами. Мне даже не нужно подкреплять это утверждение статистическими данными, потому что я уверен: любой читающий этот абзац с ним согласится!

Кроме того, в мире программного обеспечения действует множество неизвестных: технологии и требования непрерывно меняются, многие вводные весьма непостоянны, задачи находятся в сложных взаимосвязях (столь же непростые закономерности возникают и между людьми). И это даже без учета внешних факторов!

ЗАЧЕМ БЕСПОКОИТЬСЯ ОБ ОЦЕНКЕ?

Если наши оценки могут быть настолько неточными, зачем вообще оценивать? Я считаю, что, даже если наши оценки не всегда верны, все равно очень важно проводить оценку. И сейчас я расскажу, почему это нужно делать.

Первая причина — нужны взвешенные решения. Предположим, я спросил пару из Сан-Франциско, где они предпочтут провести отпуск — в Австралии или в Мексике. Конечно, им может нравиться одно из этих мест, но тут в игру вступают два других важных фактора — время и бюджет. Допустим, они выберут поездку в Австралию (да, я немного пристрастен), однако им может не хватить отпуска (времени) для долгого путешествия либо денег (австралийский доллар — очень сильная валюта). Как же им понять, могут ли они позволить себе такую поездку? Что ж, они просто оценят, сколько времени она займет и сколько будет стоить. Тот же принцип применяется и к требованиям, входящим в вишлисты наших программных продуктов.

Вторая причина — ставить цели. Если вы в чем-то похожи на меня, то, установив для себя дедлайн, вы делаете все возможное, чтобы в него уложиться. Конечно, время от времени ваши оценки будут неточны — и в этом случае вы не должны совершать непосильные подвиги, — однако сам процесс оценки и постановки целей поможет вам оставаться сосредоточенным и максимизировать результаты.

Теперь, когда вы убедились, что оценка — это полезное упражнение, начнем разбираться, что же такое относительная оценка.

ЧТО ТАКОЕ ОТНОСИТЕЛЬНАЯ ОЦЕНКА?

Относительная оценка применяется скорее на уровне бэклога продукта, чем на уровне бэклога спринта. Элементы бэклога спринта могут оцениваться в традиционных единицах времени (таких как часы) главным образом потому, что оценку мы проводим для одного спринта (то есть речь идет о днях, а не о месяцах) и требования детализируем достаточно подробно. С другой стороны, элементы бэклога продукта

описаны более расплывчато и могут в совокупности занимать не один месяц работы, что делает оценку, основанную на времени, очень сложной, если не невозможной.

Относительная оценка основывается на принципе, согласно которому *сравнение* гораздо быстрее и точнее, чем *декомпозиция.* Вместо того чтобы декомпозировать требование на составные задачи (и давать оценку уже им), команды сравнивают относительные усилия по реализации нового требования с относительными усилиями по реализации ранее оцененного требования. Приведу аналогию, чтобы пояснить, что я имею в виду.

Взбираемся по лестнице

Предположим, перед нами четыре здания. Три из них — современные, а четвертое — старое и ветхое. Все они разных размеров. Нас попросили оценить, за сколько мы поднимемся по лестнице на верхний этаж каждого из зданий (см. рис. 5.1).

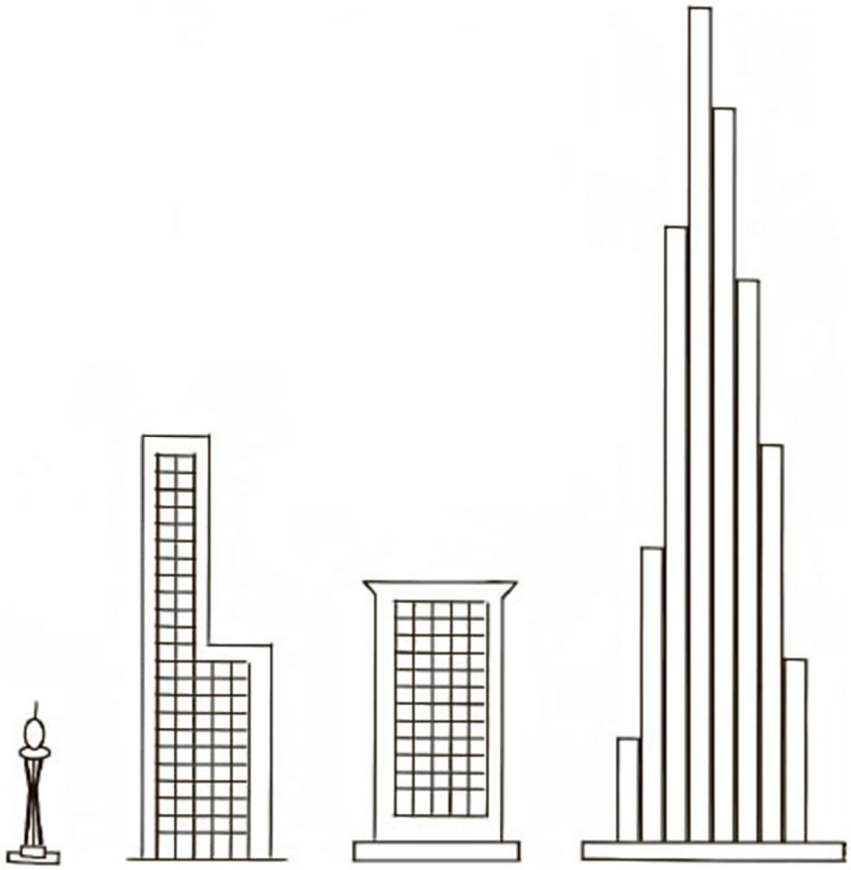

Рис. 5.1. Как оценить, сколько мы будем подниматься по ступенькам?

Если мы никогда не выполняли такую задачу, то столкнемся с несколькими неизвестными. Так, мы не уверены в нашей физической подготовке или не знаем, какие препятствия нужно будет преодолеть на лестничных клетках.

Так что же нам делать? Мы можем подсчитать этажи в каждом здании, а затем оценить, сколько времени нам понадобится, чтобы преодолеть

уже известное число лестничных пролетов, хоть мы и не знаем наш уровень физической подготовки или состояние лестничных клеток. Эта оценка не только отнимет много времени, но и будет крайне неточной, если наши допущения окажутся ошибочными.

Давайте рассмотрим другой вариант. Сперва классифицируем здания по «шкале усилий» и самому маленькому присвоим 10 пунктов — поинтов. Цифру 10 мы выбрали произвольно — могли поставить 100, 1000 или взять любое другое число (вы скоро поймете, почему это не имеет значения). Теперь посмотрим на здание 2. Оно примерно в три раза больше первого, поэтому присвоим ему 30 поинтов. Третье здание (более старое) находится где-то посередине, поэтому мы могли бы присвоить ему 20 поинтов, однако из-за его ветхости мы можем столкнуться с препятствиями при подъеме. Учитывая эти факторы, присвоим ему 25 поинтов. Последнее крупное здание примерно в два раза больше второго (30 поинтов), поэтому оценим его в 60 поинтов (см. рис. 5.2).

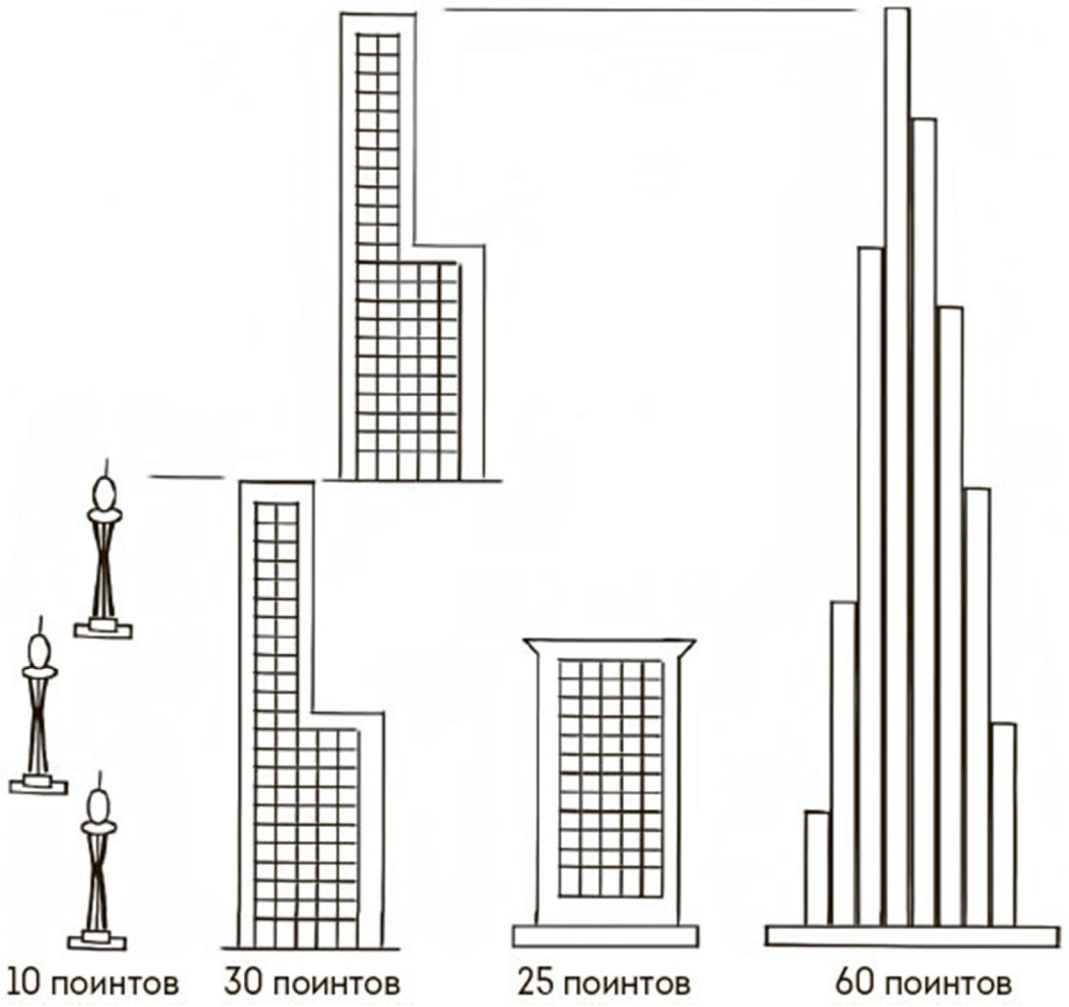

Рис. 5.2. Мы можем оценить эти здания в относительных единицах, быстро выполнив несколько сравнений

Заметьте, эти пункты (или поинты) — просто относительные калибровочные отметки, помогающие нам сравнивать здания. Цифры

не являются единицами измерения длины или времени — это просто калибровочные метки.

Это небольшое упражнение позволяет быстро оценить усилия, необходимые для наших четырех подъемов, но не в абсолютном выражении, а в относительном. Но эта информация — лишь первый фрагмент головоломки. Мы получили представление об относительных усилиях по восхождению на одно здание по сравнению с другим. Однако нам все еще нужно оценить общую продолжительность подъема на все четыре здания.

Что дальше? Может быть, потратить немного времени и проверить свою спортивную форму, а заодно и состояние лестниц? Давайте ограничим этот эксперимент 10 минутами (пусть это будет продолжительностью нашего спринта) и посмотрим, как далеко нам удастся продвинуться (см. рис. 5.3).

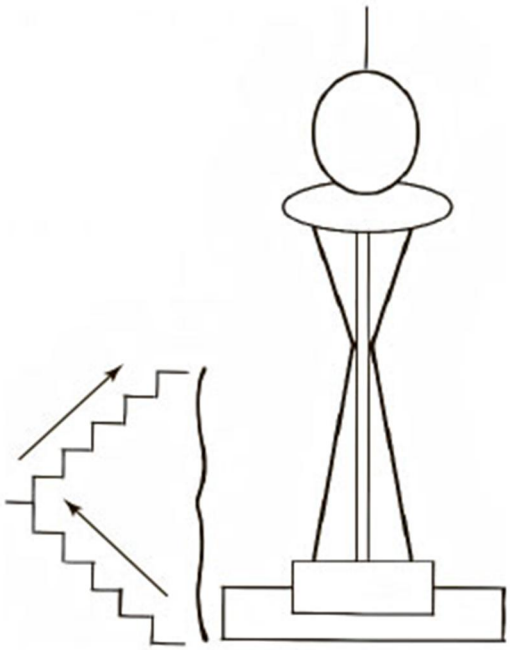

Рис. 5.3. После 10 минут подъема по лестнице мы прошли половину пути в «10-поинтовом» здании

Идем к лестничной клетке и через 10 минут оказываемся на полпути в здании 1 (оцененном в 10 поинтов). Отсюда можно сделать вывод о скорости нашего подъема, или, другими словами, об объеме работы (в поинтах), который мы можем выполнить в течение нашего 10-минутного спринта. Мы прошли половину лестницы в первом здании и можем сказать, что наша скорость составляет 5 поинтов за спринт, или, более коротко, просто 5 поинтов.

«Но нам нужно знать, сколько времени займет подъем на все четыре здания», — скажете вы. Как насчет простой экстраполяции? Давайте начнем с суммирования всего количества работы, складывая относительные оценки зданий: 10 + 30 + 25 + 60 = 125 поинтов.

Теперь берем нашу скорость (помните, она составляет 5 поинтов) и делим суммарные 125 поинтов на 5 поинтов в спринт, получаем 25 спринтов. Мы знаем, что каждый спринт занимает 10 минут, значит, нам потребуется 250 минут. После этого мы можем добавить 50 минут (20% от расчетного времени) в резерв, чтобы отдышаться и спуститься вниз на лифте. И вуаля — мы можем дать приблизительную оценку: 300 минут или 5 часов, чтобы выполнить это упражнение.

ОТНОСИТЕЛЬНАЯ ОЦЕНКА В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ

Давайте применим этот подход к нашим проектам по разработке программного обеспечения. Вместо того чтобы оценивать наше умение взбираться по лестницам, мы оценим усилия, которые потребуются для реализации элементов бэклога продукта.

Вначале определим усилия, необходимые для реализации элемента бэклога продукта, через три параметра: сложность, повторы и риски (см. рис. 5.4).

Рис. 5.4. Три фактора, которые определяют, сколько усилий потребуется приложить для завершения элемента бэклога продукта

Позвольте мне объяснить разницу: у нас может быть элемент бэклога, который требует разработки *сложного* алгоритма оптимизации. Это не обязательно потребует написания множества строк кода, однако на размышления и анализ уйдет немало времени.

Далее, у нас может быть элемент бэклога, который ориентирован на пользовательский интерфейс и требует тонкой настройки HTML для нескольких типов и версий браузеров. Хотя сама по себе эта работа несложная, она требует большого количества *повторов*, а с ними испытаний и проверок на ошибки.

Другой элемент бэклога продукта может потребовать взаимодействия с новым сторонним продуктом, с которым мы раньше не работали. Это требование сопряжено с риском, а на преодоление проблем при знакомстве с продуктом может уйти значительное время.

При оценке элементов бэклога необходимо принять во внимание все эти факторы.

Еще один момент, который стоит отметить: нам не обязательно детализировать техническое задание, чтобы оценить усилия. Если владелец продукта хочет создать экран входа в систему, для его оценки не нужно продумывать точную механику, алгоритм работы или иметь макеты экрана и т. д. Это потребуется позже, когда мы начнем реализовывать требование в спринте. Все, что нужно знать на раннем этапе, — как много усилий потребуется для реализации функции входа в систему относительно, например, системы поиска, которую мы оценивали до этого. Если функции поиска мы присвоили 20 поинтов, то функции входа в систему следует назначить 5 поинтов — при условии, что для ее реализации потребуется четверть усилий.

СКОРОСТЬ

Мы уже обсуждали основное назначение метрики скорости. Однако вот еще несколько важных особенностей, которые надо иметь в виду:

- Скорость рассчитывается как сумма поинтов всех элементов бэклога, завершенных в спринте.
- Наиболее распространенный подход к работе с частично завершенными элементами бэклога — включать поинты по ним только в тот спринт, в котором элемент бэклога достиг критериев готовности (см. лайфхак 11).
- Безусловно, скорость можно определить по одному спринту, однако это значение не обязательно будет совпадать с долгосрочным средним показателем, ведь от спринта к спринту скорость меняется. Колебания могут быть вызваны самыми разными причинами, включая влияние частично завершенных историй, препятствия, загруженность участников команды — и это далеко не все. Использование средней скорости или скользящей средней скорости последних трех спринтов — самый простой вариант расчета показательного значения. Для более верного расчета скорости я рекомендую использовать бесплатный калькулятор диапазона скорости Кона46. Учтите, для этого вам понадобятся данные как минимум за пять спринтов.
- Скорость рассчитывается при одном и том же составе команды и одной и той же длине спринта. В противном случае рассчитать ее будет гораздо сложнее.

ОТНОСИТЕЛЬНАЯ ОЦЕНКА НА ПРАКТИКЕ

Для внедрения относительной оценки на практике многие команды играют в отличную игру под названием «Покер планирования». Лайфхак 14 объясняет механику этой эффективной

техники и предлагает набор советов и приемов, чтобы сделать ее настолько эффективной и действенной, насколько это возможно.

Оценка — дело сложное. Очень сложное. Разработка программного обеспечения связана с высоким уровнем сложности (и многими неизвестными). Кроме того, она требует совершенства и отсутствия дефектов, чтобы программное обеспечение исправно работало. Из-за всех этих факторов ни один подход к оценке не будет абсолютно надежным. Однако я верю, что относительная оценка в стори-поинтах так же точна, как и любые другие альтернативные варианты оценки, но по сравнению с ними она гораздо более проста и изящна.

ЛАЙФХАК 14. ПОКЕР ПЛАНИРОВАНИЯ В ТЕМПЕ

Пополнив свой арсенал относительными оценками в виде стори-поинтов, теперь вы можете противостоять силам, делающим оценку в программном обеспечении такой болезненной. Относительная оценка проста, логична и гораздо интереснее, чем другие способы оценки — скучные, затянутые и вводящие в заблуждение.

Техника, которая позволяет применять относительные оценки, — это игра, которую изобрел Джеймс Греннинг и популяризировал Майк Кон. Она называется «покер планирования» и основывается на методе, который был разработан в 1970-х годах, — широкополосном дельфийском методе (Wideband Delphi) (который, в свою очередь, базировался на более ранней версии дельфийского метода (Delphi), изобретенной в RAND-Corporation в 1950-х годах). Этот подход широко практиковали кросс-функциональные команды экспертов, чтобы получать оценки гораздо более точные (по сравнению с теми, что делал один человек).

НАЧИНАЕМ ИГРУ

Этот способ называется «покером планирования», потому что команды буквально играют в карты. Но вместо пик, червей, треф и бубен они используют карты, представляющие значения стори-поинтов. Общепринятая система значений поинтов — это модифицированная Майком Коном последовательность чисел Фибоначчи.

½, 1, 2, 3, 5, 8, 13, 20, 40, 100, ∞ (бесконечность)

Предвосхищая ваше удивление, скажу, что считаю справедливой интерпретацией карты «бесконечность» что-то вроде «Вау! Это слишком большая задача, чтобы ее оценить. Определенно ее необходимо разбить на несколько, прежде чем давать оценку».

Я поклонник использования модифицированной последовательности Фибоначчи, потому что она помогает отразить возрастающую

неопределенность по мере того, как требования становятся все больше и больше (см. рис. 5.5), и одновременно не попасть в ловушку избыточной точности (с этим связаны изменения от 21 до 20, с 42 до 40 и так далее).

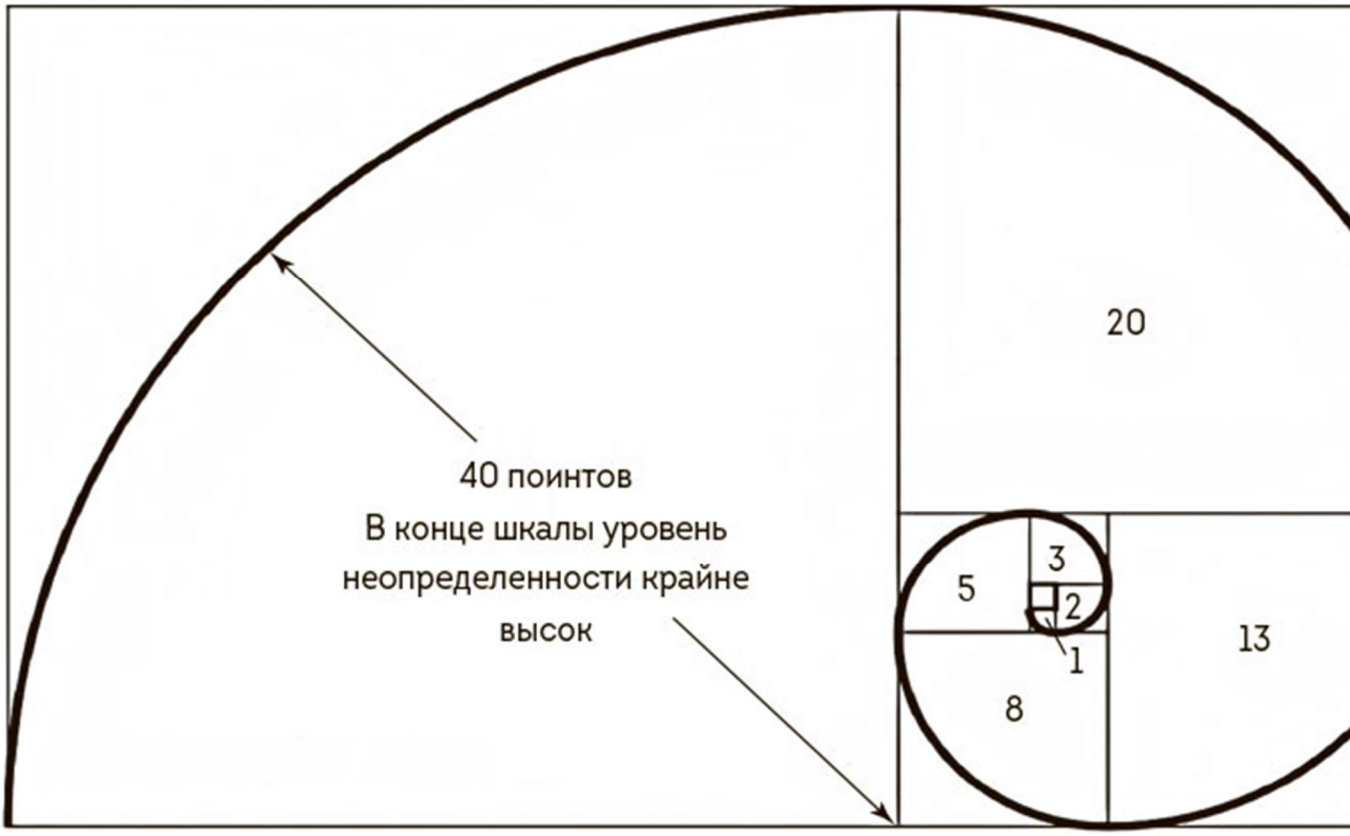

Рис. 5.5. Модифицированная последовательность чисел Фибоначчи — это аппроксимация логарифмической «золотой спирали», где неопределенность увеличивается по мере увеличения требований

И все же есть одна потенциальная проблема, особенно при работе с новыми командами. Значения поинтов не должны коррелировать с определенными единицами времени или интервалами. Вот почему при использовании чисел Фибоначчи возникает проблема: люди могут приобрести плохую привычку приравнивать 13 поинтов, например, к 13 часам.

Чтобы не допустить этого, некоторые команды применяют более абстрактные системы оценки — такие как размеры футболок:

XS, S, M, L, XL, XXL

Лично я не использую этот дополнительный уровень абстракции, потому что он требует еще одного шага — перевода в числовое значение при прогнозировании дат релизов. Помните, как в лайфхаке 13 мы рассчитали время, которое потребуется для подъема на здания, разделив на числовое значение скорости?

МЕХАНИКА ПОКЕРА ПЛАНИРОВАНИЯ

Прежде чем приступить к подсказкам и советам, чтобы ваши встречи не превратились в марафон до поздней ночи, давайте проведем краткий обзор механики игры.

Встреча по покеру планирования проходит следующим образом:

1. Владелец продукта разъясняет верхний элемент бэклога продукта, а затем приглашает команду задавать вопросы, чтобы уточнить объем работ и желаемые результаты. Любые изменения в описании или критериях приемки элемента бэклога продукта фиксируются по мере уточнения.
2. Каждый участник команды кладет лицом вниз карту, которая, по его мнению, лучше всего отражает объем усилий, необходимый для реализации элемента бэклога продукта.
3. После того как карты окажутся на столе, все участники команды одновременно открывают их.
4. Если единодушия в оценках нет, то владелец старшей карты быстро (не более нескольких минут) обсуждает элемент бэклога с владельцем младшей карты, при этом все участники команды присутствуют при этом диалоге.
5. С новыми данными, которые выяснятся в процессе дискуссии, команда возвращается к шагу 2.
6. Шаги от 2 до 5 повторяются до тех пор, пока в команде не будет единодушия по рассматриваемому элементу бэклога продукта.
7. После того как рассматриваемому элементу бэклога присвоена определенная оценка, процесс, начиная с шага 1, запускается для следующего элемента бэклога продукта (см. рис. 5.6).

Рис. 5.6. Порядок типичного цикла покера планирования

Обратите внимание: scrum-мастер задействован во встрече как фасилитатор, но не принимает участия непосредственно в оценке.

Принципиальное: каждый участник команды должен оценить усилия для всего элемента бэклога, а не только ту часть, которая относится к его специализации. Например, программист должен оценить усилия не только по написанию кода, но и по тестированию и развертыванию — интересный подход, не так ли?

Итак, как же каждый участвует в оценке работы, которая не входит в сферу его компетенций? Все участники команды должны давать совокупную оценку, основываясь на опыте. Несмотря на их неучастие в тестировании, они вспомнят, что происходило, когда велась работа над аналогичным элементом бэклога в прошлом. Так, хотя программирование не было слишком сложным, тестирование стало настоящим кошмаром из-за множественных интеграций со сторонней платежной системой. Обычно люди предполагают, что им предстоит оценить только свою конкретную роль, и из-за этого новые команды

обычно начинают с очень разрозненных оценок (остерегайтесь этой ловушки).

КОГДА ИГРАТЬ В ПОКЕР ПЛАНИРОВАНИЯ

Первая сессия покера планирования должна состояться после формирования бэклога продукта. Последующие игры могут проводиться всякий раз, когда новый элемент добавляется в бэклог продукта, или в редкой ситуации, когда требуется переоценка. Переоценка требуется, когда целый класс элементов бэклога внезапно становится меньше или больше (в относительной степени). Когда такое происходит? Предположим, что ряд элементов бэклога зависит от интеграции с третьей стороной. API третьей стороны было ненадежным, и при работе с ним нужно было использовать обходные пути, не говоря уже о большом объеме дополнительного тестирования. Допустим, третья сторона наконец-то выпустила новый, суперпродвинутый интерфейс, что устранило необходимость в обходных путях и дополнительном тестировании, — тогда любой элемент бэклога, связанный с этой интеграцией, становится сравнительно меньше.

РАЗОГРЕВАЕМ КОМАНДУ

Чтобы подготовить всю команду к покеру планирования, важно подобрать небольшое количество эталонных элементов бэклога, соответствующих значениям очков в колоде карт (см. лайфхак 15). Это позволяет быстро создать определенный измерительный шаблон, благодаря которому команда может моментально вспомнить, что такое «13-поинтовый» или «1-поинтовый» элемент бэклога (и все, что между ними). Разошлите эти эталонные значения за несколько дней до начала сессии покера планирования с коротеньким напоминанием утром перед встречей.

БОЛЬШИЕ КАРТЫ ДЛЯ БОЛЬШИХ СЛУЧАЕВ

Я обычно рекомендую изымать из колоды большие карты (20, 40, 100, бесконечность), а также карту ½. Таким образом вы сокращаете выбор карт до шести, а чем меньше вариантов выбора, тем меньше аналитический паралич. Кроме того, такой набор карт отбивает охоту у владельцев продукта приносить для оценки в команду слишком большие или пространные истории. Несмотря на это, Майк Кон предлагает актуальный сценарий, в рамках которого найдется место и большим картам:

Предположим, ваш руководитель хочет понять общий размер рассматриваемого проекта. Ему не нужна максимально точная оценка. Достаточно будет оценки вроде «около года» или «от трех до шести месяцев». Чтобы ответить на этот вопрос, вы захотите сформировать

бэклог продукта и при этом потратить не больше усилий, чем требуется. Раз вашему руководителю нужна верхнеуровневая оценка, вы можете сформировать верхнеуровневый бэклог. В этот бэклог достаточно включить только большие пользовательские истории — эпики, которые описывают значительные части функционала продукта. Эти самые эпики как раз могут быть оценены большими значениями стори-поинтов.

В любом случае использование карт с большими значениями будет сигнализировать о высоком уровне неопределенности на данном этапе. И лучшая оценка, которая может быть использована в таких случаях, — это оценка общей величины, а не конкретной продолжительности по времени.

НЕ УДВАИВАЙТЕ

Часто бывает, что ряд элементов бэклога продукта оказывается в зависимости от какого-нибудь важного исследования или технических работ. Если так случилось, убедитесь, что одна и та же работа не оценена дважды. Эта работа, связанная с целым рядом элементов бэклога, должна быть включена в оценку только одного элемента, но не каждого из них. Вам решать, в какой элемент бэклога включить эти дополнительные задачи, но постарайтесь выбрать тот элемент, который вы планируете реализовать первым. Хотя этот совет выглядит самоочевидным, вы можете обнаружить, что ваша команда намерена оценивать каждый элемент бэклога отдельно, в изоляции от остальных. Так оно и будет, если вы не проясните этот момент.

ДОСТИГАЕМ СОГЛАСИЯ

После небольшого обсуждения команда проведет свой первый раунд покера планирования. Если по его итогам не удалось достичь согласия, я рекомендую задать команде следующие вопросы:

- Учли ли вы все необходимые работы, а не только те, что выполнялись в рамках вашего функционала?
- Насколько твердо вы уверены в своей оценке? Если у вас есть сомнения, хотите ли вы ее изменить?
- Есть ли кто-то, кто колеблется между двумя оценками? Если да, то готовы ли вы подвинуться в сторону той оценки, у которой больше голосов?

Если согласия достигнуть не удалось и после этих вопросов, настало время профасилитировать быструю дискуссию между теми, кто голосовал за самое высокое и самое низкое значение. Основная трудность здесь — не допустить, чтобы дискуссия переросла в затянувшийся спор со множеством технических подробностей. Суть этого общения — все участники команды должны основывать свои аргументы

на относительном сравнении оцениваемых элементов бэклога с эталонными (а не на сложностях грядущей технической реализации). Вспоминаем лайфхак 13: относительные оценки базируются скорее на сравнении, а не на декомпозиции.

Наконец, если команда попросту не может выбрать между двумя соседними значениями, вы должны перестраховаться и выбрать большее.

ТЕЛЕФОНЫ МОГУТ ПОМОЧЬ

Если только вы не строгий надсмотрщик, иногда на встречах участники вашей команды будут проверять мессенджеры. Но если они еще и играют в Angry Birds47, у вас большие проблемы! Вместо того чтобы выполнять роль брюзжащего учителя, сделайте телефоны частью встречи. Предложите команде установить одно из лицензионных приложений для покера планирования и используйте телефоны вместо обычных игральных карт. Игра в покер планирования на телефоне не только сильно затруднит игру на нем же в Angry Birds, но и избавит вас от трудоемкой задачи по сортировке игральных карт в конце встречи!

ЭТО ВСЕ О ПРЕИМУЩЕСТВАХ

После того как вы несколько раз сыграете в покер планирования и примените советы из этого раздела, преимущества оценок станут для вас очевидными. Но что делать, если руководитель вызовет вас на ковер из-за того, что ваша команда играет в карты на деньги компании? Вот несколько дополнительных аргументов, чтобы превратить раздосадованного начальника в адвоката покера планирования:

- Возможность быстро оценивать долгосрочный бэклог продукта без подробного технического задания и составления сложных карт зависимостей.
- Возможность получить оценки от кросс-функциональной команды специалистов, чтобы избежать завышения или занижения оценок.
- Возможность эффективно использовать знания, полученные в результате выполнения предыдущей работы.
- Возможность дать команде немного повеселиться, выполняя рутинную и скучную задачу по оценке. Встреча по покеру планирования интерактивная, живая и гораздо более быстрая, чем традиционные марафоны по оценке задач.

ПОМНИТЕ О ЗАКОНЕ ПАРКИНСОНА

На основе собственного опыта могу сказать, что, если scrum-мастер не контролирует темп и концентрацию внимания на встречах по покеру

планирования, они становятся бесконечной говорильней или даже превращаются в яростные баталии (и я даже не знаю, что хуже).

Ограничивайте дискуссии по времени и всегда помните закон Сирила Паркинсона: «Работа заполняет [все] время, отпущенное на нее».

Уверяю вас, если вы примените приемы из этого лайфхака, то встречи по покеру планирования не только станут убойными (в смысле более быстрыми, а не более жестокими), но и все участники будут наслаждаться ими намного больше!

ЛАЙФХАК 15. ПЕРЕХОДИМ К ОТНОСИТЕЛЬНЫМ ОЦЕНКАМ

Надеюсь, раз вы дошли до этого лайфхака, значит, вы убедились в том, что относительная оценка — отличный способ двигаться вперед. Гаснет свет, глаза спрятаны за темными очками, карты готовы к сдаче для вашей первой сессии покера планирования (см. лайфхак 14). Но подождите, с чего начать? Что на самом деле означает пользовательская история в 1 стори-поинт? Как насчет 13-поинтовой? Как наилучшим способом откалибровать истории, чтобы команде было от чего оттолкнуться в дальнейшем? Если вы задались этими вопросами, пожалуйста, читайте дальше.

ПОДХОД

Один из подходов к калибровке историй, которым любят пользоваться команды, состоит в том, чтобы найти самую маленькую историю в продуктовом бэклоге и присвоить ей значение в ½ стори-поинта (при условии, что используется последовательность Фибоначчи). После того как найдена эта отправная точка, команда идет по списку пользовательских историй и оценивает в 1 стори-поинт любые истории, которые приблизительно в два раза больше, чем те, что были оценены в ½ стори-поинта; в 2 стори-поинта — те, что в два раза больше 1-поинтовых и так далее.

Этот подход, безусловно, работает. Он кажется простым на первый взгляд, но в реальности на него может уйти значительно больше времени, чем вы изначально предполагали. Во-первых, команде придется прошерстить весь продуктовый бэклог, чтобы обнаружить кандидатов на самую маленькую историю. А во-вторых, команде предстоит достичь согласия в том, какую историю выбрать в качестве отправной точки.

Имейте в виду: для команды это новый процесс, поэтому нужно сократить любую неопределенность настолько, насколько это возможно. Именно по этим причинам я люблю использовать подход к калибровке историй на основании задач, выполненных в прошлом.

ИСПОЛЬЗУЕМ ДАННЫЕ ПО СДЕЛАННОЙ РАБОТЕ

Использование данных по уже проделанной работе позволяет сопоставить известные величины (старая завершенная работа) с новыми значениями стори-поинтов Фибоначчи (или любой другой шкалы, которую вы выберете).

Использование выполненных ранее работ дает команде два существенных преимущества: знакомые данные и согласованность.

Знакомые данные

Очевидно, любая команда лучше знакома с работой, которую выполняла раньше, чем с той, которую ей только предстоит выполнить. Это знание оказывается особенно полезным при игре в покер планирования (см. лайфхак 14), ведь вместо того, чтобы сравнивать одну будущую неизвестную работу с другой (как в первом подходе, описанном ранее), команды могут сравнивать грядущую неизвестную работу с прошлой известной. Этот подход не только устраняет элемент неопределенности, но и ускоряет процесс сравнения, ведь команде легче вспомнить работу, которая уже была сделана.

Согласованность

Если сформировать набор эталонных значений на основании информации об уже проделанной работе (для разных значений поинтов в колоде для покера планирования), то эти же эталонные значения можно использовать во всех проектах, над которыми работает одна и та же команда. Подобная подготовка ускорит все дальнейшие процедуры оценки, потому что первоначальный процесс создания «матрицы эталонных значений» требуется провести только один раз (в отличие от тех случаев, когда формируется и представляется новый бэклог продукта).

СОЗДАНИЕ СОПОСТАВЛЕНИЙ

Чтобы присвоить уже проделанной работе поинты, потребуется пять шагов. Шаги 2 и 3 я добавил в алгоритм, вдохновившись статьей Джеймса Гренннинга «Партия покера планирования»48, которая описывает похожий подход (но только используя новый бэклог продукта, а не уже проделанную работу).

Шаг 1. Составляем список

Находим недавний проект, который был выполнен этой же командой (или, по крайней мере, бо́льшей ее частью). Составляем список работ, которые были выполнены по этому проекту, и записываем их на учетные карточки (если список был в цифровом формате). Если эти работы

сформулированы не в формате пользовательских историй (см. лайфхак 10), их нужно переформулировать, чтобы обеспечить однородность для сравнения (см. рис. 5.7).

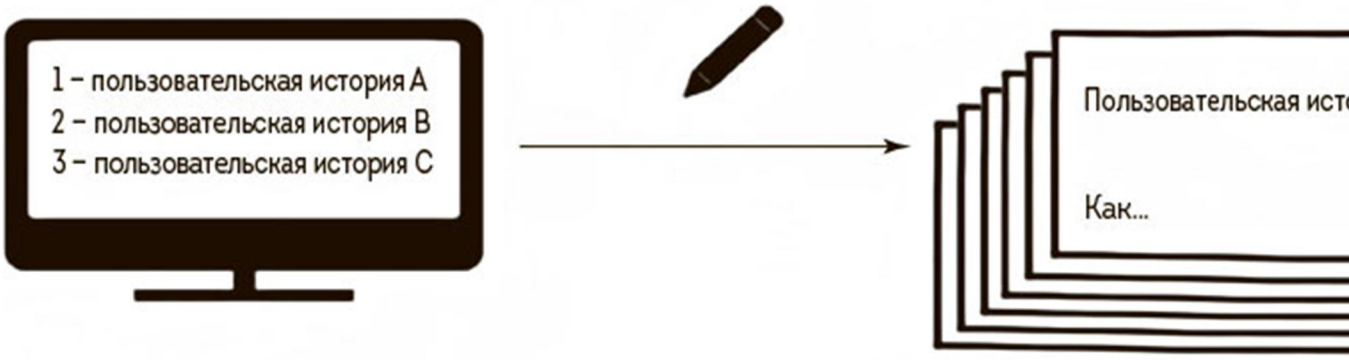

Рис. 5.7. Перенесите оцифрованные требования на учетные карточки

Шаг 2. Сортируем и складываем

Для следующего шага вам понадобятся большой стол и команда разработки. Начиная с первой учетной карточки, прочитайте вслух пользовательскую историю и положите карточку на стол (см. рис. 5.8).

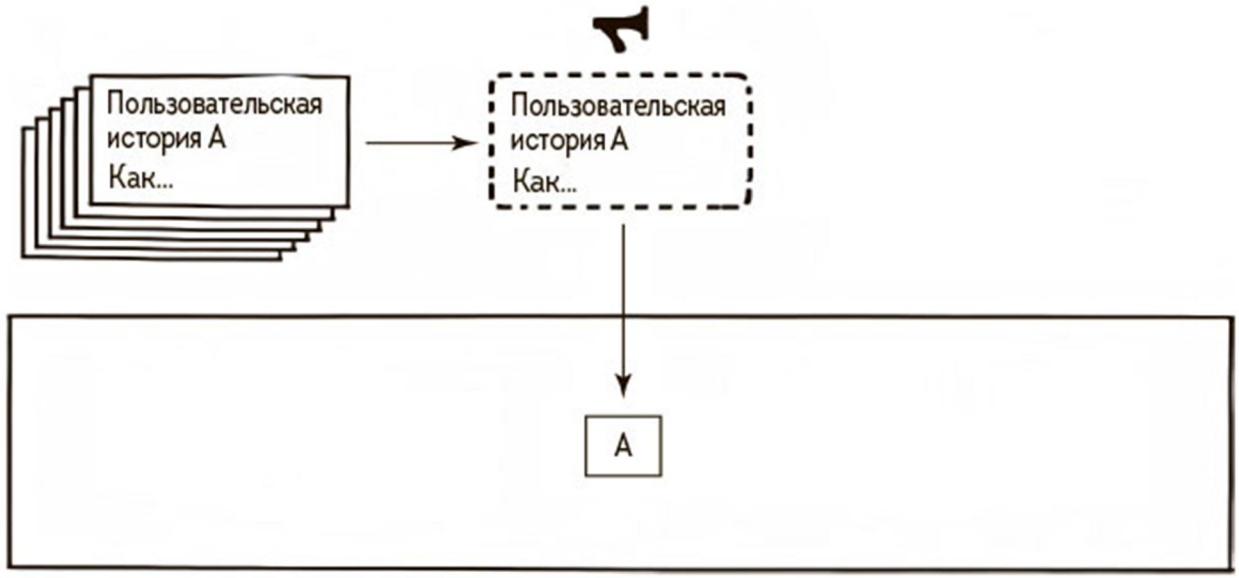

Рис. 5.8. Прочтите вслух первую карточку с пользовательской историей и разместите на столе

Возьмите следующую и спросите команду, считают ли они, что выполнение этой пользовательской истории потребовало больше, меньше или такое же количество усилий, как пользовательская история с первой карточки (см. рис. 5.9). Если усилий потребовалось меньше, поместите ее слева от первой карточки, если больше — справа, а если столько же — поверх первой карточки. Если у команды возникли разногласия или она вконец запуталась, «сожгите» карточку (только не на самом деле).

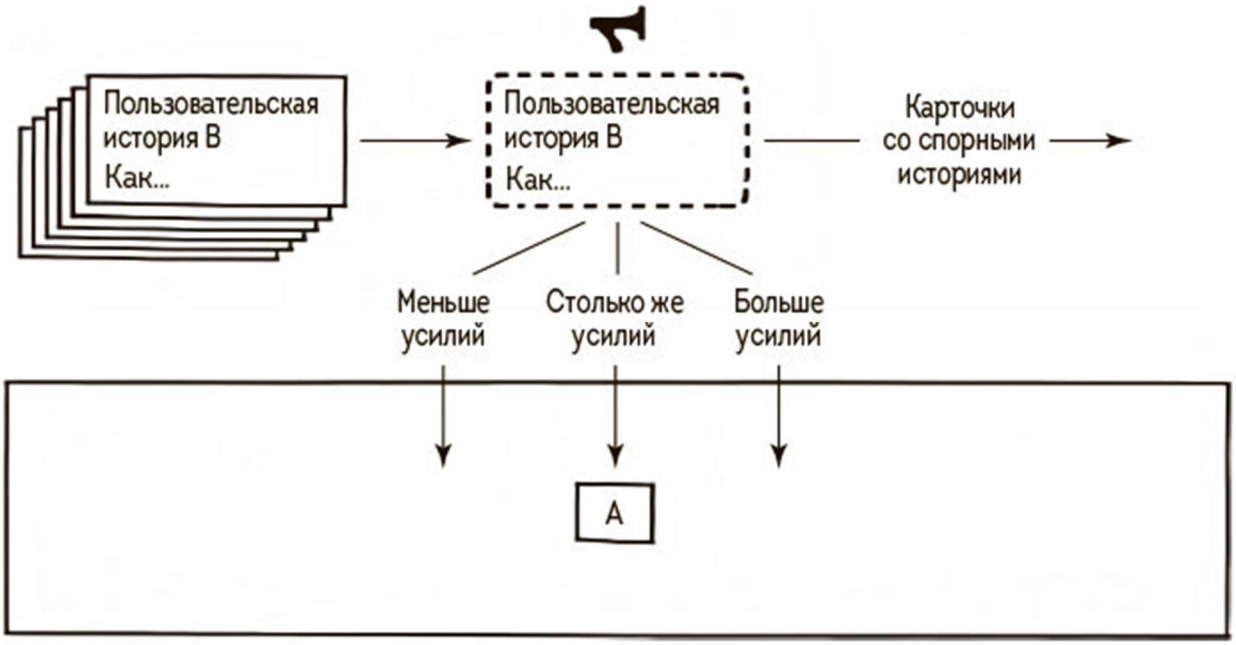

Рис. 5.9. Прочтите вторую историю и вместе с командой решите, потребовала ли ее реализация больше, меньше или столько же усилий, как первая

Возьмите следующую карточку и положите либо слева от предыдущих карточек (если ее выполнение потребовало меньше усилий), справа (если усилий потребовалось больше), между карточками (если усилия, которые вы затратили, находятся где-то посередине между предыдущими двумя) или поверх одной из карточек (если усилий потребовалось примерно столько же). Повторите процесс для всех учетных карточек (см. рис. 5.10).

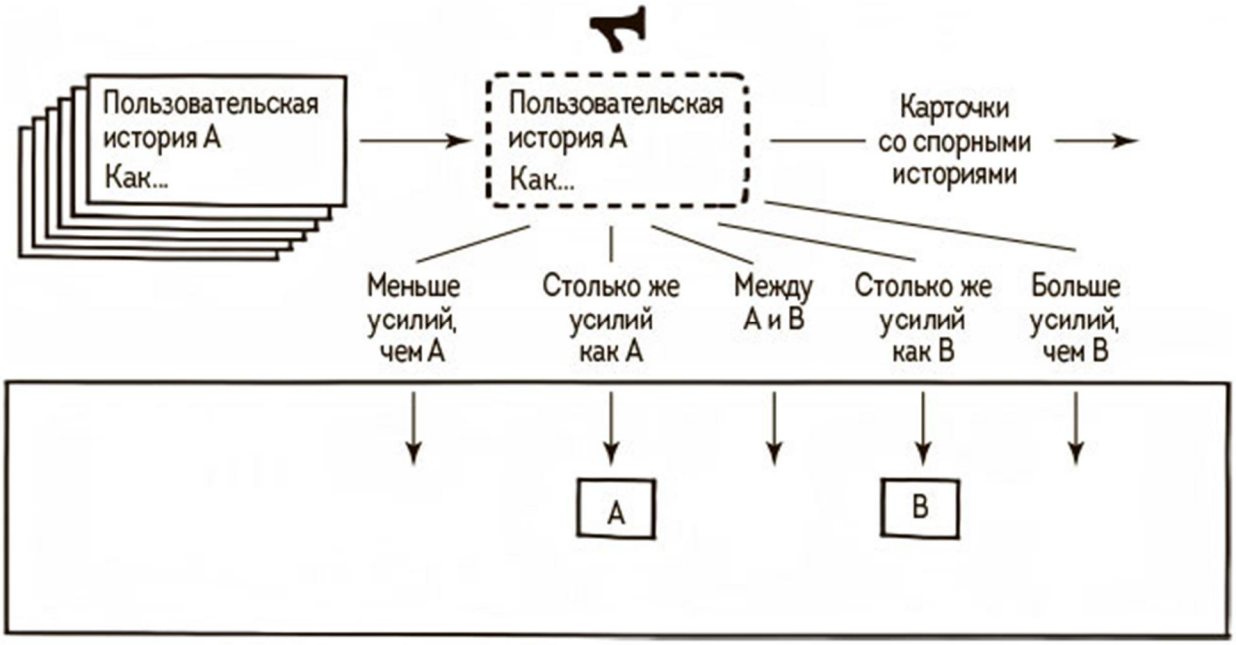

Рис. 5.10. Прочтите следующую историю и вместе с командой решите, как она соотносится с предшествующими историями

Шаг 3. Оцениваем

К этому этапу на столе должны появиться несколько стопок карточек разных размеров. Обратите внимание: я очень условно называю это стопкой — по итогам предыдущих шагов в ней может оказаться всего одна карточка. Стопка на левом конце стола будет содержать самые маленькие пользовательские истории, а стопка на правом конце — самые большие.

Теперь самое время сыграть в покер планирования (см. лайфхак 14). Всем карточкам из крайней левой стопки автоматически присваиваем значение 1 поинт (см. рис. 5.11). Я предпочитаю оставлять самое маленькое значение ½ поинта для элементарных изменений, например исправить надпись или выровнять текстовое поле. Если ваша стопка с самой маленькой работой не состоит из таких вот крошечных задач, предлагаю присвоить ей значение 1 поинт, а не ½.

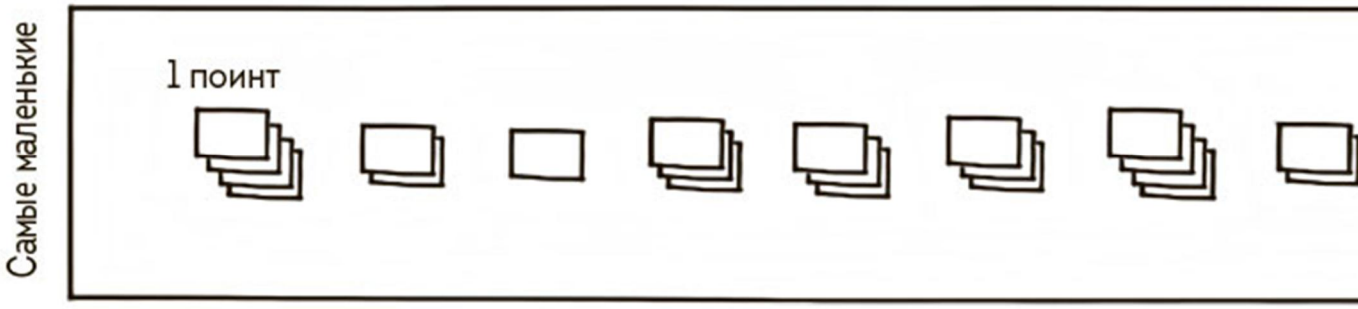

Рис. 5.11. После сортировки все самые маленькие истории будут лежать на одном конце стола, а большие — на другом

Переходим к карточкам из второй стопки с небольшими историями (лежит справа от стопки историй, оцененных в 1 поинт) и оцениваем усилия, которые потребовались на их завершение, — в сравнении с историями из первой стопки (например, в три раза больше усилий).

После того как вы присвоили значения поинтов каждой стопке, положите карточку с этим значением над стопкой (для наглядности). Так, в нашем примере вторая стопка будет помечена карточкой в 3 поинта.

Шаг 4. Создаем подгруппы

Если все прошло успешно, ваша сессия покера планирования пройдет гладко (благодаря приемам из лайфхака 14) и на столе появятся несколько стопок пользовательских историй с соответствующими каждой значениями поинтов.

В идеальном мире у вас нашелся бы стек для каждого из значений поинтов в той системе, которую вы планируете использовать. Но не переживайте, если этого не произошло (см. рис. 5.12). В конце концов, если у вас есть пара эталонных историй, вы уже можете начинать.

Рис. 5.12. После игры в покер планирования и присваивания значений поинтов вы можете получить примерно такие стопки

Если вам есть из чего выбирать и в каждой стопке оказалось по нескольку историй, вы можете разбить их на тематические подгруппы (см. рис. 5.13). Так, в стопке в 5 поинтов окажется три разные пяти-поинтовые истории. Хотя они уже сгруппированы вместе (на основании того количества усилий, которые нужно приложить для их реализации), у них могут быть очень разные точки приложения усилий. История 1 может быть связана со сложностями в оптимизации данных, история 2 — с пользовательским интерфейсом, а история 3 — потребовать интеграции со сторонним продуктом. Если разделить эти истории на подгруппы, возможность сравнивать яблоки с яблоками (когда оцениваем бэклог нового продукта) станет реальностью.

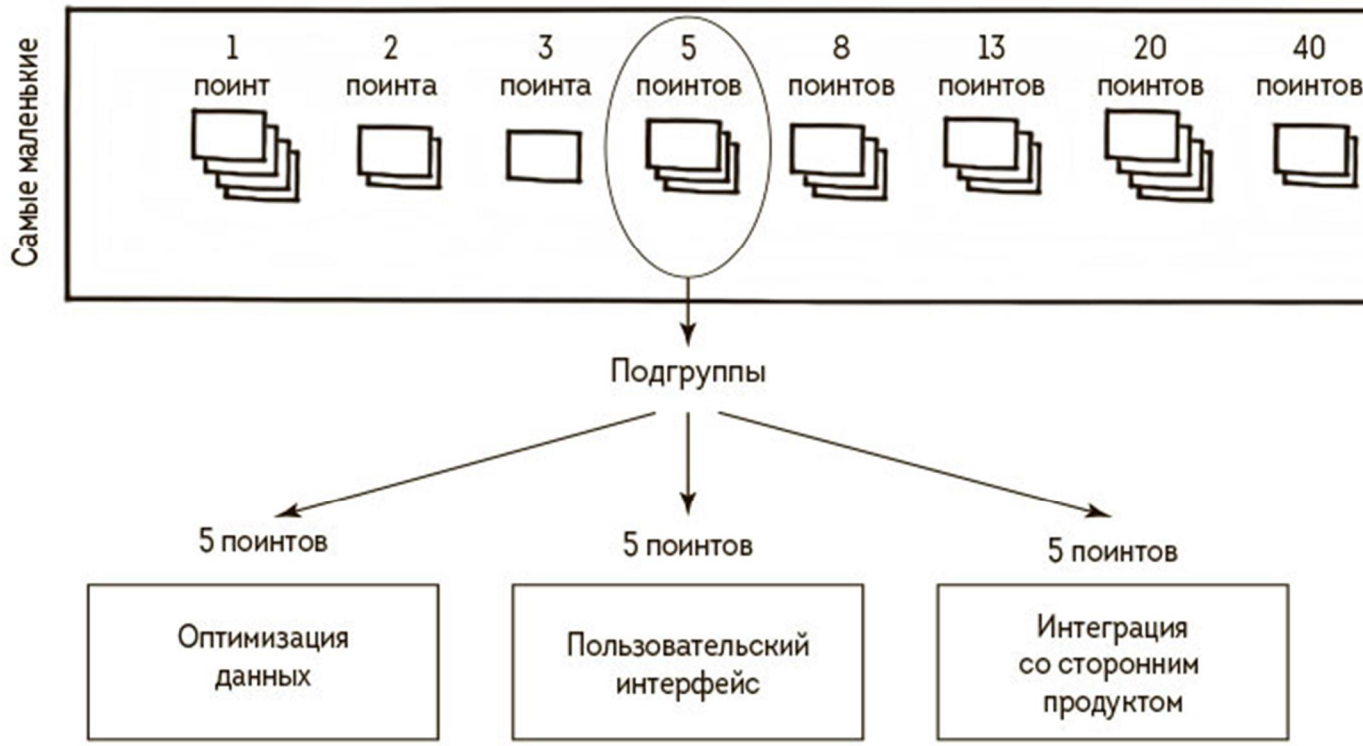

Рис. 5.13. Вы можете разбить истории из одной стопки на подгруппы в зависимости от их направления работы

Шаг 5. Производим финальный отбор

Последний шаг в этом упражнении по «калибровке» историй — выбрать по одной из каждой стопки (или подгруппы, если вы провели разделение, описанное на предыдущем шаге). Эти дошедшие до финала победители и станут эталонными историями, которые помогут начать проводить покер планирования (нового бэклога продукта). С учетом того, что истории уже были распределены по группам, вы можете выбрать эталонную историю случайным образом. Или, если вы более требовательны, команда может выбрать те истории, которые ей более знакомы.

ПРОДОЛЖАЙТЕ ИСПОЛЬЗОВАТЬ «ОТРАБОТАННЫЕ МАТЕРИАЛЫ»

Хотя вы уже завершили упражнение по первоначальной калибровке историй, я рекомендую проникнуться этой практикой и продолжать использовать ее. После завершения очередного проекта добавляйте любые завершенные истории в коллекцию эталонных историй, чтобы непрерывно пополнять библиотеку историй, которые не только знакомы команде, но и легко соотносятся с разными требованиями.

Теперь у вас есть всё. Вы вооружены алгоритмом использования информации об уже проделанной работе для создания шкалы эталонных историй. Благодаря информации об уже проделанной работе команда получает дополнительные преимущества от знакомых данных и согласованности, что облегчает переход к относительным оценкам и делает его менее запутанным.

ЗАКЛЮЧЕНИЕ

Три лайфхака этой главы посвящены выбору тактики, инструментов и приемов, которые помогут вашей команде во всем разобраться и перейти в мир относительных оценок. Давайте вспомним, что мы обсудили.

Лайфхак 13. Относительные оценки

- основные причины, чтобы оценивать требования;
- объяснение относительной оценки с использованием простых метафор;
- факторы, которые влияют на показательную скорость.

Лайфхак 14. Покер планирования в темпе

- порядок проведения сессии покера планирования;
- советы, чтобы убедиться, что команда разогрета и готова к игре;
- «побочные» эффекты покера планирования.

Лайфхак 15. Переходим к относительным оценкам

- преимущества использования данных об уже проделанной работе для калибровки изначальных эталонных поинтов;
- как соотнести старые, уже реализованные требования и сторипоинты;
- процесс создания широкого набора наглядных эталонных поинтов для будущих сессий оценки.

Глава 6

СПРАШИВАЕМ С КАЧЕСТВА

Не секрет, что компромиссное качество — спутник проектов по разработке программного обеспечения, реализуемых по традиционной модели водопада. Главным образом это происходит из-за весьма рискованной практики интегрировать и тестировать все сразу в самом конце. Хотя использование в Scrum итеративной и инкрементальной разработки значительно снижает эти риски, мы никогда не сможем исключить все дефекты.

Три следующих лайфхака научат избегать ошибок и исправлять их, если эти досадные баги уже случились.

Лайфхак 16 «Ба! Да это scrum-баг!» предлагает ряд новых определений, принципов и процессов, с помощью которых во время спринтов вы научитесь брать ошибки под контроль.

Лайфхак 17 «Мы по-прежнему любим тестировщиков!» раскрывает новые роли, которые берет на себя тестировщик в scrum-команде.

Наконец, лайфхак 18 «Нация автоматизации» выявляет отправные точки, важные для автоматизации тестирования.

ЛАЙФХАК 16. БА! ДА ЭТО SCRUM-БАГ!

Хотя нам больше не нужно бороться с реальными мотыльками, проникающими в вакуумные трубки (да-да, именно так возник термин «баг»), их цифровые потомки — завсегдатаи каждой кодовой базы на планете. По мере того как баги менялись, менялся и способ борьбы с ними. И особенно это важно для нового гибкого образа мышления.

Раньше, в «водопадном» мире, обработка багов шла последовательно (см. рис. 6.1).

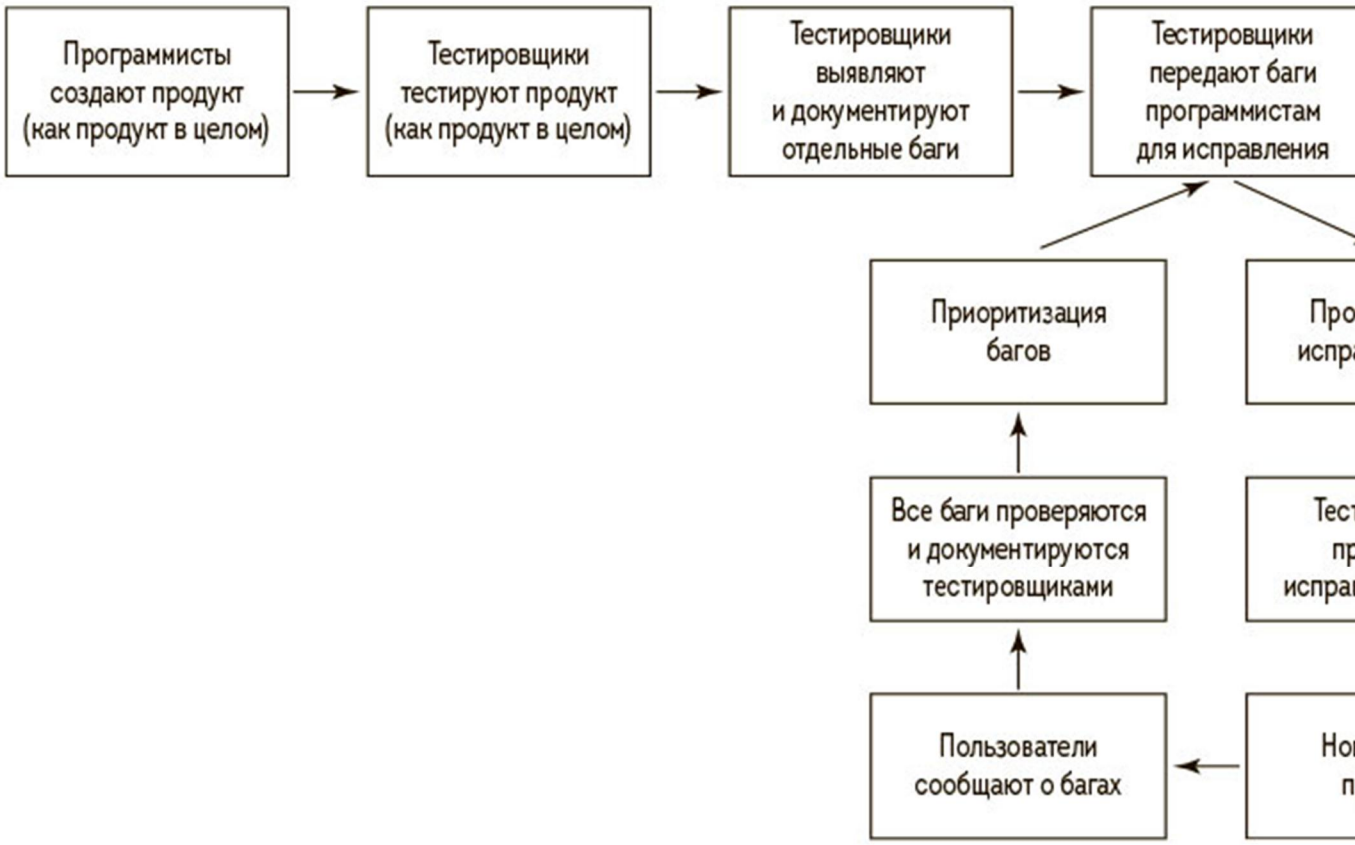

Рис. 6.1. Упрощенный порядок последовательной работы с багами в проектах, реализуемых по водопадной модели

Возьмем этот процесс в качестве отправной точки и посмотрим, что предстоит изменить при использовании Scrum. При этом важно понимать, что если команды собираются поставлять работающий функционал часто и как можно раньше, то программирование и тестирование должны идти в тандеме, а не одно следом за другим.

НОВЫЕ ОПРЕДЕЛЕНИЯ

Прежде чем разобрать новые процессы работы с багами, подходящие для Scrum, я хотел бы сформулировать основные определения и принципы.

Определение 1: ошибки

Ошибка — это проблема, возникшая во время спринта и тесно связанная с пользовательской историей (см. лайфхак 10), которая еще не достигла критериев готовности (см. лайфхак 11). Как правило, ошибки обнаруживаются во время спринта, в котором ведется работа над соответствующей пользовательской историей, либо программистом, либо при автоматизированной сборке (см. лайфхак 18), либо тестировщиком, проводящим исследовательское тестирование, либо владельцем продукта на этапе промежуточного контроля (см. лайфхак 12).

Ошибка *не* является элементом бэклога продукта, а должна рассматриваться как часть эволюционирующих критериев приемки пользовательской истории. Фактически до тех пор, пока ошибка не будет устранена, пользовательская история не может считаться завершенной. В таком случае ошибка является частью текущей пользовательской истории, а не независимым (хоть и связанным с ней) элементом бэклога продукта.

Определение 2: баги

Баг становится багом, только если он обнаружен после того, как пользовательская история была принята владельцем продукта. Вот почему баги, как правило, обнаруживаются пользователями (после релиза) или с помощью автоматического регрессионного тестирования (сопровождающего внедрение последующих пользовательских историй).

Баг — это одна разновидность элементов бэклога продукта, а пользовательская история — другая. Баги и пользовательские истории должны приоритизироваться вместе, в одном и том же бэклоге продукта, и оцениваться по одним и тем же критериям с использованием относительных единиц (см. лайфхак 13). Конкретный баг может относиться к определенной пользовательской истории, но должен рассматриваться, приоритизироваться и отслеживаться независимо от нее. Возвращаясь к лайфхаку 10, хочу сказать, что теоретически баг можно представить в классическом формате пользовательской истории, однако сам я в большинстве случаев не считаю такой формат подходящим.

НОВЫЕ ПРИНЦИПЫ

Теперь давайте рассмотрим три новых принципа.

Принцип 1. Избавьтесь от формализма

Давайте вспомним второй принцип Agile-манифеста: «Работающее ПО важнее исчерпывающей документации». В молодости, с головой уйдя в водопадную модель, я заметил, что и тестировщики, и программисты тратят немало времени на тщательное документирование мельчайших деталей даже относительно небольших ошибок. Помню, регулярно задавался вопросом, не занимает ли подготовка документации по этим проклятым ошибкам больше времени, чем их исправление. Scrum основывается на максимально возможном личном общении в режиме реального времени (а не на формализованных письменных отчетах об ошибках). Однако, если документация все же требуется, она должна соответствовать задачам и содержать необходимый минимум данных.

Принцип 2. Реагируйте на ошибки незамедлительно

Нет ничего хуже, чем ваш собственный устаревший код. Хотя нет, минуточку! Чужой устаревший код еще хуже! К сожалению, в то время когда мы все следовали алгоритму работы с ошибками, подробно описанному в начале этого лайфхака, было принято возвращаться к работе над ошибками в коде, от которого мы только что (причем достаточно успешно) отошли. На решение старых проблем (будь то ваши собственные ошибки или ошибки коллег, ушедших в отпуск) может быть потрачено слишком много времени, и, откровенно говоря, чаще всего впустую. Чем раньше проблема будет найдена, тем дешевле ее исправить. Именно поэтому в Scrum тестирование тесно переплетено с программированием.

Принцип 3. Это не конец, пока вы не дошли до конца

На самом деле, пока пользовательская история не будет соответствовать критериям готовности (см. лайфхак 11), для пользователя она не существует. Пользователи заинтересованы в конечных результатах и бизнес-ценности. Если пользовательская история еще не доведена до конца, она должна стать приоритетом для разработчиков. Им не следует переключаться на другую работу, пока эта задача не будет выполнена!

НОВЫЙ ПОДХОД

Приняв новые принципы для работы с багами, давайте сосредоточимся на нескольких подходах, которым я рекомендую следовать в спринте.

Сценарий 1. Тестировщик заканчивает исследовательское тестирование пользовательской истории и обнаруживает ошибку.

Во-первых, поскольку пользовательская история — приоритет номер один для программиста, работающего над ней (см. принцип 3), тестировщик может спокойно подойти к программисту и объяснить и/или показать ему ошибку сразу же, как обнаружит ее. В силу максимальной приоритетности пользовательской истории программист должен бросить то, чем занимается, и немедленно устранить обнаруженную ошибку. При таком раскладе достаточно устного обсуждения и необходимости в документации нет (если предположить, что ошибку тут же исправили и проверили).

Сценарий 2 аналогичен сценарию 1, но в этот раз программист занимается другой ошибкой, относящейся к этой же пользовательской истории.

В этом случае тестировщик, найдя очередную ошибку, оглядывается и видит программиста в наушниках, занятого исправлением предыдущей ошибки. Последнее, чего хочет тестировщик, — помешать ему устранить предыдущую ошибку. В таком случае важно зафиксировать детали

ошибки, чтобы тестировщик не забыл их и мог продолжить исследовательское тестирование.

Повторюсь, *ошибка* должна рассматриваться как часть критериев приемки пользовательской истории. Это избавляет тестировщика от необходимости создавать новый баг, детально описывать, классифицировать, приоритизировать и так далее. Я рекомендую просто добавить еще одну строчку в критерии приемки с указанием даты / времени, инициалов тестировщика и тезисно разъяснить суть проблемы. Когда программист освободится, можно будет сразу обсудить все ошибки, используя заметки в качестве подсказок. К тому же документация позволяет программисту ознакомиться с описанием дефектов, даже если тестировщика нет рядом.

Сценарий 3. В процессе финального приемочного пользовательского тестирования перед релизом обнаружился ряд небольших багов, связанных с пользовательским интерфейсом, которые пропустили в процессе разработки.

И снова сокращаем время, затрачиваемое на ненужный формализм. В этой ситуации я рекомендую создать отдельный элемент бэклога в качестве условного контейнера для фиксирования мелких багов. Каждая правка в отдельности может занять всего несколько минут, поэтому на создание отдельных элементов бэклога для каждой ошибки может уйти больше времени, чем на их исправление!

Следуйте этому подходу, только если:

- небольшие баги имеют одинаковый приоритет;
- баги отчасти взаимосвязаны, и потому резонно устранять их одновременно.

Если эти условия не соблюдены, создавайте отдельные элементы бэклога для каждой ошибки (даже если те и кажутся незначительными).

Сценарий 4. В ходе спринта обнаружился критический баг на продакшене, и кто-то из участников scrum-команды должен его устранить.

Первый вопрос, который следует задать: *«Насколько это действительно критично?» Иными словами, может ли его исправление подождать до следующего спринта?* Вспомним лайфхак 1: последнее, что вы хотите, — менять цель спринта. Если баг может подождать, он должен быть заведен в виде отдельного элемента бэклога продукта, приоритизирован владельцем продукта и потенциально взят в работу на следующем планировании спринта.

Как действовать, если обнаруженный баг оказался одним из тех страшных зол, на которые надо реагировать незамедлительно? Тогда задайте другой вопрос: *«Сколько времени потребуется, чтобы исправить этот баг?»* Вспомните лайфхак 8, из него мы усвоили, что

неразумно планировать всю емкость спринта исключительно под новые задачи. Нужно резервировать время на решение проблем, не связанных непосредственно с проектом. Это буферное время также может быть использовано для устранения внезапно возникшего бага без ущерба спринту.

Если же решение проблемы займет больше времени, чем было зарезервировано на форс-мажор, у вас есть два варианта. Во-первых, вы можете рассматривать эти проблемы как препятствия и отслеживать их соответствующим образом (см. лайфхак 9). Во-вторых, если проблема настолько серьезна, что полностью нивелирует цель спринта, остается запасной выход, пусть и нежелательный, — отмена спринта. Она может быть инициирована владельцем продукта. Отмена спринта завершает текущий спринт и возвращает команду к планированию нового спринта.

ПРЕВРАЩЕНИЕ МОТЫЛЬКОВ В БАБОЧЕК

Баги, безусловно, могут причинять боль, и, нравится нам это или нет, они никогда не станут вымершим видом. Однако мы узнали лучшие способы борьбы с ними. Мы усвоили, что избавляться от свежих ошибок проще, чем разбираться с застарелыми и усугубляющимися. А еще то, что документировать каждую ошибку — пустая трата времени.

Scrum предлагает работать с тестированием и багами совсем не так, как это принято в рамках традиционных подходов. Приняв это, а также новые принципы, вы избежите напрасной траты времени и сбоев в коммуникации, которые раньше мешали командам превращать мотыльков в бабочек.

ЛАЙФХАК 17. МЫ ПО-ПРЕЖНЕМУ ЛЮБИМ ТЕСТИРОВЩИКОВ!

На самом деле мы не только до сих пор любим тестировщиков, мы еще сильнее их любим в нашем новом scrum-мире! Чувствую, этот важный момент стоит подчеркнуть, и я скажу вам почему.

Помню, как я увлеченно рассказывал о Scrum моей первой scrum-команде. Я был уверен, что все подхватят мой заразительный энтузиазм, и действительно собрал кучу одобрительных кивков и улыбок. Однако, присмотревшись, я увидел, как ерзают некоторые тестировщики и как бегают их глаза (а это свидетельство дискомфорта и страха). Чтобы понять причину, нам придется заглянуть в прошлое и бегло рассмотреть, что же произошло с функцией тестирования в последнее время.

ВОДОПАД ДРУЖБЫ

В значительной степени благодаря давнему внедрению модели водопада сформировался нынешний подход к тестированию. Во многих компаниях сильная, независимая команда тестировщиков, стоящая наравне с командой программистов, стала нормой. Появились стандарты тестирования, открылись пути профессионального развития для тестировщиков (именно для них), и команда тестирования «завладела» целым этапом в процессе разработки программного обеспечения по модели водопада.

Но вот пришел Scrum (и его agile-собратья), и внезапно жизнь изменилась. Тестирование стало обязанностью каждого участника команды, юнит-тестирование — стандартной практикой программистов, даже функциональные тесты могут быть автоматизированы программистами. Взволнованные умы задаются вопросом: «А как сюда вписываются тестировщики?» Прежде чем идти дальше и не допустить ненужной паники среди читателей, перейду сразу к делу и скажу, что работа тестировщиков никогда прежде не была такой важной. Лайза Криспин и Джанет Грегори[49], авторы «Гибкого тестирования», подчеркивают, что общекомандный подход — главное отличие agile-разработки от традиционной разработки. Некоторые тестировщики принимают эту разницу и сразу же становятся сторонниками Scrum, а другие просто боятся нового порядка.

ПЕРЕМЕНЫ ВИТАЮТ В ВОЗДУХЕ

Изменения пугают. Криспин и Грегори видят несколько причин, по которым переход на agile может привести в растерянность некоторых тестировщиков. По их мнению, страх потери профессиональной идентичности — основная причина беспокойства тестировщиков. Вот как это проявляется[50]:

- боязнь утратить свою идентичность как специалистов, отвечающих за контроль качества;
- страх того, что не хватит компетенций для работы в agile-команде и они потеряют место;
- боязнь не получить необходимую поддержку при распределении по командам разработки.

Ниже я покажу, что в настоящей scrum-среде ни один из этих страхов неоправдан. Да, изменение профессиональной идентичности имеет место быть. Однако все профессиональные тестировщики, с которыми я работал, либо немедленно, либо со временем принимали новую расширенную идентичность с распростертыми объятиями.

В период перемен вполне естественно романтизировать прошлое и цепляться за все, что казалось надежным, а не вспоминать отрицательные и дурные моменты. Тестировщикам не стоит забывать, что прежде жизнь отнюдь не была прогулкой по парку (даже если им встречались красивые

водопады). Образ, запечатленный моей памятью, — это измученные, выгоревшие под конец «водопадного» проекта тестировщики. «Традиционные команды тестировщиков привыкли к скоростному осатанелому тестированию в конце проекта… в agile-проектах вы можете работать в спокойном темпе»51.

НОВАЯ ПРОФЕССИОНАЛЬНАЯ ИДЕНТИЧНОСТЬ

Как помочь тестировщикам принять свою роль в новом мире, где «водопады высохли»?

Для начала давайте признаем, что слон в комнате есть52: это страх остаться не у дел. Хотя по большому счету тестировщики должны чувствовать себя в безопасности, потому что они уникальны. Они обладают исключительным набором навыков и способом мышления, которые имеют решающее значение для успеха любого проекта по разработке программного обеспечения. Я люблю описание, предложенное Ником Дженкинсом в книге A Software Testing Primer («Азбука тестировщиков программного обеспечения»):

В основе «хорошего тестирования» лежит специальная философия. Профессиональный тестировщик подходит к продукту так, будто тот уже сломан: у него есть дефекты, и работа тестировщика — обнаружить их… Программисты исполнены оптимизма, они считают, что вносимые ими изменения являются верным решением конкретной проблемы… Благодаря скептическому подходу тестировщиков достигается баланс в отношении продукта. Своими «исследованиями» они стремятся пролить свет на темную часть проектов.

Словом, тестировщики мыслят альтернативными шаблонами (для решения проблем), которые прямо противоположны образу мышления программистов.

Теперь, когда мы разобрались с большим страхом, давайте рассмотрим несколько заманчивых новых субролей, которые может играть scrum-тестировщик. Эти роли явно более значимы и лежат за пределами скучного монотонного ручного тестирования, занимавшего прежде непропорционально большое место в жизни тестировщика (см. рис. 6.2).

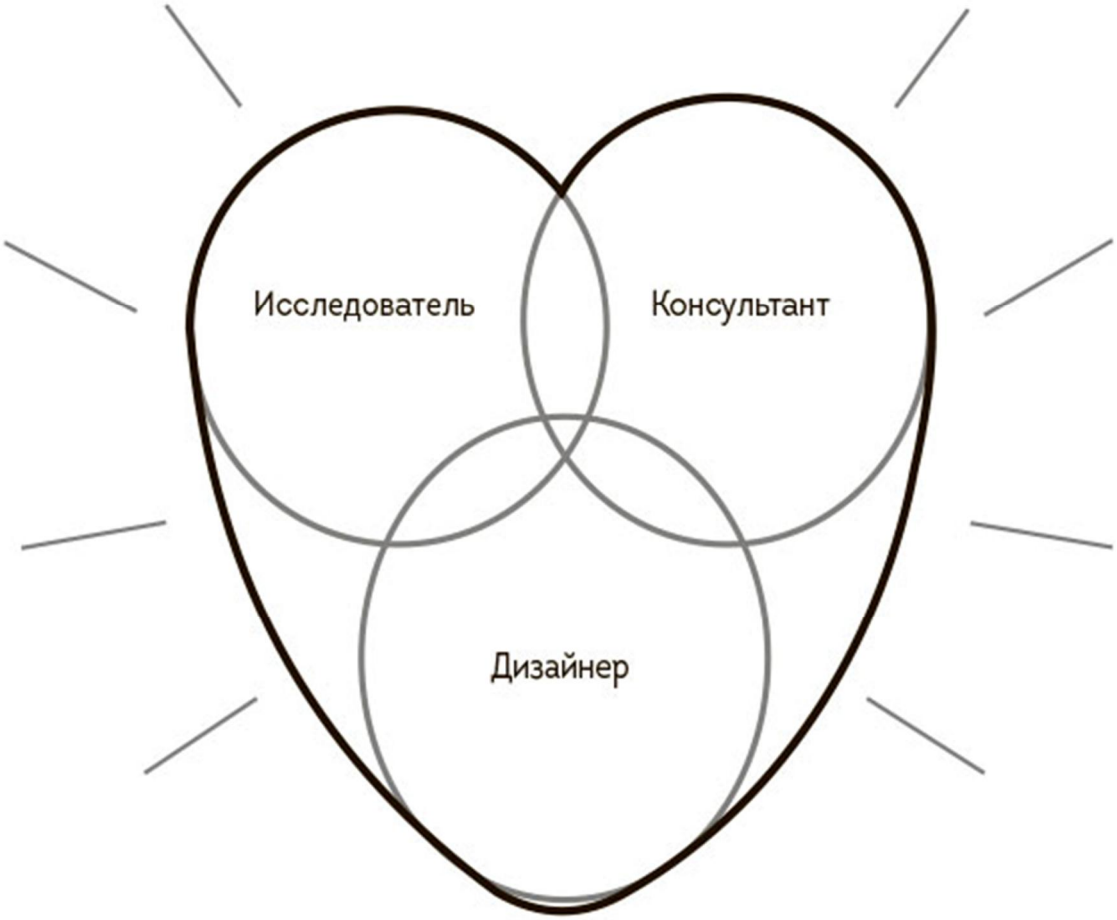

Рис. 6.2. В Scrum мы по-прежнему любим тестировщиков, особенно с их новыми scrum-функциями

ТЕСТИРОВЩИК-КОНСУЛЬТАНТ

Тестировщики — мастера своего дела, они находятся в уникальном положении, которое позволяет им помогать нетестировщикам улучшать их тестирование. Учитывая фокус Scrum на регулярной поставке качественного работающего программного обеспечения, это никогда еще не было настолько важно. Начнем с того, что тестировщик может (и должен) стать опорой программистам, которые только осваивают разработку через тестирование (test-driven development).

Хотя парное программирование является, безусловно, эффективной техникой экстремального программирования (XP) (которое иногда применяется scrum-командами), я считаю «парное тестирование» (когда тестировщик работает в паре с программистом) потенциально еще более эффективным методом, потому что оно предоставляет дополнительную возможность обмена функциональными навыками. Это также способствует уважительному отношению к навыкам и способностям друг друга.

Консультации тестировщиков могут принести немалую пользу и UX-дизайнерам, так как научат предвидеть потенциальные проблемы, связанные с более сложными сценариями поведения пользователей.

Наконец, владелец продукта может использовать присущее тестировщику глубокое понимание основных критериев приемки: тестировщик может поддерживать его во время промежуточного контроля внутри спринта (см. лайфхак 12) и помогать с окончательной проверкой завершенных пользовательских историй.

ТЕСТИРОВЩИК-ДИЗАЙНЕР

На мой взгляд, основной навык тестировщика — тест-дизайн. Независимо от того, кто проводит или использует тесты, наиболее эффективные тест-кейсы разрабатывает самый опытный и профессиональный тестировщик в команде.

Хорошо продуманные тесты не только формируют основу тестирования, но и вносят заметный вклад в технический дизайн во время планирования спринта. Когда тестировщик участвует в разработке тест-кейсов пользовательских историй до планирования спринта, я могу заверить вас, что планирование пройдет более гладко, быстро и с меньшим количеством споров. Для тех, кто боится скатиться к «водопадным» спринтам, следуя этому совету, я приведу мысли Майка Кона53:

Быть частью команды в этом (текущем) спринте и потратить немного времени, чтобы заглянуть вперед, — не то же самое, что работать впереди команды целый спринт… приоритет команды — поставка всего, что она обязалась выполнить в этом спринте. Кроме того, ее задача — смотреть вперед в том же направлении, в каком, как ожидается, смотрит и владелец продукта.

ТЕСТИРОВЩИК-ИССЛЕДОВАТЕЛЬ

Из лайфхака 18 вы узнаете, что автоматизация тестирования — неотъемлемая часть успеха Scrum. Тем не менее даже при максимально полной автоматизации тестирования всегда будет потребность в ручном исследовательском тестировании, которое не заменить ни при каком уровне автоматизации. Эта область тестирования, без сомнения, больше похожа на искусство, чем на науку. Для тех, у кого сложилось ложное впечатление, будто исследовательское тестирование — синоним интуитивного тестирования54 или «горилла»-тестирования55, следующий комментарий Криспин и Грегори позволит по-новому посмотреть на особенности этой области56:

В исследовательском тестировании у каждого тестировщика свой подход к проблеме и у каждого уникальный стиль работы. Тем не менее есть определенные качества, типичные для хорошего исследовательского тестировщика. Хороший тестировщик:

- систематичен, но везде выискивает «душок» (аномалии и противоречия);
- учится распознавать проблемы с помощью «оракула»57 (принцип или механизм, с помощью которого мы распознаем проблему);
- выбирает тему, роль или миссию, чтобы задать направление тестирования;
- ограничивает время сессий и переключения на другие задачи;
- думает о том, что будут делать пользователь-эксперт или пользователь-новичок;
- проводит исследования вместе с экспертами в той или иной области;
- проверяет аналогичные приложения или приложения конкурентов.

НОВОЕ НАЧАЛО

В scrum-команде каждый отвечает за тестирование. Теперь мы не занимаемся проблемами качества по остаточному принципу. Тестирование должно стать неотъемлемой частью каждого этапа разработки пользовательской истории, в том числе до того, как первая строка функционального кода будет написана.

Переход к Scrum должен стать для тестировщиков захватывающим перерождением. Избавление от кандалов ручного тестирования освободит scrum-тестировщиков и даст им возможность сосредоточиться на том, что они делают лучше всего: тест-дизайн, консультирование и исследовательское тестирование. Наконец-то тестировщики смогут проявить свои уникальные навыки гораздо более интересными способами, чем раньше.

ЛАЙФХАК 18. НАЦИЯ АВТОМАТИЗАЦИИ

У вас есть простой выбор: оседлать идею автоматизации, которая перенесет вас в прекрасные места, наполненные Scrum, или страдать, соскальзывая по склону «scrummer-пада».

Использовать Scrum без автоматизации все равно что вести спортивный автомобиль по разбитой грунтовой дороге — вы не сможете почувствовать всю мощь вашей потрясающей машины, страшно расстроитесь, скорее всего, попадете в аварию и обвините в этом автомобиль. Как указывают Джеймс Шор и Шейн Уорден58:

Разработка программного обеспечения предъявляет высокие требования. Она требует совершенства и постоянных усилий на протяжении нескольких месяцев и даже лет. В лучшем случае из-за ошибок код просто не скомпилируется. В худшем ошибки будут ждать своего часа, чтобы проявиться именно в тот момент, когда ущерб от них будет максимальным.

В *Scrum Guide*, написанном Кеном Швабером и Джеффом Сазерлендом, вообще не упоминаются практики разработки программного обеспечения, там нет даже слов *«программное обеспечение» и «инженерные практики».* Scrum здесь рассматривается на более высоком уровне и обобщенно описывается как «фреймворк для разработки и поддержки сложных продуктов».

При этом ни один эксперт никогда не скажет, что Scrum (в контексте его применения в программном обеспечении) не приносит больше пользы в сочетании с сильными практиками по автоматизации разработки программного обеспечения. Такими, которые вы найдете в наборе практик экстремального программирования (XP)[59]. В этом лайфхаке я расскажу, почему так оно и есть.

Автоматизация — обширная тема, ей посвящено множество книг. В этом лайфхаке я дам вам несколько общих советов, чтобы вы начали свой путь к автоматизации. Есть много уровней, много инструментов и разных комбинаций инструментов. Однако я намерен сделать этот лайфхак простым и лаконичным, чтобы избежать аналитического паралича.

Мы сфокусируемся на нескольких ключевых практиках автоматизации, которые тесно связаны между собой, включая непрерывную интеграцию, автоматизацию тестирования, автоматизацию сборки / развертывания и относительно новую концепцию непрерывной поставки.

НЕПРЕРЫВНАЯ ИНТЕГРАЦИЯ

Мартин Фаулер, подписавший Agile-манифест в числе первых, так описывает ключевую практику непрерывной интеграции[60]:

Непрерывная интеграция — это практика разработки программного обеспечения, при которой участники команды часто интегрируют свою работу в общую ветку. Обычно каждый участник команды интегрируется минимум раз в день, что приводит к нескольким интеграциям в течение дня. Каждая интеграция проверяется во время автоматизированной сборки (включая тест) для максимально быстрого обнаружения ошибок интеграции.

В чем же тогда выгода (если она еще неочевидна)? Опять же я не думаю, что смогу описать лучше Фаулера:

Интеграция — это долгий и непредсказуемый процесс… Проблема отложенной интеграции в том, что очень трудно предсказать, сколько времени она займет. И что еще хуже, крайне сложно оценить, насколько далеко вы уже продвинулись. Так вы сами себя загоняете в «слепую зону» на одном из [самых напряженных] этапов проекта — даже если вы еще не опоздали (а это большая редкость)[61].

Отладить непрерывную интеграцию — отличный способ начать свой путь автоматизации. Scrum-тренер Кейн Мар особо подчеркивает этот момент.

Разработка инкремента потенциально готового кода без непрерывной интеграции практически невозможна после третьего или четвертого спринта. Просто потому, что количество изменений и регрессионного тестирования станет неподъемным»62.

Непрерывная интеграция — классическая иллюстрация пословицы *«Один стежок, сделанный вовремя, экономит девять».* Решая небольшие проблемы интеграции каждый день (вместо того, чтобы пытаться совладать с лавиной осложнений под конец проекта), команды избавляют себя от сильного стресса и запредельной тревоги.

Сервер непрерывной интеграции постоянно отслеживает любой новый код, автоматически запуская после проверки новую сборку. В этом процессе сервер может (и должен) выполнять любые автоматические тесты. Поскольку каждый день запускается множество сборок непрерывной интеграции, очень важно обеспечить максимально быструю сборку — если она занимает более 10–15 минут, в разработке возникают «бутылочные горлышки», что мешает достичь цели. Ограничение в 10–15 минут означает, что сборка непрерывной интеграции не включает более медленные функциональные тесты (их запускают во вторичную сборку). Обсудим это чуть позже.

АВТОМАТИЗАЦИЯ ТЕСТИРОВАНИЯ

Не нужно быть гением, чтобы понять, что случится, если автоматизация тестирования не будет внедрена. К третьему или четвертому спринту ручного регрессионного тестирования станет так много, что некоторым участникам команды покажется соблазнительным запустить узконаправленный «спринт тестирования» (см. рис. 6.3) и тем самым решить проблему. Идея так себе, но еще хуже будет, если некоторые умники предложат «команде тестирования» провести спринт (или больше) над завершением регрессионного тестирования. Если у вас произошло нечто подобное, значит, ваша команда скатывается обратно в неинкрементальный мир разработки по модели «водопада».

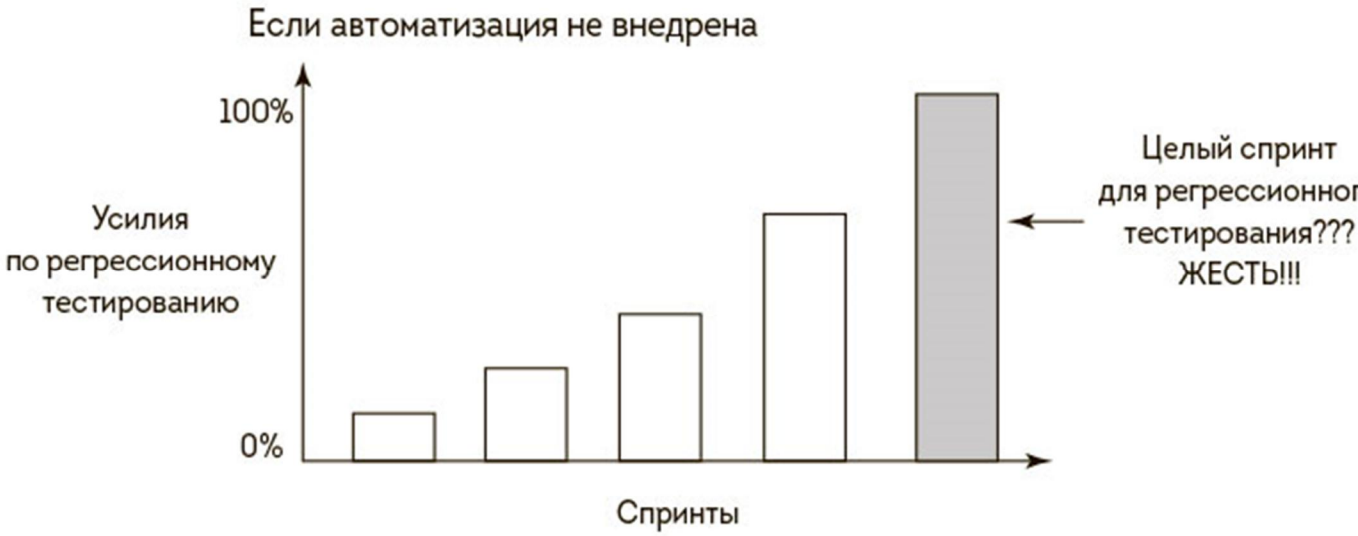

Рис. 6.3. Если автоматизация тестирования не внедрена, «тестовый спринт» караулит вас за углом

Хорошая новость: после автоматизации тестирования нет причин опасаться, что вас отбросит обратно в темные века. Плохая новость, как объясняют Лайза Криспин и Джанет Грегори, в том, что «автоматизация невозможна без крупных инвестиций, которые могут не окупиться сразу. Требуется время и исследования, чтобы решить, какие фреймворки для автоматизации тестирования использовать и разрабатывать ли их самостоятельно или внедрять внешние инструменты»[63].

Согласен, инвестиции необходимы, ведь плюсы автоматизации намного перевешивают минусы, связанные с расходами и ее внедрением. Вот некоторые преимущества, которые отмечают Криспин и Грегори:

- ручное тестирование занимает слишком много времени (по сравнению с автоматизированным);
- ошибок в ручных процессах гораздо больше;
- автоматизация освобождает людей для более полезной работы;
- автоматизированные регрессионные тесты создают безопасную сеть;
- автоматизированные тесты позволяют получать обратную связь рано и часто;
- тесты и примеры, которые драйвят разработку, могут давать больше;
- тесты позволяют составить документацию;
- инвестиции в автоматизацию могут окупиться сторицей.

Виды автоматизированного тестирования

Первое знакомство с миром автоматизированного тестирования может быть немного пугающим. Мало того, что существует множество уровней тестирования, отрасль до сих пор не выработала стандартного нейминга для них, а также не определила точные границы для каждого уровня.

Поэтому названия и описания разных уровней, которые я буду использовать дальше, могут отличаться от тех, с которыми вы знакомы.

Юнит-тестирование

Юнит-тесты предназначены для тестирования блоков программного кода самого низкого уровня (таких как метод в классе) и обычно реализуются с помощью одного из фреймворков xUnit. Эти тесты должны быть реализованы посредством разработки через тестирование, обеспечивающей дополнительное преимущество.

Если ваши программисты используют разработку через тестирование в качестве инструмента для написания своих тестов, значит, они не только создают отличный пакет для регрессии, но и используют тесты для разработки высококачественного надежного кода64.

Для тех из вас, кто еще не знаком с разработкой через тестирование, Рон Джеффрис, один из подписантов Agile-манифеста и scrum-тренер, предлагает отличное объяснение:

Разработка через тестирование требует написания всего кода, и только того кода, который нужен для прохождения тестов, написанных в первую очередь. Это помогает мне сосредоточиться на том, ЧТО должен делать код, прежде чем фокусироваться на том, КАК он это сделает. В результате возникает простой код, который легко тестировать. Разработка через тестирование — это не глупая, чисто механическая практика, а скорее медитативная сосредоточенность на том, что происходит. Это уменьшает количество моих ошибок и, более того, снижает напряжение65.

Функциональное тестирование

Функциональное тестирование также часто называют *приемочным тестированием.* Возможно, однажды мы будем называть его *тестированием пользовательских историй*, поскольку глобальная цель — тестировать и автоматизировать всю функциональность каждой конкретной пользовательской истории от начала и до конца.

Такие типы тестов могут включать или не включать автоматизацию тестирования пользовательского интерфейса (UI). Этот уровень обычно требует дополнительного времени на выполнение тестов пользовательского интерфейса (UI), к тому же он довольно хрупок (пользовательский интерфейс часто корректируется в процессе разработки). Для тех, кто хочет тестировать все, кроме UI, будут эффективны такие инструменты, как FitNesse66. Для тестирования всех уровней можно использовать Selenium67. И у FitNesse, и у Selenium открытый исходный код, и оба они общедоступны.

Интегрированное тестирование

Интегрированное тестирование (часто его еще называют *системным тестированием*) — гарантия того, что новая функциональность впишется в экосистему проекта, а не будет функционировать сама по себе. Например, разрабатываемый продукт, возможно, будет необходимо интегрировать с другими внутренними продуктами, такими как инструменты администрирования, или со сторонними продуктами, такими как платежные шлюзы.

Тестирование производительности

Альтернативные названия существуют и для тестирования производительности: *нагрузочное тестирование* и *стресс-тестирование.* Основное внимание в этом случае уделяется тестированию работы продукта под высокими нагрузками, обычно вызываемыми наплывом пользователей и/или увеличением объема необходимой обработки данных.

Пока вы не производите релизы с частотой спринта (или даже чаще), вам не обязательно тестировать производительность каждый спринт. «Специальные виды тестирования, такие как интегрированное тестирование, тестирование производительности, юзабилити-тестирование и так далее, можно не выполнять в каждом спринте»[68]. Однако не делайте интервалы между интеграцией и тестированием производительности слишком длинными, потому что ошибки, обнаруженные поздно, могут потребовать «вернуться к чертежной доске» и начать все заново.

АВТОМАТИЗАЦИЯ ДЕПЛОЯ

Если вы думаете, что внедрить Scrum и работать в хорошенькой, чистенькой среде разработки достаточно, то пришло время столкнуться с суровой реальностью. Если ваша команда не готова уверенно одним нажатием кнопки отправить свой код из безопасной среды разработки на большой и скверный продакшен, вам предстоит признать, что ваш поэтапный подход еще одна разновидность «scrummer-пада». Проще говоря, ваша команда должна сделать все возможное, чтобы полностью автоматизировать процесс сборки и деплоя для всех используемых сред.

Помня об этом, давайте рассмотрим основные среды и их связь с разными автоматизированными тестами.

Среда разработки

Ранее в этом лайфхаке мы сосредоточились на преимуществах сервера непрерывной интеграции и на том, что он должен автоматически запускать сборку после каждой проверки.

Помимо сборки в ходе непрерывной интеграции я рекомендую запускать вручную (и не так часто) повторную сборку в среде разработки

(традиционно называемую ночной сборкой). Эта сборка не ограничена по времени и потому может включать полный комплекс тестов (в том числе все тяжелые функциональные и UI-тесты, занимающие существенно больше времени). Шор и Уорден указывают на некоторые дополнительные функции, которые несет на себе повторная сборка:

Помимо компиляции исходного кода и выполнения тестов она должна формировать настройки реестра, запускать схемы баз данных, настраивать веб-серверы, запускать процессы — все, что необходимо для создания и тестирования программного обеспечения с нуля без вмешательства человека[69].

Основная сложность здесь — обеспечить максимальное сходство среды разработки с тестовой средой и продакшеном. По разным причинам, таким как лицензирование и скорость, у вас не получится скопировать все связанные компоненты, однако вы сможете имитировать их, используя мок-объекты[70] для симуляции реальности. Кроме того, даже если копировать весь набор данных, используемый на продакшене, нежелательно (например, он очень много весит, а вы хотите сэкономить время), то набор данных, используемый в среде разработки, должен как минимум отражать реальные данные, чтобы обеспечить их целостность.

Почему это так важно? Постоянно «репетируя» релизы на продакшен, вы проводите системное тестирование каждый день, а не в самом конце, когда время на исправление ошибок весьма ограничено.

Тестовая среда

Если среда разработки должна имитировать продакшен, то тестовая среда должна быть *идентична* ему, то есть содержать полный набор данных, а также все сторонние продукты и компоненты, с которыми взаимодействует ваш продукт.

В этой среде может проводиться интегрированное тестирование, часто она становится основным местом для тестирования производительности.

НЕПРЕРЫВНАЯ ПОСТАВКА И SCRUM

Все больше людей в agile-сообществе применяют такие подходы, как *непрерывное развертывание* или *непрерывная поставка.* Хотя термины часто используются как синонимы, Джез Хамбл, соавтор книги «Непрерывное развертывание ПО»[71], объясняет разницу между ними:

Непрерывное развертывание предполагает непрерывную поставку, однако обратное утверждение неверно. Непрерывная поставка — это передача релиза в руки бизнеса, а не в руки IT. Реализация непрерывной поставки означает, что ваше программное обеспечение всегда готово к релизу на протяжении всего жизненного цикла — что любая сборка потенциально может быть выпущена для пользователей одним

нажатием кнопки в считаные секунды или минуты, с помощью полностью автоматизированного процесса.

Таким образом, в то время как непрерывная поставка делает каждую сборку программы потенциально годной для пользователей, непрерывное развертывание фактически *выпускает* каждое изменение «на бой» — пользователям (иногда несколько раз в день). Я говорю про эти подходы, потому что хочу развеять несколько мифов.

Первый миф. Scrum и непрерывное развертывание / поставка исключают друг друга. В некоторых кругах бытует мнение, что если вы используете Scrum, то можете выпустить релиз только в конце спринта. Это не так. Scrum подразумевает, что в конце спринта должны быть готовы к релизу инкременты продукта. Но это не значит, что вы не можете производить релиз и во время спринта, — просто сделайте это частью критериев готовности (если это применяется повсеместно) или частью критериев приемки для конкретной пользовательской истории (если она имеет срочность выпуска).

Второй миф, с которым я часто сталкиваюсь: Scrum обязывает выпускать релиз в конце каждого спринта. Опять же это не так. Те, кто в это верит, должны осознать разницу между словами *выпуск релиза* и *готовность к выпуску релиза.* Scrum не говорит, что вы должны выпускать релиз в конце каждого спринта, он говорит, что вы должны сделать все возможное, чтобы иметь что-то готовое к выпуску релиза в конце спринта.

ПУТЕШЕСТВИЕ НАЧИНАЕТСЯ С МАЛЕНЬКОГО ШАГА

Просто начните с чего-нибудь. Знаю, что даже после прочтения такого короткого лайфхака, как этот, вам может показаться, что все это слишком сложно. И если вы это почувствовали, пожалуйста, осознайте, что при автоматизации хоть что-то лучше, чем ничего. Один юнит-тест лучше, чем ни одного. Автоматизированная сборка, занимающая 30 минут, лучше той, что отнимает часы ручной работы.

Если вы только внедряете автоматизацию, советую выделять на нее определенный процент времени в спринте. Начните с непрерывной интеграции. Затем автоматизируйте б*о*льшую часть сборок. Следом сосредоточьтесь на юнит-тестах для нового критически важного кода, а уже потом расширяйтесь на весь новый код. Следующим шагом может стать добавление к старому коду нескольких функциональных тестов.

Выбор за вами. Однако в любом случае начните с чего-нибудь. И помните: без автоматизации вы теряете время, хуже того, полагаетесь на людей, которые склонны ошибаться и чей ресурс ограничен по времени. Чтобы подкрепить утверждение Шора и Уордена о совершенстве, хочу закончить этот лайфхак классикой от Фредерика Брукса, автора книги «Мифический человеко-месяц»[72]:

Человеку не свойственно совершенство, и оно является необходимым лишь в немногих сферах его деятельности. Мне кажется, что при освоении программирования труднее всего привыкнуть к требованию совершенства.

ЗАКЛЮЧЕНИЕ

Три лайфхака этой главы посвящены выбору тактики, инструментам и советам, которые помогут вашей команде отслеживать дефекты и управлять ими в scrum-проекте. Давайте вспомним, что мы обсудили.

Лайфхак 16. Ба! Да это scrum-баг!

- отличие багов от ошибок;
- основополагающие принципы для работы с дефектами;
- подходы к отслеживанию дефектов и управлению ими внутри спринтов.

Лайфхак 17. Мы по-прежнему любим тестировщиков!

- эволюция роли тестировщика;
- почему квалифицированные тестировщики жизненно необходимы для высокоэффективных scrum-команд;
- ключевые функции, на которых должны фокусироваться тестировщики: консультирование, тест-дизайн и исследование.

Лайфхак 18. Нация автоматизации

- автоматизация как фактор, препятствующий скатыванию обратно к модели водопада;
- выбор отправных точек для автоматизации;
- Scrum и непрерывная поставка прекрасно работают вместе.

Глава 7

МОНИТОРИНГ И МЕТРИКИ

Поддержка scrum-проекта на плаву требует частой настройки и до-настройки. К счастью, основные процессы спринта позволяют регулярно отслеживать прогресс. Единичные наблюдения можно существенно дополнить комплексом из тщательно отобранных метрик, дашбордов и практик по синхронизации команды.

Три следующих лайфхака посвящены нескольким полезным методам и инструментам, помогающим оценить, насколько хорошо движется ваш проект.

Лайфхак 19 «Метрики, которые имеют значение» предлагает набор эффективных метрик, чтобы отслеживать текущий прогресс.

В лайфхаке 20 «Выдающиеся стендапы» я поделюсь с вами советами и приемами, благодаря которым ваши ежедневные scrum не превратятся в нескончаемую болтовню.

И наконец, из лайфхака 21 «Укрощаем доску задач» вы узнаете, как выжать максимум из визуализации работы всей команды.

ЛАЙФХАК 19. МЕТРИКИ, КОТОРЫЕ ИМЕЮТ ЗНАЧЕНИЕ

Вы начали работать, используя Scrum, и ваша команда вырвалась вперед, успешно завершая спринт за спринтом. Все идет отлично, пока не приходит какой-то умник из руководства и не говорит что-то вроде «Та-а-а-а-ак, в теории Scrum выглядит прекрасно, но какими метриками вы будете пользоваться, чтобы показать, насколько вы действительно эффективны?»

Нравится вам это или нет, но люди любят все измерять и сравнивать, поэтому от метрик вам не спрятаться, не скрыться. В этом лайфхаке я предложу вам несколько действительно полезных метрик, незаменимых при использовании Scrum.

ВИДЫ МЕТРИК

Когда речь заходит о метриках, у меня есть самый важный совет: используйте метрики во благо, а не во зло. Поскольку нет единых и общедоступных критериев, чтобы отличить полезные метрики от вредных, предлагаю систему, которой пользуюсь сам.

- **Полезные метрики** используются как маркер, чтобы помочь команде понять, где она находится. А также (и это гораздо важнее) как руководство по инспекции и адаптации процессов для непрерывного улучшения.
- **Вредные метрики** — бескомпромиссный инструмент для постоянного микроменеджмента личной эффективности каждого участника команды и, еще хуже, для наказаний и морального подавления.

Предлагаю разделить метрики по двум основным категориям:

1. Метрики для scrum-проекта (тема этого лайфхака).
2. Метрики при масштабировании Scrum (тема лайфхака 28).

ЧЕТЫРЕ ЗНАЧИМЫЕ МЕТРИКИ

Ниже я расскажу о четырех метриках для scrum-проекта, которые считаю наиболее полезными.

1. Диаграмма сгорания спринта.
2. Диаграмма сгорания релиза.
3. Диаграмма препятствий в спринте.
4. Диаграмма исправления ошибок.

Диаграмма сгорания спринта

Диаграмма сгорания спринта — это метрика для прогнозирования, помогающая отслеживать прогресс текущего спринта.

Как она создается

Диаграмма сгорания спринта строится следующим образом:

1. Для каждого дня в спринте рассчитайте суммарное время, остающееся для решения всех задач из бэклога спринта.
2. Проведите линию, соединив суммарное время текущего дня и суммарное время предыдущего дня (рис. 7.1).

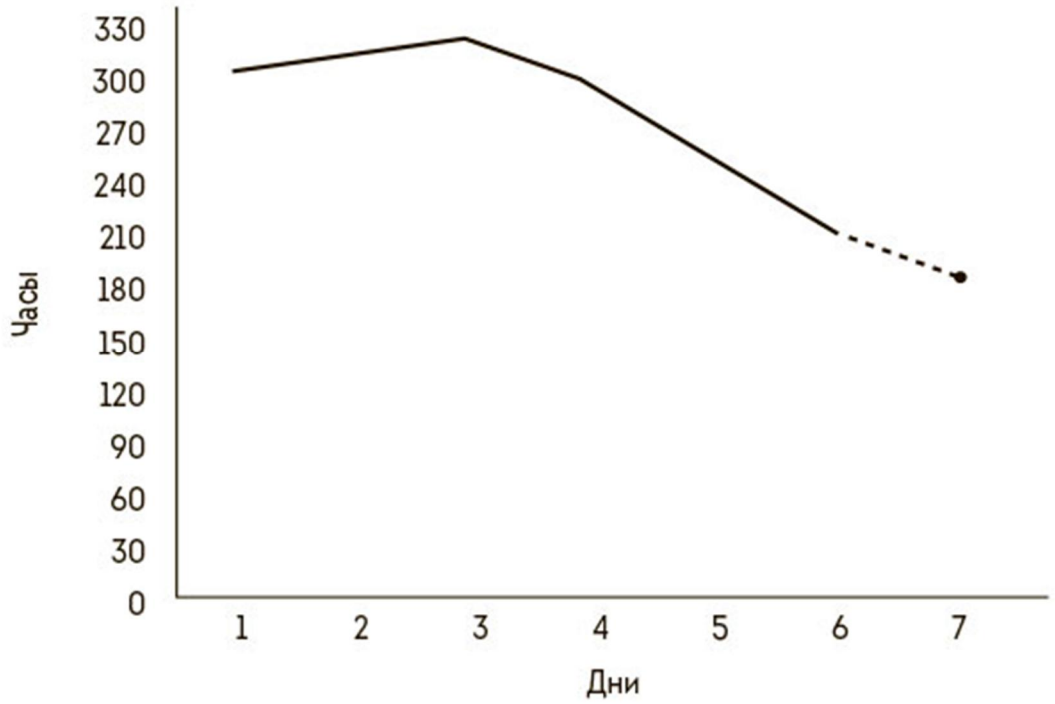

Рис. 7.1. Диаграмма сгорания спринта, обновляемая на ежедневной основе

Когда она создается

Диаграмму сгорания спринта генерируют в конце каждого дня спринта, кроме последнего. Последний день спринта отводится на обзор спринта, ретроспективу и планирование следующего спринта (см. лайфхак 8).

О чем она говорит

Диаграмма сгорания спринта служит ежедневным индикатором для scrum-команды, помогает управлять потоком задач и отслеживать прогресс. Если ваша диаграмма сгорания спринта похожа на рис. 7.2, значит:

1. В бэклог спринта были добавлены новые задачи, которые не рассматривались при планировании.
2. Некоторые задачи были оценены некорректно.
3. Кто-то из участников команды взял незапланированный отгул.
4. Появились помехи, препятствующие прогрессу.

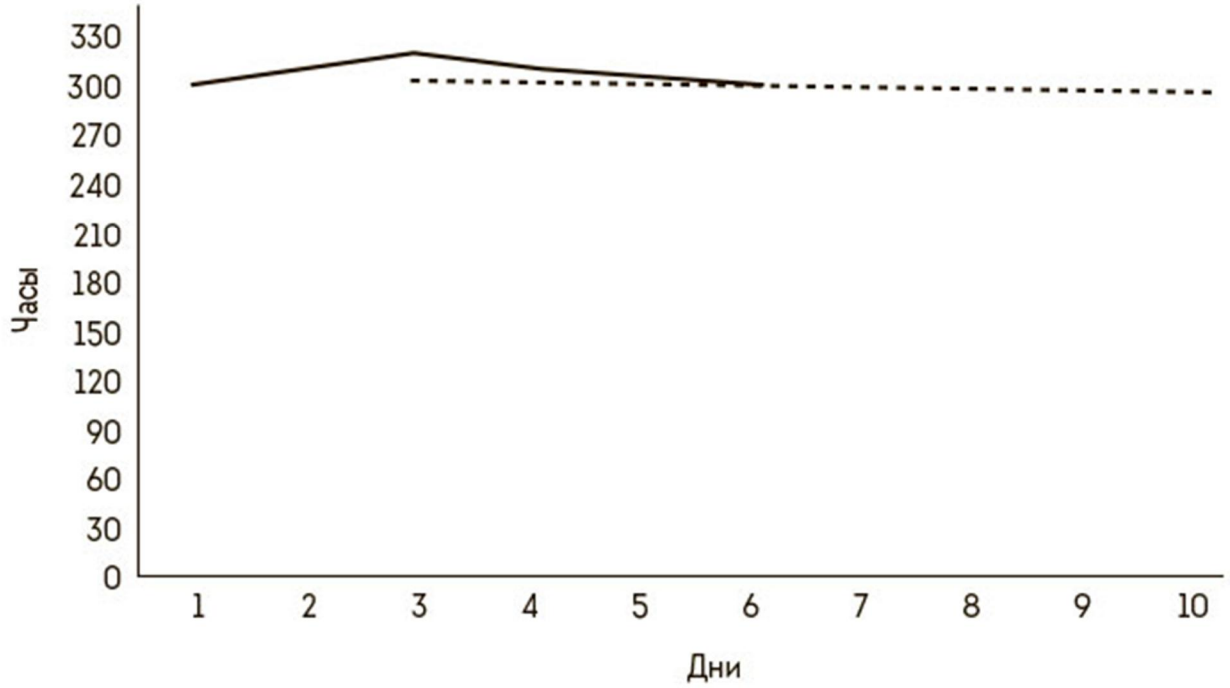

Рис. 7.2. Команда отстает от графика. Время поговорить с владельцем продукта о возможности снижения объема работ в спринте

Разумеется, все четыре фактора сразу могли оказать влияние и привести к задержке.

На планировании спринта многие команды рисуют «воображаемую» линию — диагональ от максимальных значений на оси Y до минимальных на оси X, которую используют как эталон для текущей линии сгорания задач. Я бы так не делал, ведь это может привести к неоправданному восприятию прогресса. Дело в том, что прогресс спринта редко следует за «иллюзорной» линией. Зачастую из-за непредвиденных обстоятельств линии сгорания растут первые несколько дней спринта. И только после того, как команда наберет обороты, устремляются вниз. «Воображаемая» линия на диаграмме сгорания спринта может ввести в заблуждение заказчика проекта, будто бы команда отстает всего лишь на день или на два.

Как можно на нее повлиять

Как быть, если диаграмма сгорания спринта показывает, что команда не достигнет целей спринта? Безусловно, вы будете делать все, что в ваших силах, для устранения препятствий (см. лайфхак 9). Кроме того, необходимо обсудить с владельцем продукта, какие работы исключить из текущего спринта. Если отклонения связаны с некорректной оценкой задач, анализ причин произошедшего поможет повысить точность планирования следующего спринта.

Хотите верьте, хотите нет, но диаграмма сгорания спринта может отражать и более радужную картину — когда линия тренда показывает, что команда завершит работу над бэклогом спринта раньше срока (см. рис. 7.3). Это ваш случай? Тогда такая диаграмма сгорания спринта должна простимулировать владельца продукта подготовить для взятия в спринт следующий элемент бэклога продукта (см. лайфхак 11), чтобы приступить к работе над ним до планирования следующего спринта.

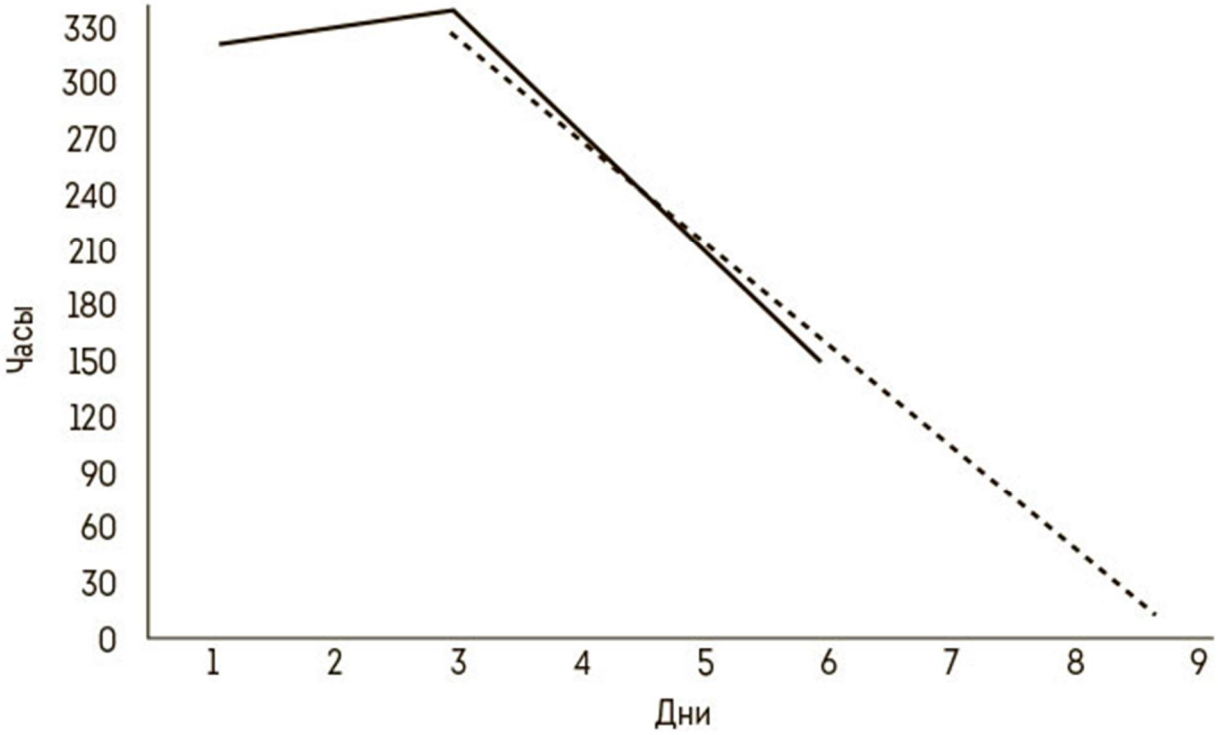

Рис. 7.3. Команда опережает график. Время поговорить с владельцем продукта о том, чтобы добавить в спринт дополнительный объем работ

Диаграмма сгорания релиза

На создание диаграммы сгорания релиза меня натолкнула «альтернативная scrum-диаграмма сгорания релиза» Майка Кона73.

Как она создается

Диаграмма сгорания релиза создается следующим способом:

1. Для каждого спринта рассчитайте сумму оставшихся поинтов для всех элементов продуктового бэклога, относящихся к следующему релизу.
2. Постройте линию тренда на основании данных пункта 1.
3. Для каждого спринта рассчитайте сумму поинтов всех элементов продуктового бэклога, добавленных после запуска проекта (если это применимо в вашем случае), и нанесите в виде отрицательных значений по оси Y.
4. Нарисуйте еще одну линию тренда на основании данных пункта 3 (см. рис. 7.4).

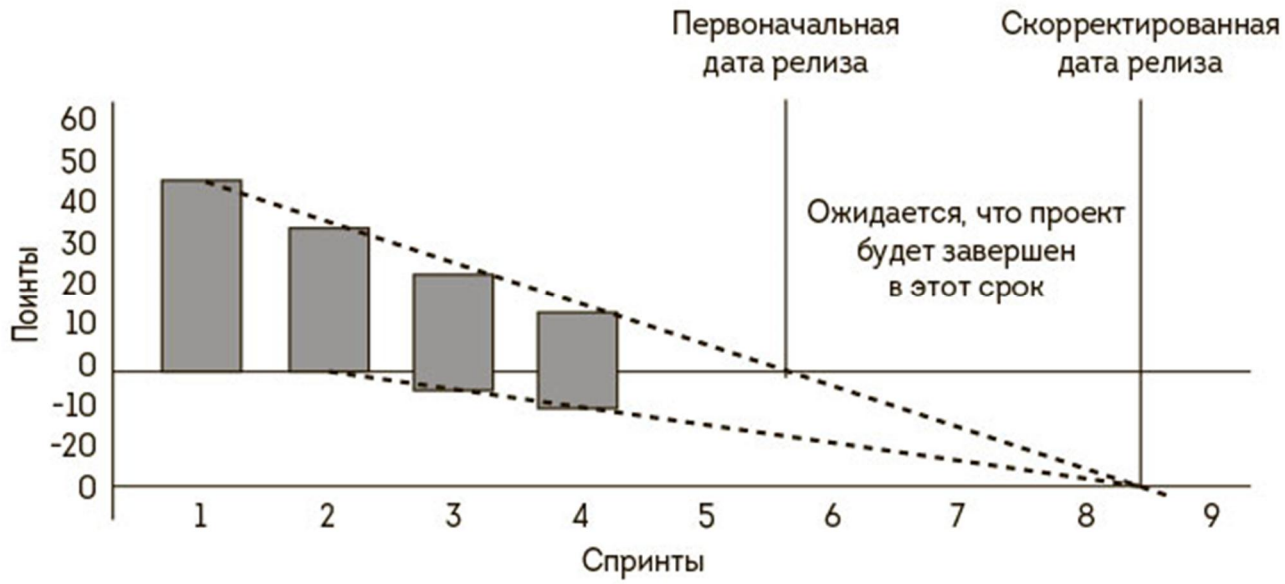

Рис. 7.4. Диаграмма сгорания релиза, которая обновляется в конце каждого спринта

Когда она создается

Диаграмма сгорания релиза создается в конце каждого спринта.

О чем она говорит

Эта метрика показывает, как меняется скорость прогресса команды в зависимости от скорости изменения объема работ. Точка, в которой пересекутся две линии тренда, покажет, сколько приблизительно спринтов потребуется для завершения релиза. Если линии тренда параллельны друг другу или расходятся, это грозный симптом того, что день релиза может никогда не настать (см. рис. 7.5).

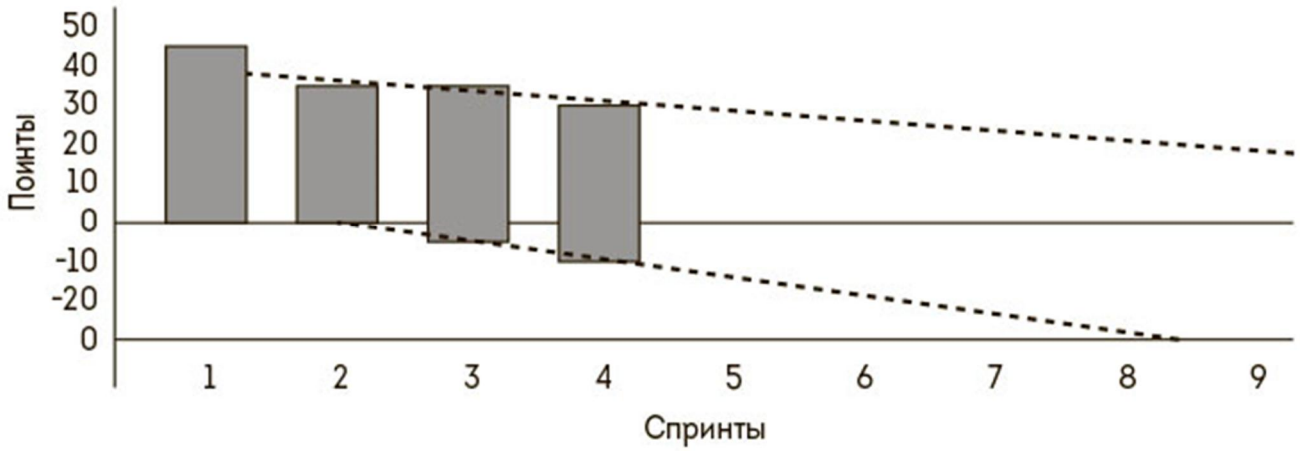

Рис. 7.5. Грозный симптом того, что релиз никогда не случится. Ваша задача — улучшить практики работы, устранить препятствия или уменьшить объем работ

Как можно на нее повлиять

Если две линии тренда не пересекаются или ожидаемая дата релиза является неприемлемой, нужно либо увеличить скорость прогресса (улучшения практики и/или устранения препятствия), либо уменьшить объем работ.

Диаграмма препятствий в спринте

Диаграмма препятствий в спринте — это метрика производительности, которая помогает команде оценить планируемый объем спринта.

Как она создается

Диаграмма препятствий в спринте создается следующим образом:

1. Для каждого спринта рассчитайте суммарное время, которое участник команды тратит на разработку задач, не связанных с бэклогом спринта.
2. Постройте линию тренда на основе данных из пункта 1 (см. рис. 7.6).

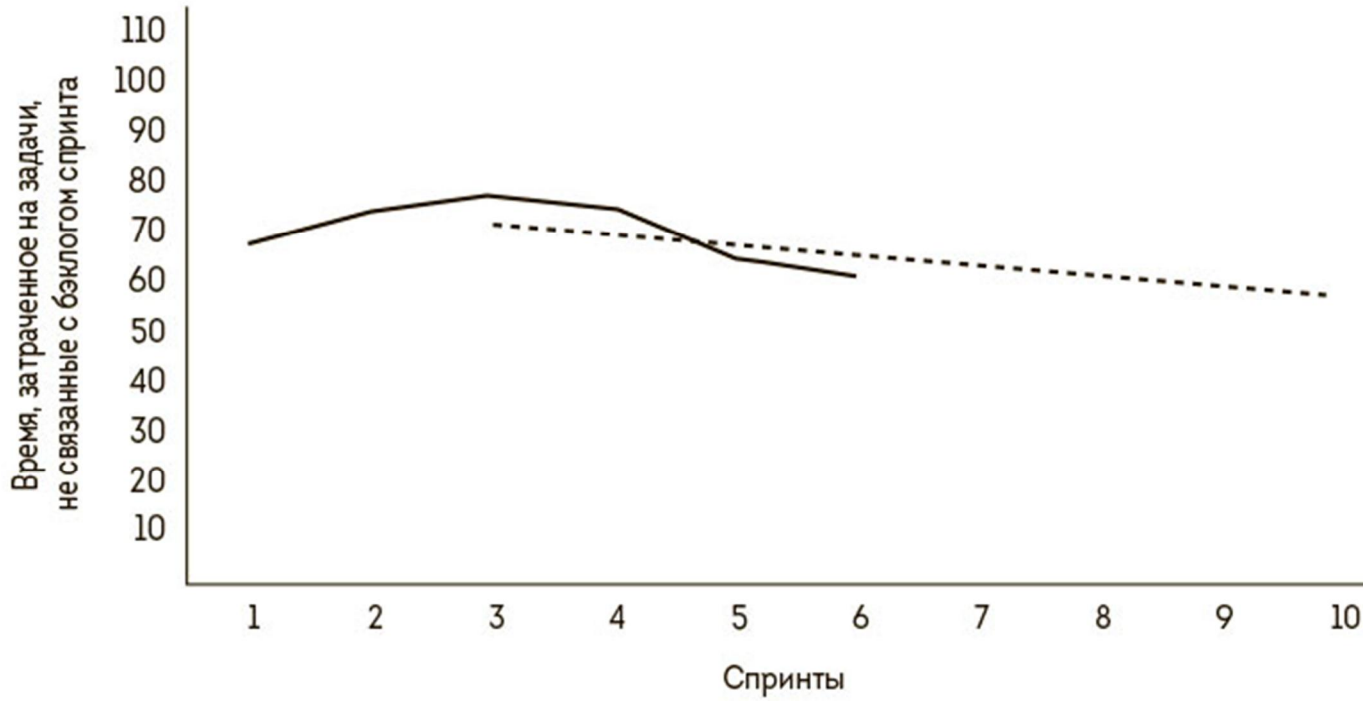

Рис. 7.6. Учитывайте линию тренда (на этом графике), оценивая объем спринта на следующем планировании

Когда она создается

Диаграмма препятствий в спринте создается в процессе планирования спринта.

О чем она говорит

С помощью визуализации времени, традиционно затрачиваемого на преодоление препятствий в спринте, можно оценить потенциальный объем дальнейших спринтов (время, которое должно быть выделено командой разработки на элементы бэклога спринта). Это особенно полезно, если ваш способ планирования основан на обязательствах (см. лайфхак 8).

Как можно на нее повлиять

В любом спринте будет ряд внешних организационных препятствий, которых вам не избежать. Диаграмма препятствий в спринте позволяет количественно оценить препятствия, а также определить, какие из них неотвратимы (например, встречи на уровне компании), а каких можно избежать (например, необходимость постоянно чинить ненадлежаще работающее оборудование).

Диаграмма исправления ошибок

Диаграмма исправления ошибок — количественная метрика, помогающая команде оценить, как много коллективных усилий было затрачено

на исправление багов по сравнению с работой над новыми требованиями.

Как она создается

Диаграмма исправления ошибок создается следующим способом:

1. Для каждого спринта рассчитайте общую скорость (сумма поинтов всех элементов бэклога спринта, включая новую функциональность и баги).
2. Для каждого спринта рассчитайте сумму поинтов, затраченных на работу с багами.
3. Проложите линию тренда для данных из пункта 2 (см. рис. 7.7).

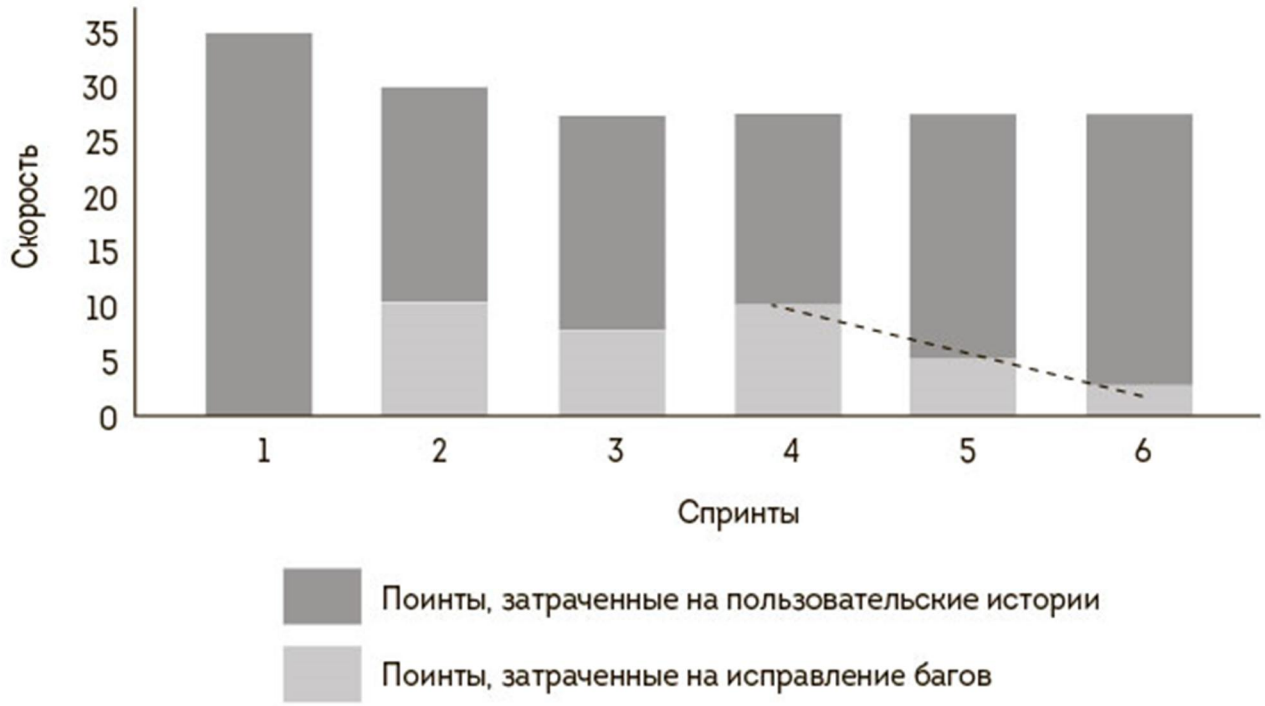

Рис. 7.7. Такой график показывает улучшение качества. Количество усилий, потраченных на исправление багов, уменьшается, при этом общая скорость команды не меняется

Когда она создается

Диаграмма исправления ошибок создается в конце каждого спринта.

О чем она говорит

Эта метрика позволяет отслеживать отклонения в качестве продукта, измеряя процентное соотношение работы над устранением багов и работы над новым функционалом в каждом спринте. Оценив структуру общей скорости (новый функционал и баги), можно получить дополнительные данные. Например, если общая скорость растет, на первый взгляд это хорошо.

Если одновременно растет объем работ, связанных с устранением багов, уровень качества падает (см. рис. 7.8). Таким образом, увеличение общей скорости работы может свидетельствовать о том, что команда просто стала быстрее исправлять баги, а это сомнительный комплимент.

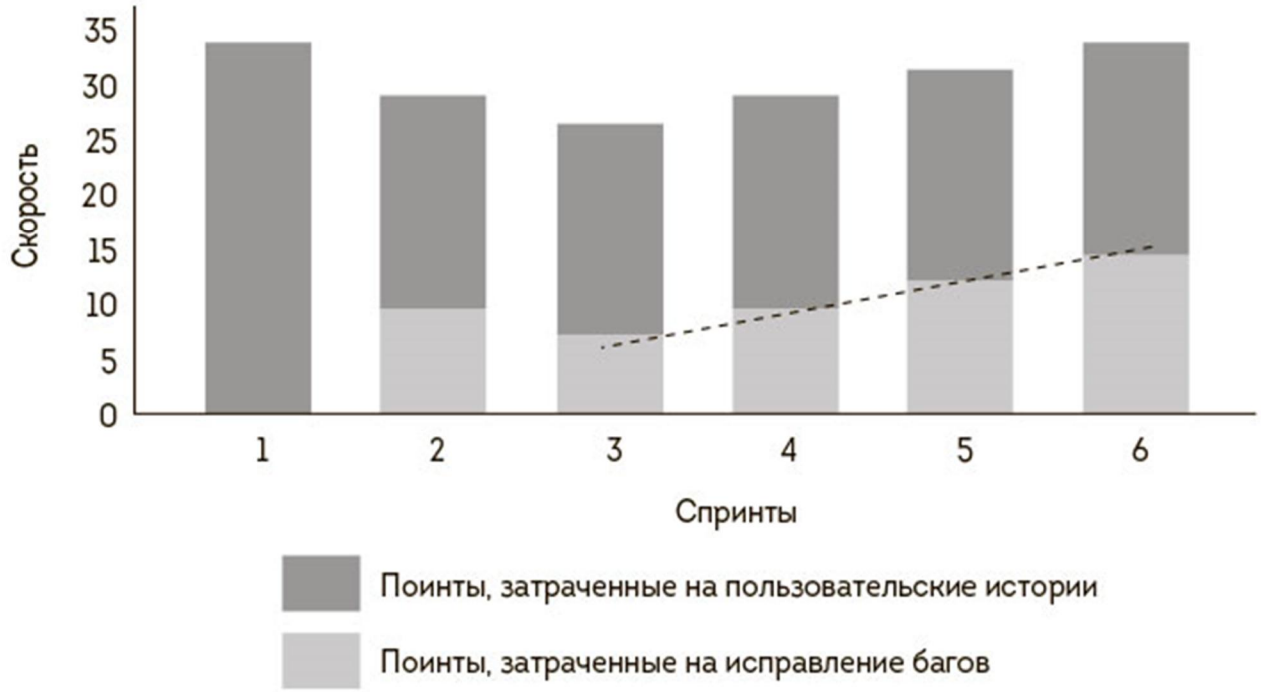

Рис. 7.8. Хотя общая скорость команды растет, это не очень хорошо, потому что качество падает. Время пересмотреть критерии готовности

Как можно на нее повлиять

Если время, затрачиваемое на устранение багов, растет, значит, текущий уровень качества является неудовлетворительным. Этот факт должен подвигнуть scrum-команду пересмотреть критерии готовности, чтобы ужесточить требования к качеству.

ОСТЕРЕГАЙТЕСЬ АНАЛИТИЧЕСКОГО ПАРАЛИЧА

Четыре предложенные метрики составляют небольшой набор потенциальных метрик, которые можно рассчитать и использовать в рамках одного scrum-проекта. Но будьте осторожны! Постоянное создание метрик может привести к той самой болезни, с которой Scrum как раз борется, — аналитическому параличу. И в завершение этого лайфхака отмечу, что очень важно применять метрики аккуратно и только с пользой, а не во вред. Пусть метрики станут вашими индикаторами, они помогут команде улучшить практики. Не используйте их для микроменеджмента и измерения индивидуальных показателей эффективности.

ЛАЙФХАК 20. ВЫДАЮЩИЕСЯ СТЕНДАПЫ

Спросите любого в отделе продаж, что такое Scrum, и помимо упоминания о доске задач с разноцветными стикерами (см. лайфхак 21), скорее всего, он вспомнит про ежедневный scrum, также известный как стендап. Ежедневный scrum — это пульс команды. А пульс здорового человека должен быть ровным, устойчивым и ритмичным. Этот лайфхак даст несколько подсказок, как улучшить стендапы, чтобы они работали как часы.

КОГДА И ГДЕ?

Прежде всего я настоятельно рекомендую проводить стендапы на ногах, а не сидя. Это тонкое, но важное отличие. Общение стоя помогает команде зарядиться энергией на весь день. А также гарантирует, что встреча пройдет живо и по делу — чтобы не разболелись ноги.

Я не сторонник навязывать команде продолжительность стендапа. С этим команда должна определиться сама. Однако вам стоит побудить каждого взять ответственность за такое время, которое будет работать в интересах команды, — настолько рано, насколько это возможно. Конечно, если ваша команда распределенная, ежедневный scrum следует назначать с учетом часовых поясов.

После того как вы договорились о времени, пора применить несколько советов по проведению стендапа. Вот мои любимые:

1. Ежедневный scrum должен начинаться вовремя: опаздывающих не ждите.
2. Любой, кто опоздал без предварительного предупреждения либо без веского или очень забавного извинения (которое вызовет смех у каждого участника команды), должен сделать финансовый взнос в банку опозданий. Эти суммы пойдут на согласованную командой благотворительность.
3. Ежедневный scrum должен выглядеть как правильный круг или полукруг вокруг доски задач, а не как аморфная клякса (см. рис. 7.9).

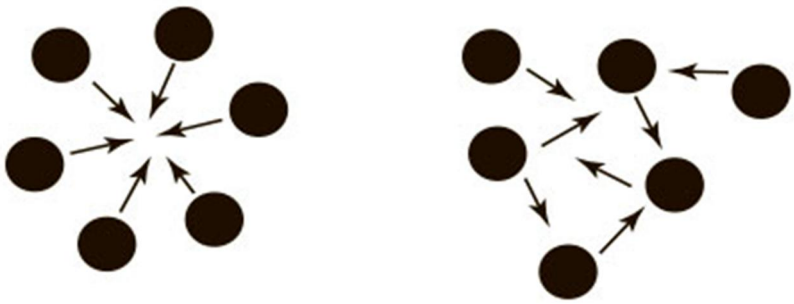

Рис. 7.9. Ежедневный scrum должен походить на правильный круг (или полукруг), а не на аморфную кляксу

КАКИЕ ВОПРОСЫ НАДО ОБСУДИТЬ?

Вот типичные вопросы, на которые каждый участник команды разработки отвечает во время ежедневного стендапа:

1. Чего я достиг вчера?
2. Чего я надеюсь добиться сегодня?74
3. Есть ли у меня препятствия или затруднения?

Хотя эти вопросы кажутся простыми, я дам вам несколько подсказок, как сделать ежедневный scrum более эффективным.

Во-первых, каждый, рассказывая о том, что было сделано и что он намеревается сделать, должен говорить только про те задачи, которые есть на доске задач. Это помогает поддерживать доску задач в актуальном состоянии (если ее забыли обновить).

На самом деле участник команды должен успеть ответить на эти три вопроса за 30 секунд. Однако по некоторым обновлениям текущего статуса задач могут завязаться дискуссии, которые утянут команду в черную дыру дебатов (пока каждый не поймет, что его ноги болят после получасового стендапа). Ежедневный scrum запросто может скатиться к обсуждению препятствий и их детальному разбору. И если вы понимаете, что команда отклоняется от основной темы, вам следует вмешаться и произнести фразу «в рабочем порядке». Другой вариант, если вы хотите действовать мягче и вежливее, — просто медленно поднимите руку. Этим жестом вы покажете команде, что, хотя понимаете важность дискуссии, тем не менее предлагаете пройтись по всем обновлениям, и только потом, в конце стендапа, команда сможет продолжить обсуждение.

GIFTS («подарки»)

Agile-консультант и блогер Джейсон Йип объясняет ключевую цель стендапа, используя акроним GIFTS75. Я люблю этот акроним, потому что его легко запомнить.

G — Good start (хорошее начало). Хорошее начало означает, что стендап должен давать команде энергию, а не забирать ее. Энергия приходит от общности цели и ее срочности.

I — Improvement (улучшение). Мы не можем решить проблемы, о которых не знаем, поэтому большая часть стендапов посвящена выявлению проблем, преодоление которых ведет к улучшению. Иными словами, улучшения — это не только про решение проблем, но и про обмен лучшими практиками и идеями.

F — Focus. Стендапы должны помогать команде держать фокус на выполнение рабочих задач ради достижения целей, а не поощрять бессмысленную активность.

T — Team (команда). Эффективную команду можно построить только с помощью регулярных коммуникаций, систематической работы

и помощи друг другу. Все это тесно связано с внутрикомандной поддержкой, когда участники делятся информацией о трудностях, с которыми столкнулись.

S — Status. Статус — это ответ на пару вопросов:

1. Как продвигается работа?
2. Есть ли еще что-нибудь, что необходимо знать команде?

НЕСКОЛЬКО КОМАНД

Над проектом могут параллельно работать несколько scrum-команд, интерфейсы у которых (как технические, так и коммуникационные) общие. В подобных случаях проводят так называемые стендапы Scrum of Scrums. На этом дополнительном стендапе присутствуют представители каждой команды.

Это хороший вариант, но я предпочитаю отправлять представителей команд на чужие стендапы в качестве наблюдателей — чтобы собрать информацию о потенциальных конфликтах или возможных рисках (см. рис. 7.10). Благодаря этому минимизируются разрывы в коммуникациях. Наблюдателями я выбираю старших разработчиков (в каждой команде). Если вы применяете этот подход, очень важно согласовать время ежедневного scrum для каждой команды, чтобы их представители могли посещать все scrum.

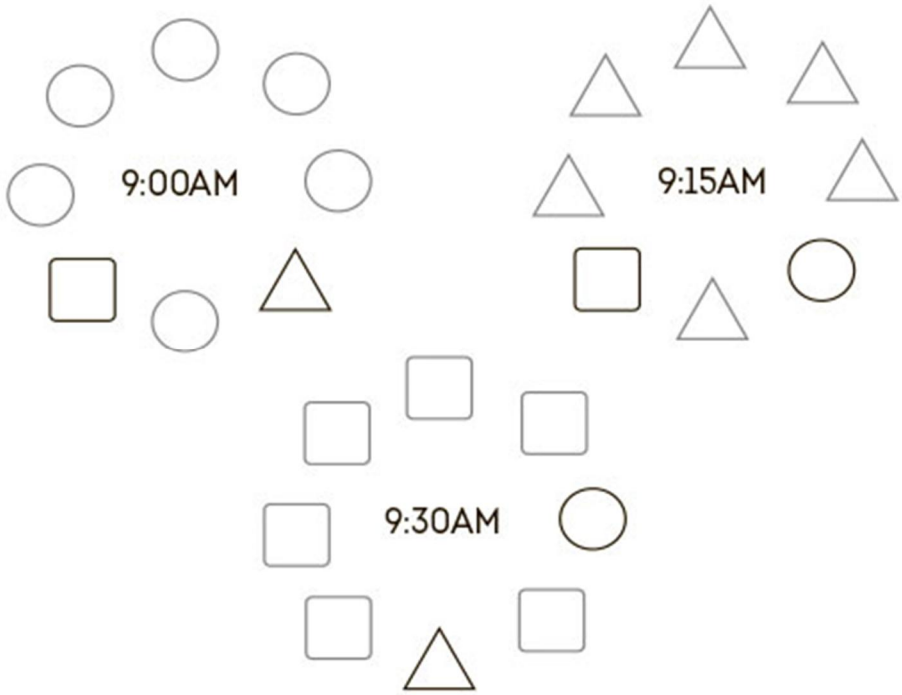

Рис. 7.10. Синхронизируйте стендапы всех команд, чтобы их представители присутствовали на каждом

ИГНОРИРУЙТЕ SCRUM-МАСТЕРА

Мне нравится способ, который предложил Кен Швабер[76]. С его точки зрения, стендапы — это возможность для команд кратко «пообщаться и синхронизироваться». Но это не перекличка или сеанс микроменеджмента, когда команда отчитывается scrum-мастеру и/или владельцу продукта. Я часто вижу, что у некоторых участников команды вошло в привычку адресовать новости только scrum-мастеру. Если вы заметили, что участник команды обращается на стендапе только к scrum-мастеру и смотрит исключительно на вас, вот вам совет: начинайте медленно отворачиваться или смотрите в потолок. Подавая такой вот сигнал, можно быстро избавить команду от этой привычки.

НЕСКОЛЬКО ДОПОЛНИТЕЛЬНЫХ ШТРИХОВ

До начала стендапа я рекомендую воодушевить команду, только не переусердствуйте — добавьте остроумное замечание или шутку. Всегда приятно начать день на позитивной ноте, чувствуя прилив энергии, а не ощущая себя на армейской поверке.

Также я люблю использовать ежедневный scrum как церемонию наказания за упавшие сборки. Команда может тренировать мускулы креативности, определяясь со своим отношением к виновнику произошедшего (см. рис. 7.11). Вариантов бывает несколько — от относительно безобидных (как монитор, на котором мигает имя виновника) до экстремальных (буханка заплесневелого хлеба на мониторе нарушителя[77] — это было легко, пока не появились плоские мониторы!). Мои предпочтения располагаются где-то посредине этой линейки — когда виновнику вручается акубра (австралийская ковбойская шляпа), которую тот должен носить в течение всего дня, включая обед.

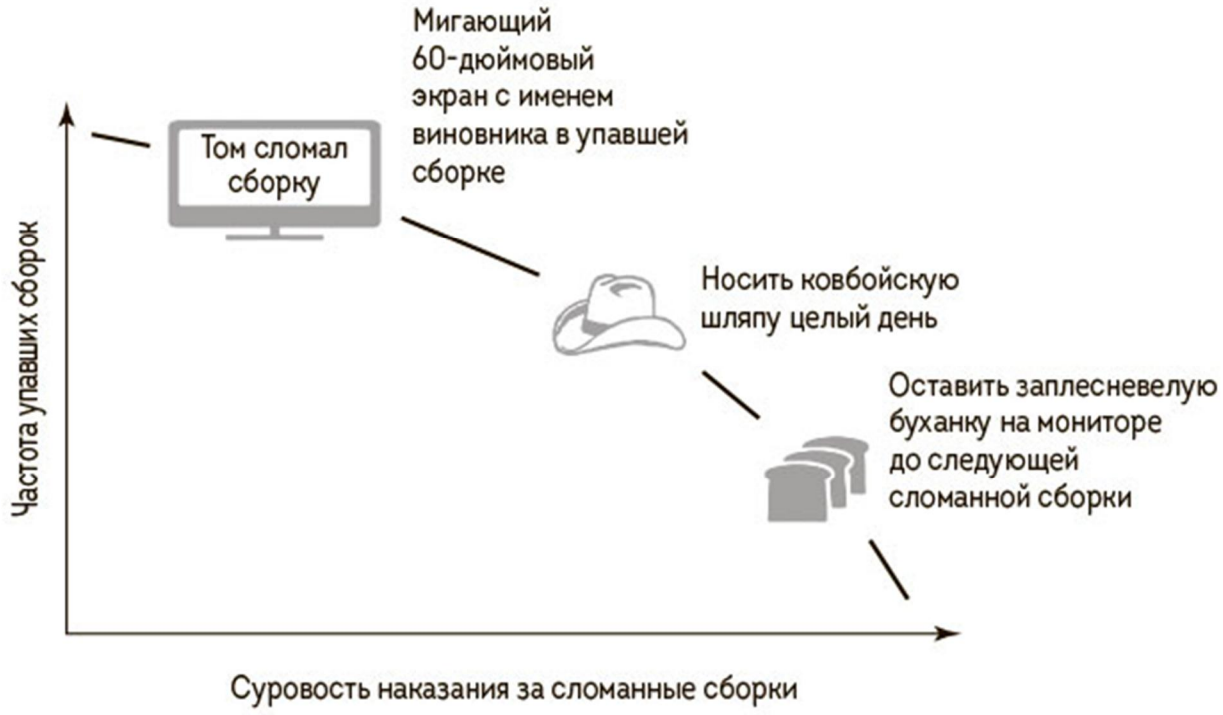

Рис. 7.11. Наказания за упавшую сборку могут варьироваться от символических до самых жестких

Несмотря на то что ежедневный scrum проходит по стандартной схеме, я люблю держать всех в напряжении, добавляя элементы случайности. Так, вы можете использовать маленький футбольный мячик, который пасуется от одного члена команды к другому в случайном порядке. Любой, кто не слушает (или просто плохо играет), смутится, пропустив мяч между ног.

ЕЖЕДНЕВНЫЙ SCRUM ЖДЕТ БОЛЬШОЙ УСПЕХ

У ежедневного scrum большое будущее. Наряду с доской задач это один из самых популярных элементов Scrum, который я то и дело встречаю в командах, не связанных с IT. Даже Wall Street Journal[78] рассказывал про растущую популярность стендапа.

Ежедневный scrum — это очень простая практика, но, если пустить его на самотек, он легко превратится в ежедневный бардак. Используйте советы из этого лайфхака, чтобы избежать хаоса, и никогда не забывайте закон Конвея: «Организации, проектирующие системы… ограничены дизайном, который копирует структуру коммуникации в этой организации»[79].

ЛАЙФХАК 21. УКРОЩАЕМ ДОСКУ ЗАДАЧ

Для тех, кто глубоко не погружен в Scrum, самый узнаваемый элемент любого scrum-проекта — командная доска задач. Центральный элемент Scrum и точка сбора для всей команды, эта доска с разноцветными стикерами уже стала символом Scrum. С популярностью начинаются модификации, и в наши дни больше нет окончательной конфигурации и единой формы доски. Хотя не существует правильного или неправильного способа организовать этот центральный элемент Scrum, тем не менее у меня есть твердое убеждение на этот счет.

ЦИФРОВАЯ ИЛИ ФИЗИЧЕСКАЯ?

Я был немного озадачен, впервые увидев физическую scrum-доску, на которой использовались бумажные стикеры, маркеры и кнопки. Почему же мы, сорвиголовы высоких технологий, откатываемся назад в темные века и используем такой доисторический инструмент вместо красивых ярких больших мониторов, которые сами строят скользящие диаграммы? Что ж, ответ прост: психология человека. Создатели доски задач знали, что делали, выбрав простую доску своим стандартом.

Есть нечто приятное в том, чтобы встать, подойти к доске (поверьте, это не очень тяжелое упражнение), отклеить стикер и перенести в колонку «Сделано». Полагаю, это действие созвучно нашей естественной

потребности в достижениях. Удовлетворить ее помогает визуализация завершенной работы — особенно для сферы деятельности, в которой большая часть сложной работы невидима для окружающих. К тому же рабочее пространство стоит сделать более ярким и мотивирующим. Как вы считаете?

МАТЕРИАЛЫ ДЛЯ ПРИВЕРЖЕНЦЕВ СТАРОЙ ШКОЛЫ

Чтобы создать физическую доску задач, вам понадобятся:

- большая маркерная доска, стена или кусок стекла;
- синий скотч (чтобы создать колонки);
- большая линейка (для ваших строк);
- маркер для белой доски (тоже для ваших строк);
- стикеры (двух цветов).

РИСУЕМ СТОЛБЦЫ

Можно использовать любые виды столбцов, но я предпочитаю следующий порядок:

Не начато / В работе / Готово к проверке / Сделано

СТРОКИ СТИКЕРОВ

Строки представляют элементы бэклога спринта, включая связанные с ними задачи, на которых нужно сфокусироваться в течение спринта.

Не используйте скотч для строк (только для колонок), ведь строки будут меняться от спринта к спринту, а переклеивание скотча каждые две недели начнет раздражать.

Каждая строка посвящена какому-то одному элементу бэклога и связанным с ним задачам. Каждый стикер — конкретная задача. Постарайтесь «нарезать» элементы бэклога на задачи «вертикально» — на самостоятельно тестируемые куски. В противном случае, когда задачи будут располагаться друг за другом, столбец «Готово к проверке» утратит смысл. Более подробно я рассказывал о разбивке элементов бэклога на задачи в лайфхаке 8.

СОДЕРЖАНИЕ СТИКЕРА

Если вы используете программное обеспечение для управления scrum-артефактами, то знаете, как тонка грань между дублированием данных (уже зафиксированных в цифровом виде) и потерей времени на это, с одной стороны, и недостатком данных на стикерах, с другой стороны. Фокус в том, чтобы написать на стикере ровно столько информации, сколько необходимо для его идентификации. Советую фиксировать:

- идентификационный номер задачи (который автоматически генерируется программой);
- инициалы участника команды, взявшего на себя задачу;
- несколько слов, описывающих задачу;
- оставшееся текущее время работы над задачей (см. рис. 7.12).

Рис. 7.12. Пример содержания стикера

Любая незапланированная задача должна быть отражена на доске задач. Для этого я рекомендую использовать стикер другого цвета (см. лайфхак 12). Так вы сможете четко определить потенциальные области улучшения при планировании спринта.

СОЗДАЕМ ДИАГРАММЫ СГОРАНИЯ СПРИНТА

Я поклонник диаграммы сгорания спринта (см. лайфхак 19). Если вы используете специальное программное обеспечение для Scrum, то сможете автоматически создавать ее на ежедневной основе. Несмотря на это, я предпочитаю обновлять диаграмму сгорания спринта вручную. Продлить линию на один день проще, чем продублировать еще одну диаграмму. К тому же мне нравится сам процесс: я наслаждаюсь обновлением диаграммы сгорания в начале каждого стендапа. Благодаря этому команде не терпится узнать, что дальше.

НЕКОТОРЫЕ ВАЖНЫЕ ДЕТАЛИ

Есть несколько других артефактов, которые целесообразно разместить на стене рядом с доской задач, распечатав крупным ярким шрифтом (см. рис. 7.13).

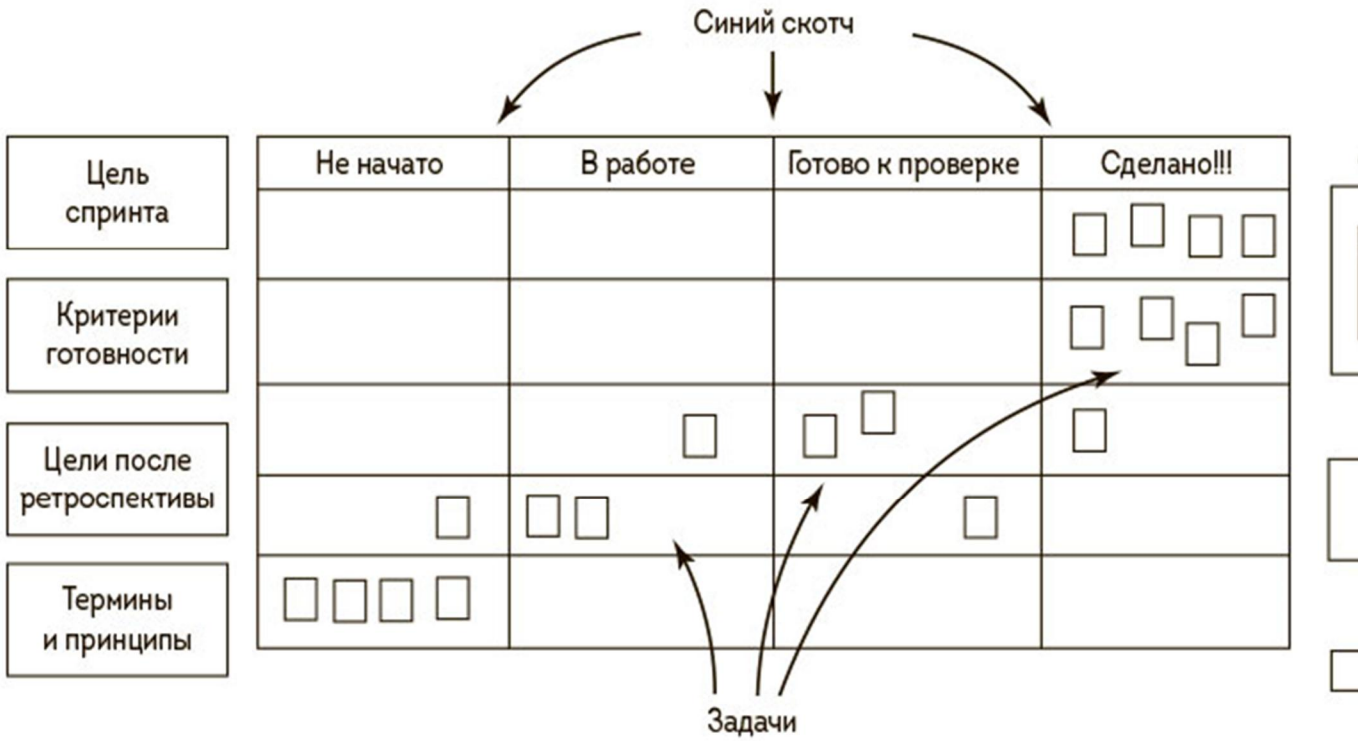

Рис. 7.13. Доска задач во всем ее великолепии с некоторыми важными деталями

Цель спринта

Цель спринта (лайфхак 8) — это ориентир для команды. Было бы упущением не поместить ее на самом видном месте.

Цели по итогам ретроспективы

После ретроспективы спринта команда должна определить приоритетные направления оптимизации процессов, чтобы сфокусироваться на них в предстоящем спринте (см. лайфхак 23). О них легко забыть, когда команда глубоко погружена в работу над текущим функционалом. Но если оставить их без внимания, команда попадет в порочный круг, когда непрерывное усовершенствование постоянно отодвигается на второй план. Я видел команды, использующие отдельную доску исключительно для задач с ретроспективы, но я рекомендую размещать их на стене рядом с основной доской.

Термины и принципы

На ранних стадиях, когда команда только начинает работать по Scrum, ей предстоит усвоить много новой информации. Помните: даже если мы перевернули знакомые процессы и привычные термины с ног на голову, мышление не меняется в одночасье. Чтобы побороть старые привычки,

очень полезно визуализировать новые понятия, такие как препятствия (см. лайфхак 9), баги и ошибки (см. лайфхак 16).

НЕ ТЕРЯЙТЕ ОЩУЩЕНИЕ РЕАЛЬНОСТИ!

На самых ранних этапах работы над проектом, до того как разные пользовательские истории трансформируются и объединятся в гармоничный, грамотно упакованный конечный продукт, команда может время от времени терять из виду общую картину и глобальную цель. Несколько советов, приведенных ниже, помогут сосредоточиться на результате.

Макеты высокого уровня

Во многих случаях еще до первого спринта владелец продукта разработал макеты ключевых пользовательских интерфейсов (это могут быть даже эскизы, нарисованные от руки). Независимо от того, насколько эти наброски предварительны, отличная идея — держать их копию рядом с доской задач, чтобы напоминать о проекте в целом.

Отзывы клиентов

Если только ваша команда не разрабатывает свой первый продукт в стартапе, у вас уже есть пользователи и они ценят то, что вы делаете, — в противном случае вы бы сидели дома, играя в компьютерные игры. Очевидно, периодически они дают вам обратную связь, говоря о том, что им нравится или не нравится в вашем продукте. Вы можете услышать: «Мы очень любим простоту вашего продукта по сравнению с продуктом ABC». Или: «Продукт XYZ работает быстрее по сравнению с вашими конкурентами». Чтобы обеспечить себе неизменный успех, держите отзывы пользователей перед глазами. Как часто вы встречали отличные продукты, которые теряли свои преимущества только потому, что отходили от тех ключевых элементов, которые делали их популярными? Знаю из личного опыта, что изучение пользовательских отзывов стимулирует более тщательное рассмотрение вопросов, когда владелец продукта или стейкхолдеры предлагают повесить дополнительный колокольчик или добавить какую-нибудь приблуду.

ВРЕМЯ ДЛЯ ВЕЧЕРИНКИ!

После особенно тяжелого спринта вам может быть чертовски приятно выкинуть все эти стикеры в ближайшую корзину для мусора. Если таким образом вы выпускаете пар, так и поступайте. Однако есть другое предложение: сохраните стикеры (в глубине какого-нибудь ящика) и используйте для вечеринки по случаю релиза, чтобы предаться ностальгии (или, возможно, посмотреть на них с отвращением)!

Хотя доска задач может восприниматься исключительно как agile-инструмент, я часто обсуждаю ее эффективность с руководителями из таких разных областей, как юриспруденция или школьное образование. Это вселяет уверенность: осталось совсем немного, и цветные доски покорят мир!

ЗАКЛЮЧЕНИЕ

В трех лайфхаках этой главы мы рассмотрели тактику, инструменты и советы, которые позволят вам оценить, как продвигается проект.

Лайфхак 19. Метрики, которые имеют значение

- разница между полезными и вредными метриками;
- выбор значимых метрик: диаграмма сгорания задач спринта, диаграмма сгорания задач релиза, диаграмма препятствий в спринте и диаграмма исправления ошибок;
- как избежать избыточности метрик.

Лайфхак 20. Выдающиеся стендапы

- организация ежедневного scrum: когда, где и как проводить;
- варианты синхронизации нескольких команд;
- как работать с типичными проблемами ежедневного scrum.

Лайфхак 21. Укрощаем доску задач

- факторы, которые нужно учесть при выборе между цифровой и физической доской задач;
- как организовать физическую доску задач;
- дополнительные артефакты, которые украсят вашу доску задач.

Глава 8

РЕТРОСПЕКТИВЫ, ОБЗОРЫ СПРИНТА И РИСК

К счастью, дни обременительных «разборов полетов», проводимых в самом конце проекта — когда они наименее полезны, — давно позади! Вместо них scrum-проекты предлагают возможности для частой инспекции и адаптации как продукта, так и процесса в конце каждого спринта, чтобы убедиться, что риски минимальны, а все уроки выучены вовремя.

Три лайфхака этой главы содержат рекомендации по максимально эффективному проведению ваших регулярных обзоров спринта и ретроспектив.

Лайфхак 22 «Чек-лист для обзора спринта» фокусируется на тех шагах, которые помогут вам убедиться, что ваша команда правильно оценивает себя во время открытого обзора спринта.

Лайфхак 23 «Ретроспектива при любых обстоятельствах» предлагает две эффективные техники проведения ретроспективы, а также раскрывает ключевые области, на которых следует сосредоточиться на этой встрече в конце спринта.

Лайфхак 24 «Рисковать и ошибаться» расскажет о том, почему важно создать такую рабочую среду, в которой будут терпимо относиться к ошибкам, чтобы сформировать культуру открытости и инноваций.

ЛАЙФХАК 22. ЧЕК-ЛИСТ ДЛЯ ОБЗОРА СПРИНТА

Предупреждение: на первый взгляд обзор спринта кажется очень простым и понятным. Но имейте в виду: без тщательной подготовки эта встреча может породить хаос, когда все стучат кулаками по столу, а слезы льются рекой.

Просто покажите стейкхолдерам, что вы сделали за последний двухнедельный спринт, — разве в этом есть какая-то сложность? Что ж, основываясь на собственном опыте, могу сказать: обзор спринта редко проходит легко. На самом деле это наиболее сложная встреча, и ее нужно фасилитировать.

Основная сложность — выровнять ожидания разных групп внешних стейкхолдеров. Часто эти люди занимают более высокое положение в бизнес-среде, чем участники команды, и, конечно же, менее погружены в проект. Не хотелось бы обобщать, но нередко их способность концентрировать внимание не отличается от поведения щенка с синдромом дефицита внимания!

ВО ВРЕМЯ ПЛАНИРОВАНИЯ СПРИНТА

Подготовка к обзору спринта на самом деле начинается еще на планировании спринта (см. рис. 8.1). При планировании спринта команда должна убедиться, что зарезервировала время на подготовку к обзору:

- подготовка основной информации для демо;
- подготовка сценария демо;
- проверка оборудования для демо.

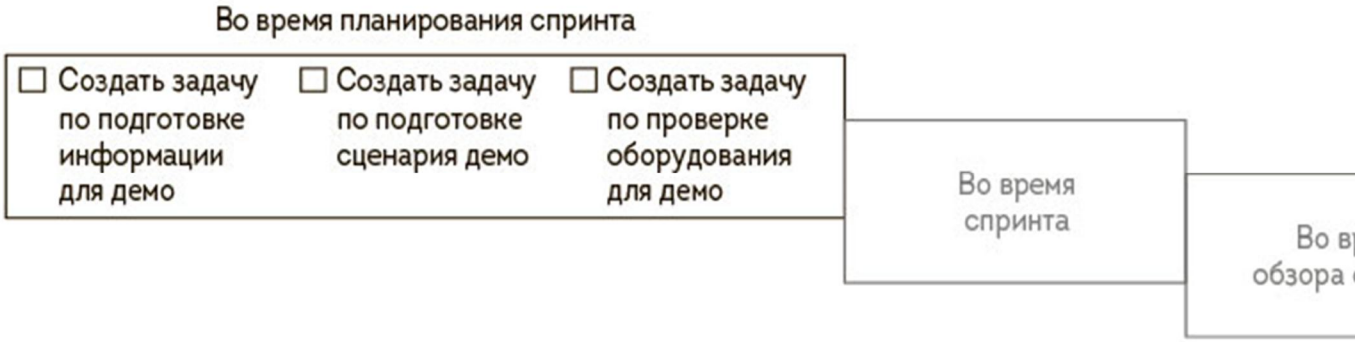

Рис. 8.1. Подготовка к обзору спринта начинается на планировании спринта

Разумеется, вы не хотите, чтобы команда потратила слишком много времени на эти задачи, а обзор спринта превратился в показуху, тем не менее эти задачи нельзя не учитывать.

Есть еще несколько аспектов, на которые нужно обратить внимание команды на планировании спринта. Во-первых, демо на обзоре спринта должно проводиться в симуляционной среде80, а не на сервере разработки (и определенно не на индивидуальном компьютере разработчика). Цель демо — не пустить пыль в глаза, а удостовериться, что продукт готов к релизу.

Во-вторых, хотя команда должна чувствовать себя расслабленно во время обзора спринта, к нему следует подходить со всей серьезностью. Нужно показать эффективность как самой команды, так и Scrum — особенно тем, кто, возможно, не имеет представления ни о том ни о другом.

ВО ВРЕМЯ СПРИНТА

Во время спринта scrum-мастер и владелец продукта должны пройти по совместному чек-листу в преддверии обзора спринта (см. рис. 8.2). Вопросы касаются места проведения обзора спринта, приглашений, участников с болезненными реакциями, докладчиков и ожиданий.

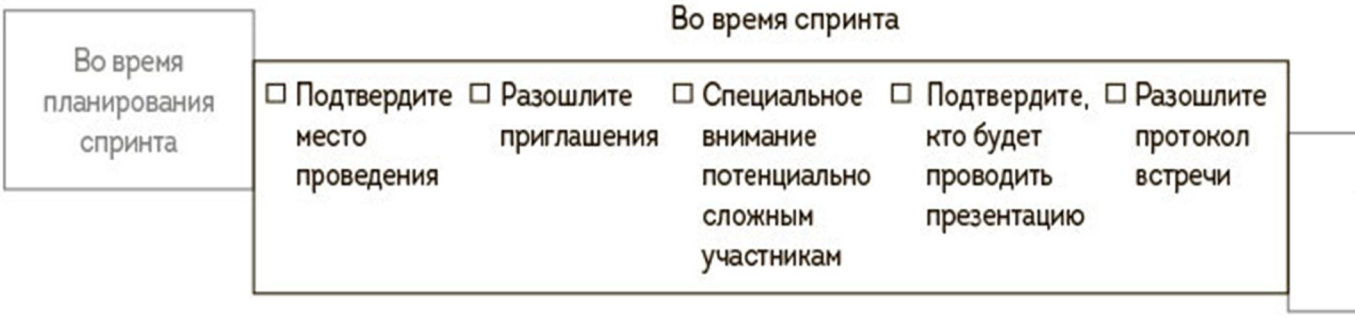

Рис. 8.2. В процессе спринта scrum-мастер и владелец продукта должны пройти по совместному чек-листу в преддверии обзора спринта

Место проведения

Подтвердите место проведения обзора спринта. Убедитесь, что все переговорные забронированы и оснащены всем необходимым. Дважды проверьте проектор, сетевые подключения, наличие белой маркерной доски и достаточного количества мест для участников. Нет ничего прискорбнее, чем так называемые эксперты, не способные настроить проектор даже после того, как встреча уже началась.

Приглашения

Убедитесь, что приглашения всем стейкхолдерам, чье присутствие обязательно, отправлены заранее. Это касается только тех, кто не является постоянным участником обзора спринта. Тем же, кто участвует регулярно, приглашения должны приходить на постоянной основе.

Сложные участники

У вас могут возникнуть опасения из-за стейкхолдера, имеющего привычку всегда находиться в центре внимания, высказывать необоснованную критику или комментировать происходящее не по делу. Если и вы столкнулись с таким восхитительным персонажем, рекомендую scrum-мастеру и/или владельцу продукта встретиться с ним до обзора спринта, чтобы провести предварительный просмотр и получить раннюю обратную связь. Люди, которым дали почувствовать себя особенными, непохожими на других, скорее всего, не станут вести себя деструктивно во время обзора спринта. В любом случае мы не хотим подделывать результаты, но нам важно избежать деморализующих встреч.

Выступающие

Ближе к концу спринта, когда вы определите, какие конкретно пользовательские истории будут продемонстрированы, убедитесь, что команда выбрала того, кто будет проводить демонстрацию. Выступить может один участник команды либо же группа — здесь нет никаких раз и навсегда утвержденных правил.

Ожидания

После того как определены истории для демонстрации, я рекомендую отправить письмо всем участникам обзора спринта, чтобы ознакомить их с кратким протоколом встречи. На ранних этапах проекта на обзоре спринта у вас мало возможностей показать готовый функционал с привлекательным пользовательским интерфейсом, что может вызвать у стейкхолдеров разочарование и раздражение. Вот почему информируя их о том, что предстоящий обзор спринта будет сфокусирован на технических работах, вы тем самым снижаете градус их ожидания.

Это письмо также служит дружеским напоминанием о том, когда и где будет проходить встреча, и вежливым предупреждением, что опоздавшим придется купить всем кофе.

ВО ВРЕМЯ ОБЗОРА СПРИНТА

Перед обзором спринта (см. рис. 8.3) вспомните следующие советы. Они помогут вам провести встречу гладко.

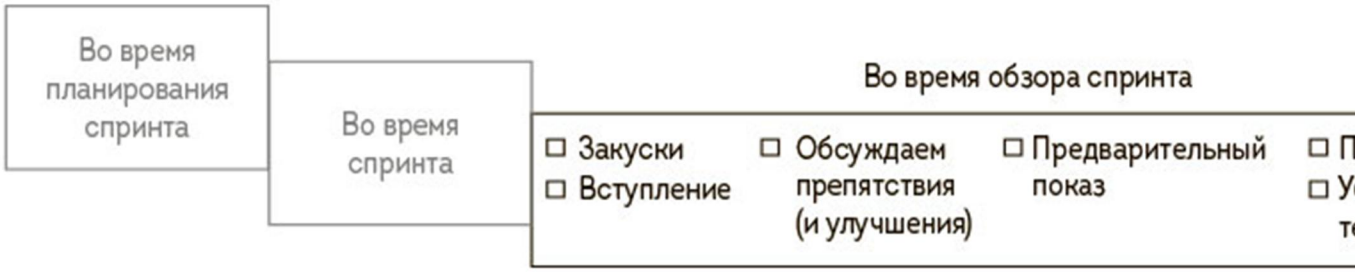

Рис. 8.3. Важные моменты, о которых нужно помнить во время обзора спринта

Закуски. Подкрепим силы

Все любят печеньки. Убедитесь, что на обзоре спринта есть то, что можно погрызть и выпить (крепкий виски, если вы действительно беспокоитесь о том, как пройдет встреча). Забавно, что позитивное или негативное впечатление от конференции или тренинга во многом зависит от того, как накормили собравшихся, поэтому не забывайте проложить путь к сердцу каждого участника через его желудок!

Вступление

Как только все расселись и завели легкую шутливую беседу, дегустируя закуски, владелец продукта должен коротко обозначить цель спринта, а также напомнить, что планируется показать. Кроме того, для спринтов в начале проекта не помешает объяснить стейкхолдерам критерии готовности (см. лайфхак 11).

Препятствия

После вступления правильным будет рассказать про те препятствия, которые повлияли на спринт, в том числе про то, почему они произошли и как с ними справились. Подобная открытость обеспечит вам бо́льшую поддержку при наличии системных препятствий. Например, с размещением команды и оборудованием рабочего пространства — когда возникают сложности с административно-хозяйственным отделом при заказе столов бо́льшего размера или расширении чилаут-зоны.

Кроме того, я рекомендую использовать эту часть встречи, чтобы кратко рассказать про ключевые улучшения, реализованные для достижения цели спринта. Не углубляйтесь в детали, ведь встреча посвящена обзору

продукта. Однако краткий перечень улучшений — отличная возможность продемонстрировать стейкхолдерам, что команда постоянно совершенствуется.

Основная часть

Демонстрация завершенной работы спринта не должна быть монологом, она должна стать приглашением к диалогу между бизнесом и scrum-командой. Это должна быть открытая и честная дискуссия о том, что уже сделано и что еще необходимо сделать.

Помните: эта встреча не слайд-шоу ради того, чтобы пустить пыль в глаза стейкхолдерам (если, конечно, ваша команда не разрабатывает PowerPoint[81] или Keynote![82]). Уверяю вас, любая дезинформация может отозваться негативными последствиями для команды.

Предварительный показ в обзоре спринта

Фундаментальный принцип обзора спринта — команда должна демонстрировать только такие истории, которые достигли критериев готовности. Однако это может беспокоить некоторых стейкхолдеров, и в реальности команда окажется под большим давлением, пока не покажет текущую работу, даже если она еще не завершена.

Вместо того чтобы упираться рогом, я рекомендую создать дополнительный пункт повестки «Скоро на экранах». Таким образом стейкхолдеры, словно бы посмотрев трейлер фильма, узнают о задачах, над которыми сейчас работает команда и которые будут показаны на предстоящих обзорах спринта, и при этом поймут, что работа над ним еще не завершена.

Перерывы и телефоны

Если встреча длится дольше часа, я предлагаю делать 5-минутные перерывы каждые 45 минут. Это поможет сохранить фокус. Но остерегайтесь «щенков с синдромом дефицита внимания», которые могут легко сбиться с пути и загулять где-нибудь в коридорах, — не давайте им уходить слишком далеко.

Кроме того, попросите каждого оставить свои смартфоны за дверью (пусть это и потребует от вас дополнительных усилий). Возможно, придется нанять вооруженных охранников, чтобы вырывать смартфоны из рук стейкхолдеров! Но как минимум вы должны объявить в начале встречи, что «для безопасности этого обзора спринта просим вас отключить все цифровые устройства». Удачи! И придумайте креативное наказание для тех, чьи телефоны зазвонят во время встречи.

ТАК НАЗЫВАЕМЫЕ ПРЕДЛОЖЕНИЯ

Без сомнения, на встрече вас завалят вопросами и предложениями. Я рекомендую сдерживать этот поток и не отклоняться от темы. Команда должна отвечать на любые вопросы, касающиеся того, что демонстрируется. А второстепенные вопросы или вопросы не по теме владелец продукта должен вынести на обсуждение «в рабочем порядке», организовав отдельную встречу с соответствующими стейкхолдерами.

Вне зависимости от того, насколько нелепыми они могут казаться, отдайте должное любым предложениям — просто запишите их на маркерной доске или на карточках. Если вам повезет, то, увидев безумные предложения, написанные черным по белому, стейкхолдеры отзовут свои «ценные рекомендации». Любые разумные предложения владельцу продукта, конечно же, следует добавить в продуктовый бэклог для дальнейшей оценки и обсуждения.

ПИКНИКИ ИЛИ СРАЖЕНИЯ

Обзор спринта может походить на летний пикник или, наоборот, превратиться в решающее сражение! Все зависит от того, приняты ли необходимые меры предосторожности, относятся ли все участники к каждой встрече серьезно и в то же время позволяют себе немного повеселиться. Не исходите из того, что уровень подготовки каждого стейкхолдера такой же, как у команды. Сделайте так, чтобы все участники встречи чувствовали себя комфортно, всегда объясняя, что происходит и почему.

Выравнивание ожиданий — вот что действительно важно. И достижение этой цели имеет решающее значение, если вы предпочитаете пикники сражениям!

ЛАЙФХАК 23. РЕТРОСПЕКТИВА ПРИ ЛЮБЫХ ОБСТОЯТЕЛЬСТВАХ

К сожалению, когда команда работает под давлением, первое, от чего отказываются, — ретроспектива спринта. Вынужден признаться: будучи начинающим scrum-мастером, время от времени я пропускал ретроспективы. Ирония в том, что подобные встречи особенно важны и полезны именно тогда, когда все идет не по плану. Соблюдайте дисциплину: каждый спринт должен завершаться ретроспективой в обязательном порядке — вопреки всем препятствиям и помехам.

ПОДКРЕПИТЕ ЦЕННОСТИ SCRUM

Важно помнить основные ценности Scrum: открытость, смелость, уважение, сфокусированность и преданность. Именно в ретроспективе спринта нужно следовать им максимально четко. Без атмосферы

открытости вы никогда не доберетесь до сути проблемы. Лишенной смелости команде не удастся преодолеть трудности. Без уважения команда не сможет высказывать конструктивную критику, а без сфокусированности и ответственности не будет прилагать достаточно усилий для решения проблем и достижения целей.

Каждый должен ориентироваться на эти ценности.

ЧТО ДЕЛАТЬ, ЕСЛИ У НАС ОДНОНЕДЕЛЬНЫЕ СПРИНТЫ?

Меня часто спрашивают: если команда работает короткими однонедельными спринтами, должна ли она проводить ретроспективу раз в два или три спринта, а не на еженедельной основе? На мой взгляд, однонедельные спринты слишком короткие (см. лайфхак 8). Однако, если в них есть потребность, не стоит пропускать ретроспективы. Просто проводите более короткие ретроспективы, но каждый спринт.

МЕСТО ПРОВЕДЕНИЯ, МЕСТО ПРОВЕДЕНИЯ И ЕЩЕ РАЗ МЕСТО ПРОВЕДЕНИЯ

От того, где вы проводите ретроспективу, зависит, сможете ли вы создать атмосферу открытости. Большой, душный зал заседаний с длинным официозным столом из красного дерева не будет этому способствовать и, может быть, даже превратит вашу ретроспективу в суд инквизиции. Вместо этого я рекомендую проводить ретроспективы в менее строгом месте, например в кафе или комнате отдыха (если у вас имеется такая). Даже кухня в силу близости к печенькам и кофе будет отличным выбором!

ПОДГОТОВКА

Идеально, если ваша команда придет на ретроспективу подготовленной, со списком проблем и предложений, и вы сможете сразу приступить к делу. Чтобы команде было легче подготовиться, рекомендую перед встречей разослать e-mail и обозначить области, на которых следует сфокусироваться. Темы могут быть такими:

- коммуникация;
- процессы;
- скоуп задач;
- качество;
- рабочая среда;
- навыки.

А теперь разберем типичные действия по улучшению по каждому из этих направлений.

Коммуникация

Какие улучшения возможны в этой области:

- Восстановить разорванные коммуникации между владельцем продукта и командой разработки (особенно если владелец продукта не находится рядом с командой).
- Устранить токсичные и бесполезные коммуникации между внешними стейкхолдерами и командой разработки.
- Восстановить разладившиеся коммуникации внутри команды (обычно из-за чрезмерного стремления все фиксировать письменно (и через e-mail) вместо личного общения).

Процессы

Какие улучшения возможны в этой области:

- Обновить программное и аппаратное обеспечение и сетевые подключения.
- Удостовериться, что scrum-процессы понятны и четко определены.
- Поддерживать инженерные стандарты, относящиеся к качеству кода, контролю версий и процессу деплоя[83].

Скоуп задач

Какие улучшения возможны в этой области:

- Убедиться, что в середине спринта не случится значительных изменений в объеме задач.
- Поддерживать единый формат описания пользовательских историй и критериев приемки.
- Убедиться, что задачи для планирования спринта понятны и четко сформулированы.

Качество

Какие улучшения возможны в этой области:

- Четко определить и поддерживать единые критерии готовности.
- Развивать практики тестирования, способствующие глубокой автоматизации тестирования.
- Способствовать тому, чтобы программисты брали на себя ответственность за тестирование собственного кода, а не отдавали это на откуп тестировщикам и/или владельцу продукта.

Рабочая среда

Какие улучшения возможны в этой области:

- Поддерживать общее рабочее пространство не слишком шумным и не создающим дополнительных трудностей.
- Обеспечивать достаточное кондиционирование воздуха или температуру.
- Удостовериться, что для комфортного существования есть все необходимое (одной микроволновой печки не хватит на 50 человек!).
- Удостовериться, что вокруг маркерной доски достаточно свободного пространства и что всем хватит инструментов для совместной работы.

Навыки

Какие улучшения возможны в этой области:

- Организовать необходимые тренинги при внедрении новых технологий.
- Проводить адаптационные тренинги для новых членов команды.
- Получать консультации специалистов, когда это требуется.

РЕЗУЛЬТАТЫ РЕТРОСПЕКТИВЫ

Когда все забурлило, несложно составить длинный перечень проблем, которые нужно решить. Но очень важно не поддаться избыточному оптимизму и не заявить, что все вопросы будут решены к следующей ретроспективе!

Наоборот, убедитесь, что все предложения задокументированы. Ваша цель — решить только часть проблем, возможно одну большую и несколько маленьких. Ничто не разрушает доверие внутри команды так, как избыток обещаний и их невыполнение. Поступайте ровно наоборот.

В завершение на большом листе бумаги выпишите согласованные со всеми действия и поместите его рядом с доской задач — в качестве постоянного напоминания для команды. Пожалуйста, не теряйте благоразумия, выставляя этот список на всеобщее обозрение. Если вы напишете «Запереть надоедливого спонсора проекта в клетке» и повесите лист на стене, это вряд ли поспособствует прогрессу в работе!

ФОРМАТ РЕТРОСПЕКТИВЫ

Чтобы привносить новизну, меняйте формат ретроспективы спринта на протяжении проекта. Можно использовать самые разные варианты, но для краткости я расскажу о двух моих фаворитах — «кругах» и «пузырях».

Круги

Метод кругов базируется на технике affinity mapping[84]. По сути, это графический инструмент для объединения разных неструктурированных идей в группы. Прежде чем приступить к рассмотрению этой техники, давайте определим четыре классических вопроса, часто задаваемых на ретроспективах Scrum:

- Что мы сделали хорошо?
- Что можно улучшить?
- Что мы должны попробовать в следующий раз?
- Какие проблемы нужно эскалировать наверх?

Я предпочитаю сократить этот список до двух вопросов:

- Что мы можем улучшить?
- Что мы делаем хорошо?

Сначала нарисуйте на маркерной доске горизонтальную линию. Слева напишите «Нуждается в улучшении», в середине — нейтральное «ОК», а справа — вдохновляющее «Идет хорошо».

Теперь, соблюдая традицию стикеров в Scrum, вручите каждому участнику команды стопку стикеров и попросите их написать свои мысли и/или проблемы.

Когда все закончат, предложите наклеить стикеры на доску вдоль шкалы — на то место, которое, по их мнению, максимально соответствует содержанию стикера. Обязательно обозначьте время, в течение которого команда выполняет это задание, чтобы избежать аналитического паралича.

Когда все стикеры окажутся на доске, настанет время объединить похожие вопросы, чтобы сформировать группы. Сделав это, возьмите хороший толстый маркер (не перманентный, иначе ваши проблемы останутся навсегда) и нарисуйте круги вокруг стикеров, отражающих схожие мысли и собранных рядом (см. рис. 8.4). Стикеры, которые не вписываются ни в одну группу, на этом этапе просто игнорируйте.

После того как вы завершили группирование, будет правильно кратко обсудить с командой стикеры, не попавшие ни в одну группу. Чаще всего их либо неправильно интерпретировали, либо неверно поняли; дело не в том, что команда не сходится в оценках.

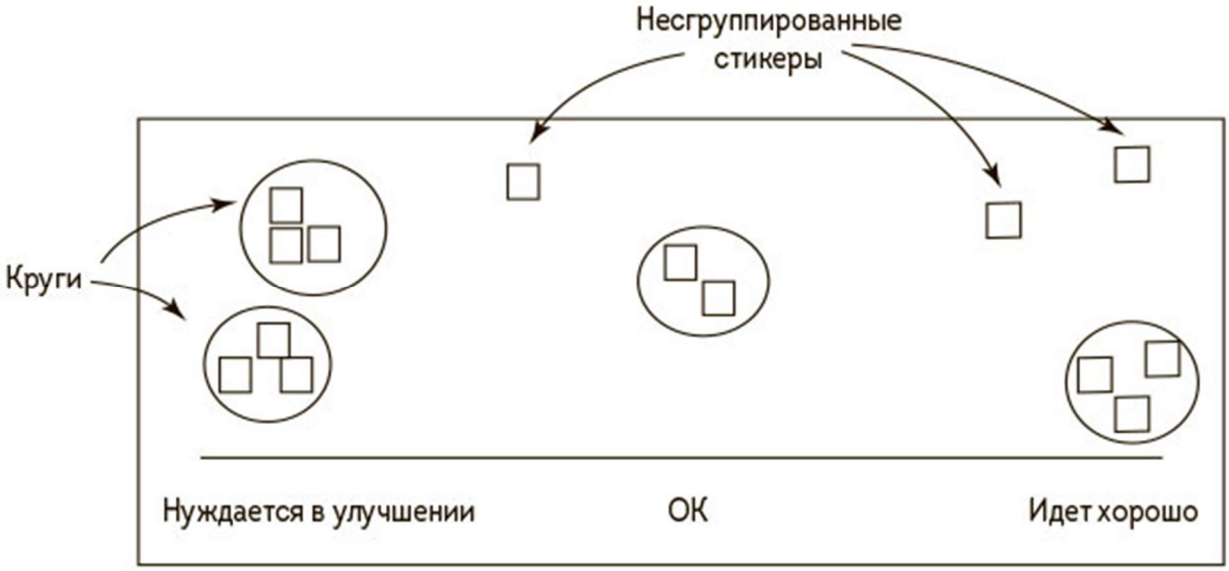

Рис. 8.4. Ретроспектива «круги»: схожие предложения сгруппированы и обведены в круг

Теперь (всей командой) обращайтесь к кругам в порядке приоритета (слева — больший приоритет, справа — меньший). Определите, что будет сделано в следующем спринте и как это поспособствует улучшениям, о которых речь шла выше.

Мне нравится такой подход по нескольким причинам:

- Он никого не ставит в неловкое положение (или в центр внимания).
- Команда постоянно двигается, оставаясь в тонусе.
- Здесь сочетаются тактильные и визуальные ощущения, что делает жизнь более интересной.
- Команда может сразу расставить приоритеты, взглянув на группы стикеров на стене.

«Пузыри»

Метод «пузырей» — определенно мой любимый подход для первых ретроспектив, когда вы начали работать с молодой scrum-командой (ее участники прежде не работали вместе).

Техника «пузырей» эффективна, ведь она способствует быстрому взаимодействию и в простой обстановке помогает узнать друг друга. К тому же не ставит никого в неловкое положение.

Как это работает? Во-первых, вы должны подготовиться к встрече, раздав бумагу и ручку каждому в команде.

Эту технику лучше всего применять, если в команде четное число участников, но это не жесткое правило.

Допустим, у нас есть команда из восьми человек. На первом шаге попросите каждого участника написать три самых острых вопроса,

на которых, по его мнению, нужно сфокусироваться в предстоящем спринте. Я рекомендую отвести на это 5 минут.

Затем объедините людей в пары (в нашем случае получится четыре пары, но не будет никакой крамолы, если в группе окажется три человека, когда у вас нечетное число участников). В течение 5 минут пары должны решить, какие три вопроса (из их общих списков) считать максимально приоритетными.

Вероятно, вы уже поняли, куда я клоню, но если еще не догадались, то следующий шаг — объединить две пары вместе. Теперь у нас две группы по четыре человека (см. рис. 8.5). Не забывайте о времени. Теперь на обсуждение вопросов и выбор трех самых важных вы можете дать командам чуть больше времени, ведь в каждой из них теперь по четыре участника.

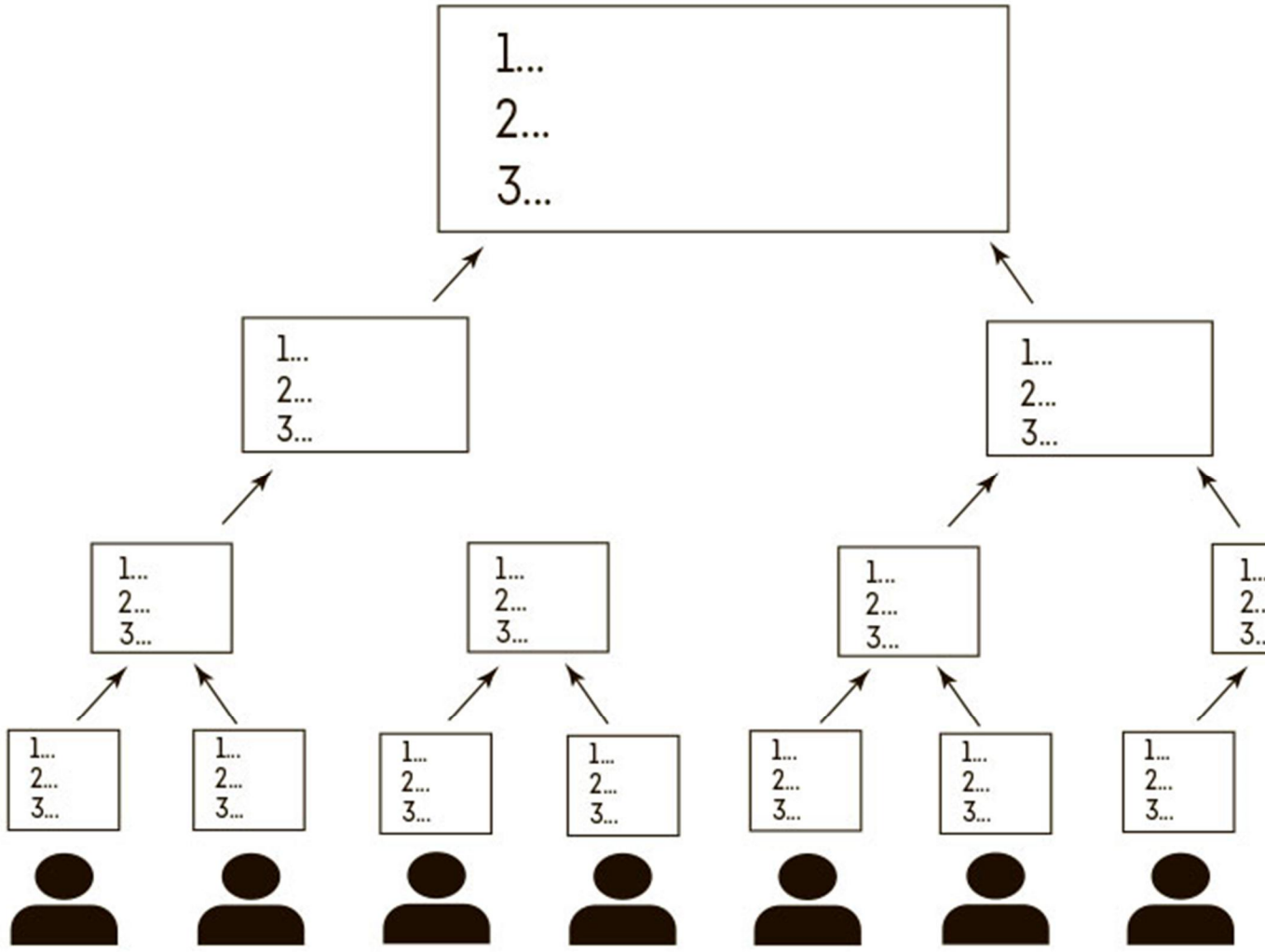

Рис. 8.5. Благодаря технике «пузырей» наиболее важные и срочные проблемы всплывают на поверхность

Предпоследний шаг — сформировать единую команду из восьми участников и выявить три проблемных «пузыря», которые и всплыли на самый верх.

На этом этапе вы, скорее всего, обнаружите, что участники преодолели стеснение и готовы открыто обсуждать проблемы, после чего вы определите, на чем следует сфокусироваться в следующем спринте.

Не пренебрегайте другими соображениями, высказанными на первых этапах подъема «пузырей». Задокументируйте их, и, когда вы придете на следующую ретроспективу, у вас уже будет удобная отправная точка для старта. Вопросы, которые оказались внизу перечней, на более поздних этапах могут стать приоритетными.

И наконец, команда может «освежиться» и повторить работу с «пузырями», чтобы выявить успешные действия и процессы, активность которых она однозначно захочет поддерживать в следующем спринте.

ОПЫТНЫЕ ПРОФИ

Когда вы работаете с закаленной в боях командой scrum-фанатов, мой совет — забудьте про все формализованные сценарии, для вашей ретроспективы любой из них покажется надуманным. Если команде комфортно, если она чувствует себя в своей тарелке, у вас не возникнет проблем собрать всех вместе и попросить открыто выразить свои опасения или, наоборот, радость. Сходите вместе на ланч или попить кофе (с блокнотом и ручкой) и просто поболтайте.

УЧАСТНИКИ РЕТРОСПЕКТИВЫ

Обширное scrum-сообщество, кажется, разделилось относительно того, должны ли владельцы продукта присутствовать на ретроспективе спринта. На мой взгляд, это обязательно, потому что они неотъемлемая часть scrum-команды. И все же до того, как ваша команда станет «хорошо отлаженной scrum-машиной», могут возникать проблемы с коммуникацией, особенно между разработчиками и владельцем продукта. Если вы ощущаете напряжение, я рекомендую вам как scrum-мастеру, в дополнение к «официальной» ретроспективе, провести неофициальную встречу отдельно с владельцем продукта.

СОХРАНЯЙТЕ НОВИЗНУ

Инспекция и адаптация — вот ваша главная цель. Ретроспектива спринта жизненно необходима, чтобы ваша команда постоянно совершенствовалась. На этих встречах вы можете применять самые разные техники, не стесняйтесь варьировать методы, чтобы ваша жизнь не была пресной. Эстер Дерби и Диана Ларсен собрали в своей книге «Agile-ретроспектива»[85] невероятное количество вариантов, которые вы можете взять на вооружение.

Самое главное: что бы ни случилось, не отменяйте ретроспективы!

ЛАЙФХАК 24. РИСКОВАТЬ И ОШИБАТЬСЯ

Моя мама работала директором детского сада, сейчас она на пенсии. У нее нет ни капли опыта в разработке программного обеспечения. Но она большая умница и умеет общаться с людьми, что делает ее выдающимся преподавателем. И хотя обсуждать с ней технические вопросы (например, рефакторинг или ревью кода) бессмысленно, я могу говорить с ней про элементы Scrum, базирующиеся на коммуникации. Например, про ретроспективы спринта или самоорганизующиеся команды.

Однажды я объяснял ей «революционные» scrum-методы, призванные помочь командам, — например, частые и открытые ретроспективы. До сих пор помню, как она удивленно спросила: «И где здесь большой секрет? Мне это кажется само собой разумеющимся». Сначала я ухмыльнулся, но после некоторых размышлений понял, что она была права! Это должно быть самоочевидным, не так ли? Почему же это стало новаторством для нашей отрасли? Я надолго задумался об этом и в итоге пришел к выводу, что ответ на этот вопрос сводится к одному большому уродливому монстру:

Страх!

Не случайно смелость — одна из ключевых ценностей Scrum. Чтобы успешно применять Scrum, нам придется преодолеть немало страхов. Очень важно страшиться страха (извините за тавтологию), потому что страх неминуемо ведет к сокращению инноваций и распространению культуры взаимных обвинений. А еще он вызывает аналитический паралич — когда любое действие команды подвергается тщательному исследованию, чтобы избежать ошибок и последующей порки. После этого пугающего вступления давайте разберем типовые страхи.

СТРАХ ПЕРЕМЕН

Страх — неизменный спутник перемен. Когда мы отказываемся от традиционных подходов и переходим на Scrum, то проводим невероятное количество изменений, не так ли? Как насчет программистов, проводящих тестирование? А что вы думаете о программном обеспечении, готовом к релизу каждый спринт? А что скажете о кросс-функциональных и самоорганизующихся командах? Не говоря уже об отсутствии менеджеров проекта! И это я перечислил только несколько изменений — вот ведь ужас!

Дело вот в чем: изменения пугают. Они выводят людей из зоны комфорта, нарушают статус-кво и часто воспринимаются как угроза теми, кому комфортно в их нынешнем положении. Увы, в компании всегда будут те, кто отвергает изменения — вне зависимости от выгод, которые те с собой несут. Данный факт поможет понять и принять эту

аксиому, в противном случае вам придется жить с чувством неизбывного разочарования. Я согласен с Майком Коном, который пишет: «Вместо того чтобы фокусироваться на тех, кто сопротивляется или выступает против Scrum, потратьте время и усилия, помогая тем, кто исполнен энтузиазма»86. На мой взгляд, это лучший способ набрать обороты и переубедить скептиков.

СВОБОДА МЕНЯТЬСЯ

Истинный ключ к успешной трансформации в том, как эта идея будет сформулирована и объяснена. Немногие чувствуют себя комфортно при ультимативных изменениях, полностью перекраивающих существующий ландшафт. Я рекомендую представлять трансформацию как длительный эксперимент. Объясните, что весь процесс перемен находится под пристальным вниманием, это не свершившийся факт. И если что-то не будет работать, команда проанализирует это и адаптируется сообща.

Осознание этого безопасного варианта зачастую помогает развеять опасения тех, кто боится перемен.

СТРАХ РАЗОБЛАЧЕНИЯ

Некоторые разработчики предпочитают, чтобы их оставили в покое и они «в своем углу» могли сфокусироваться на работе (хорошо это или плохо). Вот почему они могут уходить в глухую оборону, когда требуется регулярно рассказывать о том, как продвигается работа. Проблема этих разработчиков, подобных закрытой книге, в том, что Scrum — это про постоянную инспекцию. Конечно, не для шпионажа, а для выявления ненужных и ошибочных действий.

«Инспекционные» встречи, такие как ежедневный scrum, обзор спринта или ретроспективы, могут выявить тех индивидов (я использую это слово специально), которые искренне не хотят совершать правильные поступки в интересах команды. Эти несколько «тухлых яиц» больше всего боятся «разоблачения». Они опасаются, что с их работы «снимут покровы» и все увидят ее низкое качество.

СВОБОДА БЫТЬ ОТКРЫТЫМ

Я вырос в Австралии, где с раннего детства мы знали об опасности солнечных лучей. Если вы слишком долго будете на солнце, станете красным, как медицинский крест, и ваша кожа будет болеть несколько дней. Однако не будем слишком суровы к источнику нашей жизни. Если недолго бывать на солнце, можно получить красивый, здоровый загар и полезную дозу витамина D.

Фокус Scrum направлен на регулярную проверку продукта и процессов — это про то, как получить здоровый загар и избежать солнечных ожогов.

Гораздо легче ухаживать за загоревшей кожей, а не лечить красную и воспаленную. Точно так же и при разработке программного обеспечения. Если часто мониторить процессы и выявлять проблемы на ранних этапах, будет гораздо меньше неприятных сюрпризов, которые на завершающих этапах потребуют сверхусилий и чрезмерного внимания.

Когда инспекцию подают в позитивном ключе, участники команды (которые искренне хотят сделать что-то хорошее) принимают подобные встречи, даже если им необходимо оставаться в своем углу, отгородившись от всего наушниками. А что же будет с «тухлыми яйцами»? Надеюсь, вы сможете обнаружить их раньше, чем они испортят всем завтрак!

СТРАХ ОШИБОК

Токсичные организации с дисфункциональной и нездоровой культурой слежки за коллегами, привычкой замалчивать ошибки и скрывать внутренние конфликты с большими сложностями будут внедрять Scrum или любые другие фреймворки, основанные на эмпирическом управлении процессом. Scrum превращает разработку программного обеспечения в открытую книгу, где ошибки отчетливо видны.

Системная проблема для подобных организаций в том, что по большей части IT-индустрия — питательная среда для ошибок, ведь риск — неотъемлемая часть разработки программного обеспечения. Роман Пихлер объясняет:

Таким образом, риск — это неотъемлемая часть разработки программного обеспечения. Ни один продукт без риска просто не выпустить. С риском связана неопределенность, и чем она выше, тем рискованнее проект. Неопределенность же, в свою очередь, связана с недостатком знаний. Чем меньше мы знаем о том, что нужно разрабатывать и как это делать, тем больше неопределенность. Таким образом, знания, неопределенность и риск взаимосвязаны[87].

СВОБОДА СОВЕРШАТЬ ОШИБКИ

Простой факт: самые важные уроки мы получаем, когда рискуем и ошибаемся. Не верите? Вот вам пример. Дэн Сенор и Сол Сингер, авторы «Нации умных людей»[88], рассказывают о таком аспекте жизни в Израиле, который более позитивен, чем истории из СМИ о постоянных конфликтах с их непростыми соседями. Несмотря на ограничения, вызванные постоянной угрозой войны и сравнительно небольшим населением — 7 миллионов[89], у этой страны наибольшее число технологических стартапов в мире, котирующихся на NASDAQ (после США и Китая). Дэн Сенор и Сол Сингер предлагают свою гипотезу:

Подход израильтян к работе и неформальный стиль проистекают из присущей им толерантности к тому, что называют «конструктивными

ошибками» или «интеллектуальными ошибками». Большинство местных инвесторов уверены, что без толерантности к большому количеству неудач настоящие инновации невозможны. В израильской армии наблюдается тенденция рассматривать все действия — как успешные, так и неуспешные — во время тренировок, симуляций и даже в реальных боях, как ценностно-нейтральные. То есть если вы идете на оправданный риск, а не действуете опрометчиво, значит, вы чему-то учитесь.

Для меня это важнейший урок гибкости на макроуровне. И мой опыт взаимодействия американцев и израильтян в области технологий только подкрепляет это убеждение: израильтяне действительно демонстрируют потрясающую терпимость к так называемой неудаче. Ашер Мозес написал в газете *Sydney Morning Herald*: «В Кремниевой долине неудачу отмечают и видят в ней возможность узнать что-то новое»90. Мы, австралийцы (как и многие другие), можем немало полезного извлечь из подобного позитивного отношения к риску и ошибкам.

Мне нравится пример, который подают такие компании по разработке программного обеспечения, как Facebook. Они часто просят разработчиков «выложить свой код на сайт [компании] на первой неделе работы»91.

Это тонкий психологический прием. Вы показываете новичкам команды, что им не нужно немедленно выдавать безупречный результат. Вы готовы к тому, что они могут ошибиться, и это не станет большой проблемой. Позволив устранить свою первую ошибку на раннем этапе в безопасной обстановке, вы создаете атмосферу «психологической безопасности»92. Так формируется комфортная рабочая среда, благотворная для просчитанного риска и инноваций.

На рис. 8.6 схематично изображены страхи, терзающие команды, и способы избавиться от них.

Рис. 8.6. Каждый страх — боязнь перемен, страх быть разоблаченным или страх ошибки — команда может преодолеть

ПОДНИМИТЕ НАСТРОЕНИЕ

Ничто не помогает справляться со страхами так, как юмор. В чрезвычайно турбулентной и стрессовой среде, в которой я когда-то работал, у нас был CEO, который здорово помогал команде расслабиться после дорогостоящей ошибки, объявляя со смешком: «Похоже, фея факапов снова в городе, не так ли, ребята?» Хотя кому-то эта фраза покажется грубоватой, всем в комнате она помогала расслабиться перед тем, как приступить к поиску решения.

Другой положительный эффект, который вы почувствуете от открытого обсуждения ошибок и уязвимостей, — установление тесных связей внутри команды. Открытость помогает разрушить искусственную броню совершенства, которую многие из нас носят, стремясь скрыть тот факт, что все мы люди и нам свойственно ошибаться.

Помните, что каждому нужен луч надежды. Те, кто готов извлечь уроки и увидеть возможности в ошибках и проблемах, вскоре обнаружат, что в своей работе они теперь полностью свободны от страха.

ЗАКЛЮЧЕНИЕ

В трех лайфхаках этой главы я рассказал о тактике и инструментах, а также дал несколько советов, чтобы помочь вашей команде проводить обзоры спринта и ретроспективы. Давайте вспомним, какие вопросы мы обсудили.

Лайфхак 22. Чек-лист для обзора спринта

- как эффективно подготовиться к обзору спринта;
- обзор спринта не должен превращаться в презентацию-монолог;
- элементы, которые нужно включить в обзор спринта.

Лайфхак 23. Ретроспектива при любых обстоятельствах

- почему так важно не отменять ретроспективу;
- два полезных формата ретроспективы: «круги» и «пузыри»;
- ключевые области улучшений для инспекции и адаптации.

Лайфхак 24. Рисковать и ошибаться

- важно развивать терпимое отношение к ошибкам ради открытой и честной обратной связи;
- типичные страхи, тормозящие команду;
- способы помочь команде преодолеть эти страхи.

Глава 9

УПРАВЛЯЕМ МЕНЕДЖЕРАМИ

Фреймворк Scrum детально не описывает участие и вклад основных стейкхолдеров, спонсоров проекта и других менеджеров. Разумеется, это не значит, что эти роли не существуют или что они не являются неотъемлемой частью успеха наших проектов.

Следующие три лайфхака предлагают несколько способов для интеграции этих ролей в Scrum.

Лайфхак 25 «Впечатление создает реальность» посвящен тому, как помочь спонсорам проекта понять Scrum, про который они могут ничего не знать.

Лайфхак 26 «Мастера и лорды» рассказывает, как chief scrum-мастер помогает организовать работу нескольких scrum-мастеров в больших компаниях.

Наконец, лайфхак 27 «Трансформация менеджеров в матричной структуре» подробно описывает выгоды от создания и поддержания организационной структуры, в центре которой стоят команды. Также этот лайфхак рассказывает, как менеджерам проектов и функциональным менеджерам работать со scrum-командами.

ЛАЙФХАК 25. ВПЕЧАТЛЕНИЕ СОЗДАЕТ РЕАЛЬНОСТЬ

Мы играем в карточные игры на наших встречах по оценке; мы украшаем рабочее пространство разноцветными стикерами; мы не мечемся, как вспугнутые индейки перед Рождеством; мы собираемся на ежедневные стендапы и смеемся над бедными разработчиками в старых ковбойских шляпах (см. лайфхак 20) — одно сплошное веселье!

Официально Scrum предусматривает только три роли — scrum-мастер, владелец продукта и команда разработки. Однако, если вы хотите, чтобы ваши scrum-проекты стали успешными, придется быстро поладить и со спонсором проекта. Это еще одна важная роль, от которой зависит финансирование большинства проектов.

К сожалению, многим из вас придется взаимодействовать со спонсорами проекта (не из сферы IT), которые убеждены, что продуктивность несовместима с весельем (для них это аксиома офисной жизни).

Я бы не хотел делать поспешных выводов, но, если один спонсор (или больше) пришел из отрасли, пронизанной культурой командного управления и неусыпного контроля, для него 18-часовой рабочий день будет нормой, а если сотрудники не выглядят так, будто им грозит четвертый инфаркт, значит, они недостаточно погружены в работу!
А теперь представьте, что чувствует этот человек, наблюдая, как сотрудники не засиживаются на работе и уходят в одно и то же время; слыша, как разобщенные прежде команды болтают между собой и тратят драгоценное рабочее время на так называемую нефункциональную работу — тестирование и рефакторинг. Уверен, в лучшем случае он испытывает дискомфорт, а в худшем — ярость!

Владелец продукта должен быть медиатором между scrum-командой и «старшими по званию» спонсорами проекта. Однако эту роль может взять на себя и scrum-мастер, помогая спонсору проекта не слетать с катушек и не психовать. Вам предстоит подключить все свое ораторское мастерство, чтобы постоянно выравнивать ожидания, понимать, как мыслит спонсор проекта, и согласовывать его видение с видением команды.

ВЫСТРАИВАЙТЕ ОТНОШЕНИЯ

Не ждите официального обзора спринта или, что еще хуже, чрезвычайной ситуации, чтобы встретиться со спонсором проекта. Проявляйте активность! Проводите со спонсором проекта регулярные встречи по синхронизации независимо от того, что происходит в данный момент, — например, по пути на ланч или отправляясь пить кофе. На этих встречах вы сможете убедить спонсора в том, что изменения носят позитивный характер, а также оценить, как он воспринимает состояние проекта в данный момент.

Корпоративная культура и принципы принятия решений могут стать одним из самых коварных и неистребимых препятствий, с которым предстоит столкнуться вашей команде, поэтому хорошие отношения со спонсором проекта дают возможность получить прямую информацию и узнать о лазейках, чтобы нивелировать потенциальные препятствия на вашем пути. Вооружившись этим знанием, вы можете разработать план их преодоления и реализовать его, когда настанет необходимость.

ТОЧКА ОТСЧЕТА

Всегда важно четко и ясно понимать цели проекта. Конечно, в случае Scrum это во многом зона ответственности владельца продукта. Однако разумно убедиться, что и спонсор проекта вносит свою лепту в видение конечного продукта, которое служит путеводной звездой от самого старта до финишной черты.

Для разработки общего видения продукта я рекомендую использовать такой простой инструмент, как бриф (или onepager — «одностраничник»). В таблице 9.1 приведен пример брифа (созданный под влиянием Джима Хайсмита и Пита Димера).

Краткая презентация	**Для… Кто… Те… В отличие…**
Целевая аудитория	1. Люди, работающие в… 2. Те, кому нужны… 3. …
Ключевые опции	1. Поделиться контентом с… 2. Персонализировать рекламные объявления в… 3. …
Выгоды/Отличия	1. Более надежное, чем… 2. Лучшая интеграция с… 3. …
Метрики/Цели	1. X пользователей в месяц 2. Кликабельность Y в месяц

	3. …
Контрольные точки	1. V 1.0 альфа 2. V 1.0 бета 3. …
Показатели производительности	1. X запросов в секунду 2. Y одновременно работающих в системе пользователей 3. …
Условия сделки	Объем работ: изменяемый Ресурсы: фиксированные График: жесткий

Таблица 9.1. Простой, на одну страницу, бриф продукта

Спонсор проекта может не знать деталей текущего бэклога продукта — ему вполне достаточно видения продукта, представленного в кратком брифе. Разногласия, возникающие из-за тех или иных требований, должны преодолеваться благодаря решениям, максимально соответствующим тому, что зафиксировано в этом документе.

ВОВЛЕКАЙТЕ ИХ

Я считаю, спонсоры проекта меньше расстраиваются из-за тех проблем, в решении которых они принимают участие. Вот почему я советую: информируйте их о трудностях, как только они возникают, предлагайте несколько вариантов решения и дайте спонсорам почувствовать причастность к процессу и корректировке курса.

Мне нравится периодически рассылать приглашения внешним участникам на встречу по покеру планирования (см. лайфхак 14). Эти встречи по оценке динамичны, информативны и пропитаны командным духом. И если они хорошо организованы, то спонсор проекта выйдет с ощущением: команда точно знает, что делать!

Само собой, вы должны приглашать спонсора проекта на регулярные обзоры спринта и вовлекать его в процесс, чтобы тот мог дать команде обратную связь и воодушевить ее (см. лайфхак 22).

ДЕРЖИТЕ ИХ В КУРСЕ

Спонсоры проекта терпеть не могут оставаться в неведении. Scrum делает акцент на прозрачности и наглядности, и потому больше не должно быть никаких секретов. Используйте эту особенность Scrum, заполняя ваш календарь встречами на месяц вперед (см. рис. 9.1):

- Проведите спонсору проекта «тур по доске задач» и повторяйте это время от времени. Объясните ему, что означают все эти цветные стикеры и почему у вас был странный небольшой всплеск на диаграмме сгорания на этой неделе (см. лайфхак 21).
- Проводите регулярные презентации или бизнес-завтраки, которые помогают углубиться в специальные практики Scrum, такие как *объяснение относительной оценки* или *понимание пользовательских историй.*

Рис. 9.1. Поддерживайте несколько точек контакта со спонсорами вашего проекта, чтобы они были в курсе происходящего

- Распространяйте книги и статьи для популяризации Scrum. Консервативные организации не любят чувствовать себя первопроходцами и первооткрывателями новых способов работы, поэтому им важно осознавать, что Scrum быстро становится мейнстримом.
- Рассказывайте о своих успехах, выпуская ежемесячное обновление историй успеха Scrum. Этот материал не должен фокусироваться только на количественных показателях, таких как уменьшение количества багов или более быстрое развертывание. Вам нужно говорить и о качественных результатах. Такие достижения, как преодоление барьеров между ранее изолированными группами, тоже достойны внимания.
- Я очень люблю делиться «байками с полей сражений». Майк Кон называет их «презентациями отчета о собственном опыте» и справедливо замечает, что «ничто не впечатляет больше, чем рассказ того, кто уже сделал это».

ПОДДЕРЖИВАЙТЕ ДИПЛОМАТИЧЕСКУЮ ДИСЦИПЛИНУ

Никогда не говори «нет» — это урок, который я усвоил не сразу. Никому не нравится слышать «нет», поэтому нам часто приходится идти на уловки, чтобы замаскировать отказ.

Вести себя как агрессивный ротвейлер, который готов дать решительный отпор и сказать «нет», прикрываясь правилами Scrum, — не лучшая идея. Даже если спонсор проекта понимает и принимает причины, из-за которых вы не можете с ним согласиться, он вряд ли будет рад услышать категорическое «нет» (такова наша природа). Спонсоры проекта нанесут ответный удар, если почувствуют, что их грубо игнорируют.

Мне нравится один способ говорить «нет». Вот как это бывает:

Спонсор: Я знаю, спринт уже спланирован и идет полным ходом, но можем ли мы ненадолго отложить пользовательскую историю ABC и вместо этого реализовать пользовательскую историю XYZ?

Я: Конечно, мистер Стейкхолдер, мы можем сделать что угодно. Однако, пожалуйста, примите во внимание последствия. Мы собьем команду с пути, а ведь она уже набрала ход, отменим спринт и, самое плохое, подорвем наши scrum-процессы и веру разработчиков в новые принципы работы.

Согласны ли вы, что негативные последствия на данном этапе перевешивают преимущества? Итак, как насчет того, что в следующем спринте, который начнется через несколько дней, мы первым делом займемся пользовательской историей XYZ?

Чаще всего такой подход творит чудеса и действительно создает беспроигрышную ситуацию: каждый уходит, получив то, что хотел, и при этом чувствует, что был великодушным и повел себя разумно.

И все же вы можете столкнуться с сопротивлением, начав обсуждать необходимость выделить время на чисто технические задачи (в отличие от функциональных задач), такие как юнит-тестирование и рефакторинг. Я нашел самый простой способ решить эту проблему — использовать статистику и кейсы других известных и очень успешных компаний. Например, когда я объясняю необходимость разобрать накопившийся технический долг, мне нравится говорить на языке спонсора и цитировать статью Wall Street Journal о eBay[93]:

Система eBay, которая включала 25 миллионов строк негибкого кода, вскоре стала долговым обязательством… Преимущества были неочевидны, зато риски более чем реальны. Но вы должны просто это сделать. В противном случае отстанете и вас вытеснят из бизнеса.

Хотя спонсор может не до конца понимать, что такое технический долг и рефакторинг, я обнаружил, что он восприимчив к аргументам типа «если это достаточно хорошо для eBay и об этом пишет Wall Street Journal, то и нам следует поступать также», — и меня это вполне устраивает.

ПОМНИТЕ, КТО ПЛАТИТ ПО СЧЕТАМ

Помните выражение: «Кто платит, тот и заказывает музыку»? Спонсоры — это те, кто платит за то, что разрабатывает scrum-команда. И они имеют полное право вести дела так, как им нравится. Осторожнее, эта возможность не должна стать вашей реальностью. Последнее, чего вы хотите, — вернуться к темному прошлому только лишь потому, что спонсор не понимает Scrum.

Если он считает, что Scrum плодит команды недисциплинированных ковбоев, ваша задача, если уж вы выбрали эту работу, — активно взаимодействовать с ним, объяснить, почему Scrum полезен и каким образом он предоставляет наилучшую возможность получить высокую отдачу на каждый вложенный доллар.

ЛАЙФХАК 26. МАСТЕРА И ЛОРДЫ

Когда один scrum-мастер превращается в двух, затем в трех, потом и в четырех, с каждым разом становится все интереснее и интереснее. Scrum явно утвердился в вашей компании, и, похоже, надолго. Это прекрасно, тем не менее необходимо возвести прочные строительные леса, чтобы способствовать быстрому позитивному росту, а также поддерживать уровень согласованности и дисциплины во всех коллективах, число которых существенно возросло.

Чтобы избегать проблем, учредите новую должность с широкими полномочиями для поддержания стандартов и согласованности. Идеальным вариантом был бы scrum-эквивалент офиса управления проектом (project management office, сокр. PMO), однако создать его — задача не из легких. Поскольку эта книга посвящена эффективным лайфхакам, я рекомендую вариант попроще (хотя бы для начала) — chief scrum-мастер.

SCRUM-МАСТЕР ПРОТИВ CHIEF SCRUM-МАСТЕРА

Хотя Scrum не предполагает никаких руководящих ролей, в реальности должно быть лицо, исполняющее такие обязанности, как управление персоналом, карьерное продвижение и наставничество в своей профессиональной области. Иными словами, chief scrum-мастер выполняет эту роль для группы scrum-мастеров, а также выступает в качестве scrum-коуча в рамках всей организации (см. рис. 9.2). Если вы предпочитаете более свободную структуру, вам стоит рассматривать chief scrum-мастера как фасилитатора в сообществе практикующих scrum-мастеров (см. лайфхак 27).

Рис. 9.2. Хотя роли разные, у них есть общее

Теперь, выяснив соотношение этих двух ролей, давайте разберем функционал каждой из них. Начнем мы с chief scrum-мастера.

ОСНОВНЫЕ ФУНКЦИИ CHIEF SCRUM-МАСТЕРА

«Scrum. Гибкая разработка ПО» Майка Кона — надежный фундамент для базовых функций PMO, основанного на Scrum. Я планирую кратко их изложить, чтобы описать позицию chief scrum-мастера. Я привожу свою интерпретацию этих функций, поэтому, если вы ищете оригинальные описания Майка Кона, пожалуйста, обратитесь к его книге.

Обучение и коучинг

Начинающим scrum-мастерам нужно учиться, чтобы стать хорошими scrum-мастерами, зрелые же scrum-мастера должны учиться тому, как стать великими scrum-мастерами. Для этого необходимо разработать специальные программы обучения.

Вызов установленным принципам взаимодействия

Майк Кон94 говорит о вызовах, которые нужно бросать командам, «скатывающимся к старым привычкам». Безусловно, это важно, но для chief scrum-мастера не менее важно бросать вызов компании в целом, если она дрейфует обратно к своим (системным по большей части) дурным привычкам.

Предложение и поддержка инструментов

Независимо от того, решит ли компания использовать высокотехнологичные или низкотехнологичные инструменты (или их

комбинацию), необходимо определить, адаптировать и контролировать версию разных шаблонов артефактов.

Определение и улучшение метрик

Метрики следует использовать с осторожностью — на пользу, а не во вред (см. лайфхак 19). Chief scrum-мастеру необходимо убедиться, что соответствующие метрики грамотно сформулированы и корректно применяются.

Помощь в создании сообщества практикующих специалистов

В идеале сообщества практикующих специалистов, ориентированные на конкретный функционал, должны органически формироваться снизу, на местах95. Тем не менее такие сообщества часто нуждаются в начальном импульсе и периодическом стимулировании, чтобы работать эффективно.

Обеспечение согласованности

Обеспечение согласованности между командами, возможно, самая главная функция chief scrum-мастера. Важно поддерживать и ее, и дисциплину, особенно когда у вас несколько команд и несколько scrum-мастеров. Если командам требуются разные подходы (по любой причине), изменения необходимо внедрять систематически и целенаправленно под руководством chief scrum-мастера.

Координация команд

С несколькими взаимосвязанными командами, работающими над одним и тем же бэклогом продукта, вам предстоит определить и поддерживать процессы, чтобы обеспечить беспрепятственную координацию.

К тому же…

Ниже приведены дополнительные функции, которые, я полагаю, можно добавить в список Майка Кона.

Непрерывное продвижение Scrum

Scrum следует продвигать не только на ранних этапах внедрения, но и по мере развития организации. Новые стейкхолдеры регулярно будут подключаться к работе команд, и им потребуется разобраться в рабочей среде Scrum и принять ее.

Формирование подхода

Помните: Scrum — это фреймворк, а не жестко предписанный метод (см. лайфхак 2), поэтому каждый конкретный подход (соответствующий поставленным целям) должен быть изначально задан chief scrum-мастером.

Образование на уровне компании

По мере того как будет внедряться каждая новая инициатива или практика Scrum, компании потребуется непрерывное обучение, чтобы все понимали и осознавали преимущества трансформации.

Выравнивание критериев готовности в командах

Координация определения и развития согласованных критериев готовности во всех командах важна для обеспечения согласованности ожиданий, особенно в ключевые периоды интеграции.

Непрерывное улучшение процессов с помощью собирательных ретроспектив

Немало полезных предложений выдвигается на ретроспективах спринтов разных команд. Важно, чтобы блестящая идея, которую предлагает команда X (и которая может оказаться полезной для других команд), эффективно применилась бы во всех группах.

Эскалация проблем

Не все препятствия, мешающие работе отдельной команды, могут быть устранены scrum-мастером. Прежде всего это касается вопросов, связанных с организационными ограничениями: рабочей среды и/или схем вознаграждения. В этих ситуациях chief scrum-мастер становится первой ступенью, на которую будет эскалирована проблема.

HR для scrum-мастеров

Как и у всех сотрудников компании, у scrum-мастеров есть свой функционал: планирование карьеры, пересмотр вознаграждений, обучение и наставничество96.

Создание рабочей среды, благоприятной для Scrum

Для успешного функционирования Scrum рабочая среда должна быть коллаборативной и комфортной (для команд), с минимальным пагубным влиянием из внешнего мира (см. лайфхак 3). Обеспечьте хорошие рабочие отношения с административно-хозяйственным отделом, который поможет переоборудовать рабочие места там, где нужно, и в то время, когда это необходимо. Это неотъемлемая часть успеха Scrum.

ОСНОВНЫЕ ФУНКЦИИ SCRUM-МАСТЕРА

Когда scrum-мастер — одинокий волк в небольшой компании, функции chief scrum-мастера, перечисленные выше, также ложатся на него. Однако если вы можете позволить себе отдельную позицию chief scrum-мастера, то scrum-мастер должен быть освобожден от этих обязанностей, чтобы сосредоточиться на следующем функционале.

Улучшение процессов

Основная обязанность scrum-мастера — определение ключевых точек улучшения с помощью количественных метрик (см. лайфхак 19) и ретроспектив спринтов (см. лайфхак 23).

Управление препятствиями

Одна из наиболее известных функций scrum-мастера — управление препятствиями, благодаря чему лежачие полицейские на пути проекта не превратятся в кирпичные стены (см. лайфхак 9).

Дипломатия

Разобщенность между группами, которым предстоит работать вместе в новой scrum-команде, будет особенно велика в первые дни. Объединение «вокруг костра» — одна из самых деликатных, но критически важных обязанностей scrum-мастера.

Коучинг

Первоочередная обязанность scrum-мастера — в полном объеме понимать фреймворк Scrum. Точнее говоря, scrum-мастер должен быть в высшей степени компетентен в подходах, используемых командами (что подразумевает экспертность в самых разных инструментах и методиках).

Управление изменениями

Изменения будут регулярно вноситься в общий объем продукта, а также в отдельные элементы бэклога продукта. Scrum-мастеру чрезвычайно важно контролировать изменения и направлять разных участников при управлении этими изменениями.

Поддержание критериев готовности

После того как scrum-команда определила критерии готовности (см. лайфхак 11), scrum-мастеру предстоит действовать в тесном контакте с владельцем продукта и разработчиками, чтобы обеспечить их поддержание.

Поддержание эффективной коммуникации

Scrum опирается на позитивное взаимодействие людей, поэтому scrum-мастер должен активно поощрять культуру, которая делает подобную коммуникацию возможной.

Обновление артефактов

Scrum-мастеру следует взаимодействовать с разработчиками, чтобы гарантировать, что артефакты спринтов будут регулярно обновляться (часто это требует искусства «контроля без ворчания, придирок и нытья»).

Организация обучающих семинаров

До запуска проекта scrum-мастер должен поработать с владельцем продукта, чтобы облегчить создание бэклога продукта. Например, с помощью семинаров по пользовательской истории и встреч по оценкам (см. лайфхак 14).

Фасилитация активностей Scrum

На протяжении всей итерации спринта происходит ряд активностей. Scrum-мастер отвечает за их бесперебойную работу, что требует от него организовывать и фасилитировать:

- Ежедневный scrum (см. лайфхак 20)
- Планирование спринта (см. лайфхак 8)
- Обзор спринта (см. лайфхак 22)
- Ретроспективу спринта (см. лайфхак 23)

Преданный своему делу scrum-мастер будет стараться оживлять эти встречи, меняя форматы и проводя их в разных местах, чтобы сделать жизнь интереснее.

УСТОЙЧИВАЯ ЭКОСИСТЕМА

Уф! Прочитав эти списки функций, вы наконец-то осознаете, что поддержание устойчивой экосистемы Scrum — нетривиальная задача.

Полагаю, ключ к успеху, особенно в связи с широким распространением Scrum, заключается в поддержании согласованности, дисциплины и непрерывного обучения. Единый chief scrum-мастер при поддержке мотивированных scrum-мастеров обеспечит достижение этих важнейших факторов успеха.

ЛАЙФХАК 27. ТРАНСФОРМАЦИЯ МЕНЕДЖЕРОВ В МАТРИЧНОЙ СТРУКТУРЕ

Ни для кого не секрет, почему нам пришлось так долго ждать появления электромобилей. Влиятельные руководители, удовлетворенные существующим положением дел, готовы подавить в зародыше любое движение за позитивные изменения и эффективность. Вы не можете их винить — это игра на вылет, в которой одна отрасль отмирает, чтобы возникла новая, более эффективная. Это та же трудная битва, в которую мы вступаем, начиная широко внедрять Scrum в крупной компании, не готовой к изменениям. Прежде всего внедрение Scrum (особенно в крупных и консервативных компаниях) требует, чтобы типичные вертикально интегрированные центры принятия решений развивались (или даже исчезали). А завершение этой эволюции невозможно без дальновидного руководства, искренне заинтересованного в постоянном улучшении, а не в подковерных играх.

В этом лайфхаке мы обсудим несколько способов, как помочь организациям, застрявшим в болоте традиционной структуры. Очевидно, это обширная и основательная тема, и ее не охватить в одном лайфхаке, но давайте будем амбициозными! А именно сосредоточимся на трех ключевых областях: 1) варианты общей организационной структуры; 2) варианты для традиционного менеджера проекта и 3) варианты для функциональных менеджеров.

ЭВОЛЮЦИЯ ИЗ МАТРИЦЫ

По мере того как работе, связанной с проектами, стали уделять все больше внимания, организационные структуры вынужденно адаптировались к новым условиям. Последним направлением эволюции стало движение к более проектно-ориентированным структурам, таким как сбалансированная матрица. Однако достаточно ли подобные структуры развиты, чтобы создавать устойчивую благоприятную среду для успешной работы scrum-проектов? Прежде чем ответить на этот вопрос, давайте вспомним основные характеристики типичных организационных структур.

Функциональные организации

Характеристики функциональной организации сводятся к следующим:

- Состоит из профильных отделов.
- Функциональные менеджеры наделены широчайшими полномочиями (см. рис. 9.3).

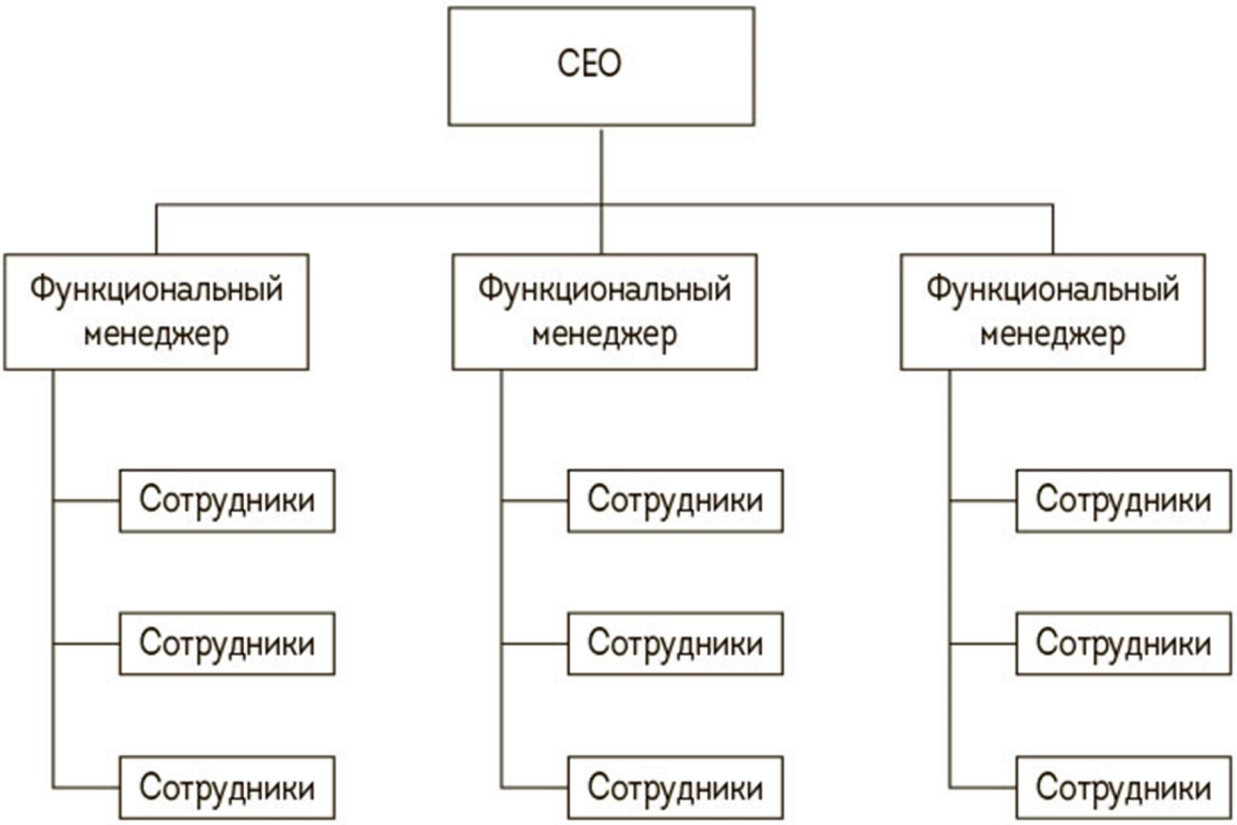

Рис. 9.3. Типичная функциональная организационная структура

Проектные организации

Характеристики проектной организации включают в себя следующие:

- Состоит из динамически формирующихся проектных команд.
- Менеджеры проектов наделены максимальными полномочиями (см. рис. 9.4).

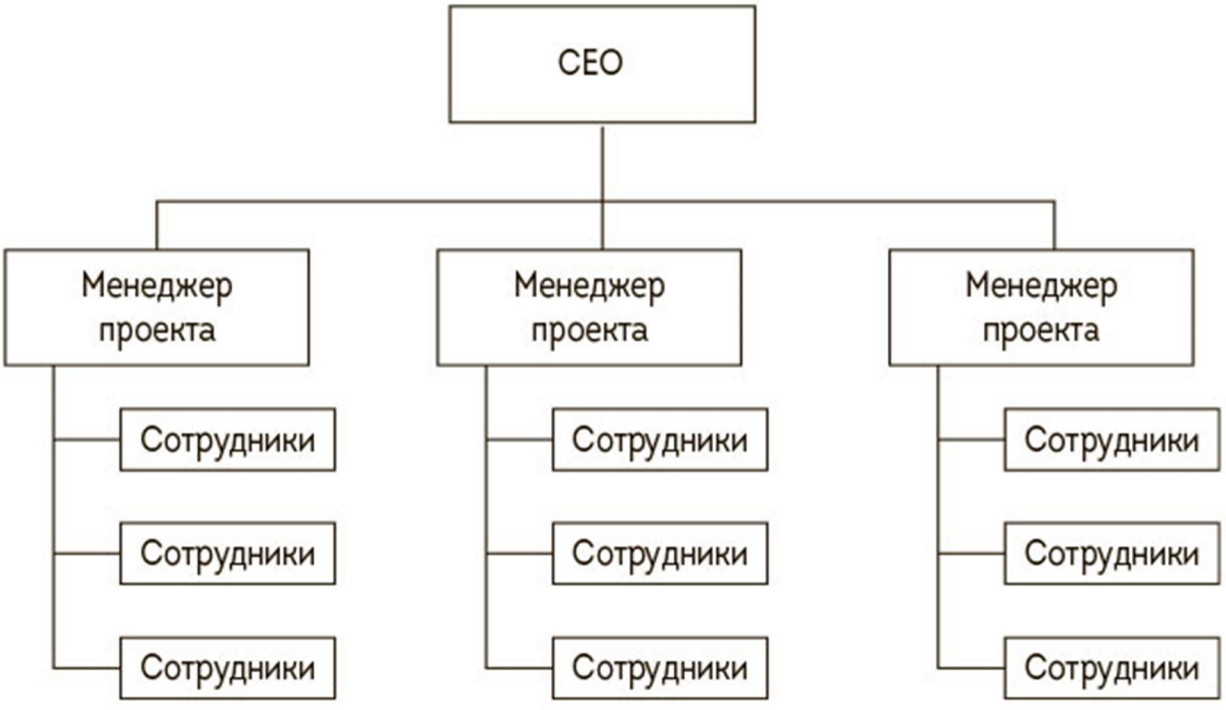

Рис. 9.4. Типичная проектная организационная структура

Сбалансированные матричные организации

Характеристики сбалансированной матричной организации включают следующие:

- Состоит из «вертикальных» профильных отделов и «горизонтальных» проектных команд.
- Полномочия (предположительно) распределены между менеджерами проектов и функциональными менеджерами (см. рис. 9.5).

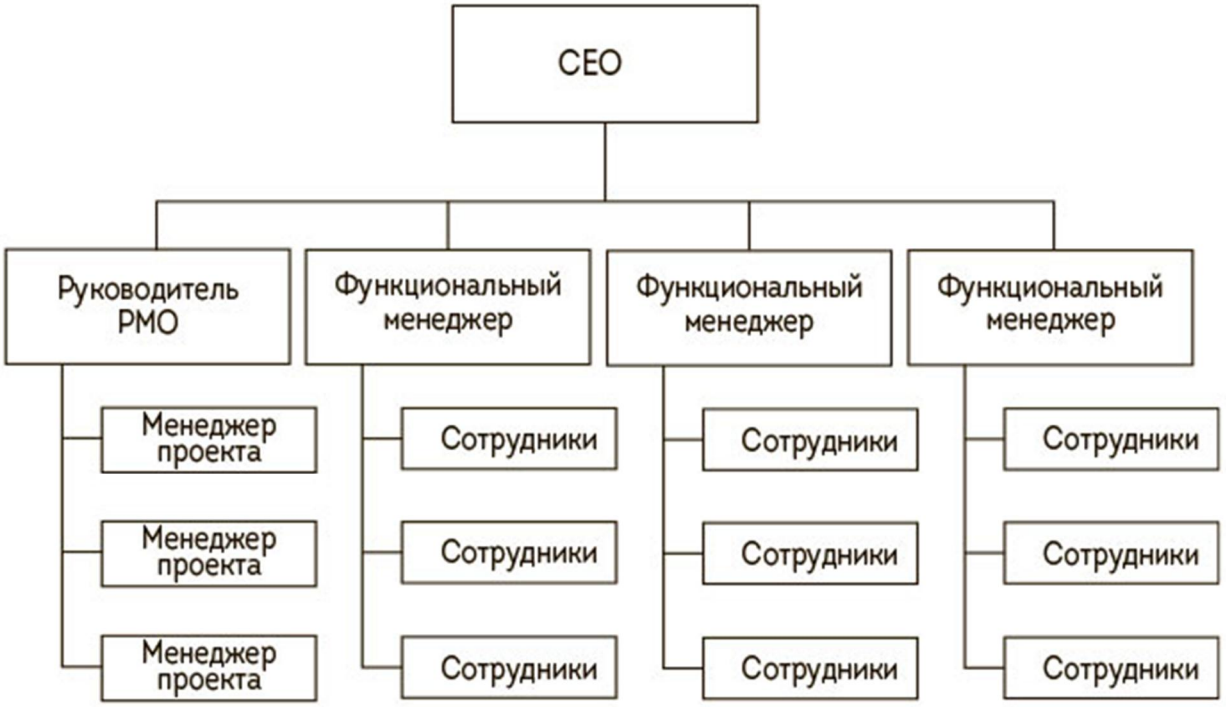

Рис. 9.5. Типичная сбалансированная матричная организационная структура

Выстраивание сбалансированной матричной структуры становится все более распространенным явлением в крупных компаниях. В рамках этой структуры проектами управляют менеджеры проектов, которые на время реализации проекта забирают сотрудников (и другие ресурсы) из-под контроля функциональных менеджеров. После его завершения временная команда будет распущена и все ее участники вернутся к своим функциональным обязанностям либо войдут в другую проектную команду. Для тех, кто не разделяет принципы Agile, подобная организационная структура — золотая середина.

Командно-ориентированные организации

Так почему же сбалансированная матричная структура все еще далека от той, которая нам нужна? Что ж, хотя Scrum является фреймворком

для эффективного управления проектами по программному обеспечению, сам проект не должен восприниматься как центральный компонент (и, конечно, функциональный отдел тоже не должен). Итак, что же является центральным компонентом, вокруг которого должна строиться организационная структура следующего поколения?

Надеюсь, ответ для вас очевиден, но если нет, то это… scrum-команда! В командно-ориентированной компании (см. рис. 9.6) проекты передаются в команды, а не люди собираются под проекты (в зависимости от их загруженности). Agile-коуч Роб Махер отмечает ключевое преимущество командно-ориентированного подхода.

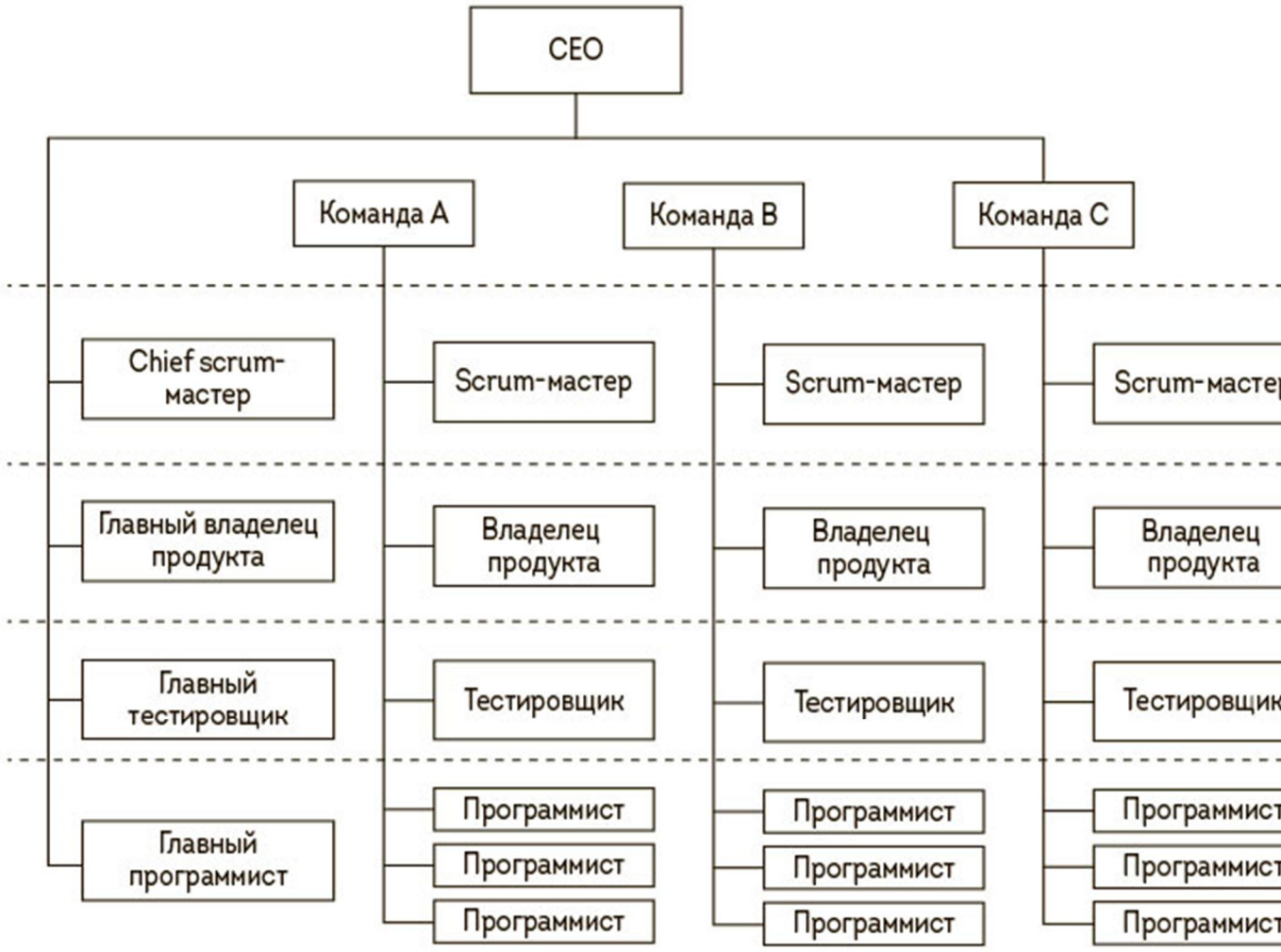

Рис. 9.6. Наиболее подходящая для Scrum организационная структура

Команды проходят проверенный жизненный цикл — формирование (forming), шторм (storming), нормализация (norming) и продуктивная работа (performing)97.

На этот процесс уходит немало времени, часто месяцы. Участники команды узнали свои сильные и слабые стороны, они научились общаться друг с другом, сотрудничать и разрешать конфликты. Зачем же разрушать то, что стало высокоэффективной командой? Есть убедительные доказательства того, что между группами, собранными

на время под проект, и низкой производительностью существует взаимосвязь98.

Помимо важнейшего преимущества — более высокой производительности — у командно-ориентированного подхода есть и другие: возможность делать более точные прогнозы на основе длительных ретроспективных данных о производительности, которые сумела накопить сформировавшаяся scrum-команда (см. лайфхак 19).

Представьте, что командно-ориентированная структура похожа на кроссфункциональные коммандос. Вместо того чтобы формировать новые подразделения для каждой следующей миссии, их поручают наиболее подходящим подразделениям, которые справятся с ними лучше остальных. Конечно, может возникнуть странная ситуация, когда в коммандос должен будет влиться специалист другого типа, но бо́льшая часть подразделения всегда остается вместе в силу сплоченности, доверия и глубоких знаний друг о друге как о соратниках по длительной совместной работе.

МЕНЕДЖЕРЫ ПРОЕКТОВ НЕ ИСЧЕЗАЮТ

Scrum предписывает только три роли: scrum-мастер, владелец продукта и команда разработчиков. Это, казалось бы, упрощенное деление неизбежно приводит к неудобным вопросам: «Хм, а что же случится с нами, с менеджерами проектов?»

Майк Кон предлагает отвечать жестко, чтобы избежать иллюзий: «В Scrum мы признаем роль менеджера проекта несостоятельной и исключаем ее». После следует вопрос (обычно в состоянии паники): «Так что же случится со всеми ключевыми функциями менеджмента проектов, такими как составление графиков работ, бюджетирование и планирование?» Я отвечаю примерно так: «Что ж, на самом деле эти обязанности будут распределены между тремя scrum-ролями в разной степени». Менеджер проекта, который теперь судорожно ищет кислородную маску, продолжает: «Итак… что это значит для меня?» Тогда я деликатно говорю: «Ну это означает, что если вы хотите играть в мире Scrum, то вам придется перейти на должность scrum-мастера, владельца продукта или разработчика — это довольно просто».

Подавленный менеджер проекта часто вздыхает и наконец выдает: «Мне нравится, как звучит слово Scrum, но я люблю быть менеджером проекта и не думаю, что хочу отказаться от этого. Думаю, Scrum не для меня».

Я видел, как это происходит, и мне это не очень нравится. Вместе с водой мы выплескиваем ребенка. Есть ли иной способ привлечь традиционных менеджеров проектов в scrum-проекты? Мое мнение — «да» (с большим «но»). «Но» означает, что в отдельно взятой scrum-команде нет места традиционному менеджеру проекта. Я вижу потенциал для его подключения, только если разработка с

использованием Scrum становится частью более крупного проекта, в котором задействованы несколько отделов. Давайте рассмотрим этот вариант.

Большая картина проекта

Чаще всего scrum-проекты по разработке не осуществляются в организационном вакууме. Как правило, под конец разработки в других отделах компании начинается бурная деятельность. Маркетинговая команда решает, как лучше позиционировать новый продукт. Команда продаж стремится понять продукт для его успешной презентации. Команда сервисного обслуживания готовится обрабатывать запросы клиентов. А финансовая команда должна интегрировать новый прайс и модели доходов на основе нового предложения (см. рис. 9.7). Вот это да! Здесь, конечно, много что необходимо координировать и много чем управлять. Полагаю, что внутриорганизационная координация, логистическое планирование, построение графиков и их отслеживание — это важнейшие функции, с которыми отлично справляется традиционный менеджер проекта.

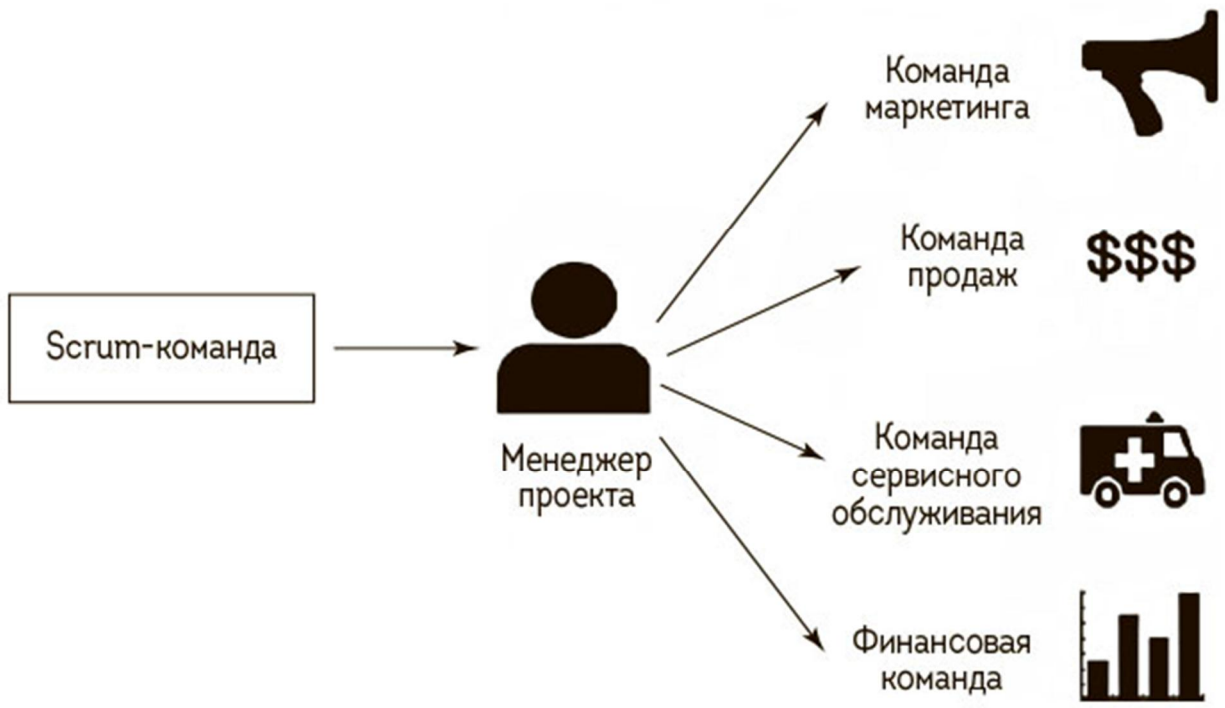

Рис. 9.7. Менеджер проекта может стать важным интерфейсом между scrum-командой(-ами) и другими отделами

Третьи стороны (сторонние организации)

По мнению Кеннета Рубина99, есть еще один вариант, где менеджер проекта может добавить ценности, — это координация усилий между scrum-командой(-ами) и подрядчиками, которые не применяют Scrum.

Менеджер проекта также может быть полезен, когда использование Scrum представляет собой лишь небольшую часть гораздо более

объемной разработки продукта или услуги. Например, при наличии субподрядчиков; внутренних команд, не работающих по Scrum; других внутренних структур, связанных с поставкой продукта.

БУДУЩЕЕ ФУНКЦИОНАЛЬНЫХ МЕНЕДЖЕРОВ

Теперь, когда мы рассмотрели несколько вариантов для менеджера проекта, давайте сфокусируемся на том, какие возможности остаются для функциональных менеджеров после внедрения Scrum.

Вот проблема: отличный разработчик стал функциональным менеджером. Эта позиция дает властные полномочия и привилегии, которые новоявленный менеджер любит и ценит. Конечно, он не может больше работать с кодом столько, сколько хотел бы. Вместо этого ему приходится тратить больше времени на скучные встречи, делегирование задач и возню с документами. Зато он может сказать своей маме, что теперь он начальник, — а это круто, не так ли?

Проблема, однако, в том, что в Scrum нет назначаемого функционального менеджера, которому необходимо распределять работу в самоорганизующихся командах. Вот почему эти менеджеры начинают задаваться вопросом, что же будет с ними. Потеряют ли они свой статус из-за Scrum?

Я в это не верю и вижу решение проблемы в том, чтобы помочь переосмыслить, что значит быть функциональным менеджером. Первое, что я обычно говорю, созвучно словам scrum-тренера и CEO Stormglass Пита Димера100: «Менеджер в Scrum — это не “няня” для команды, а скорее наставник или гуру, который помогает учиться, расти и работать». *Обычно на этом этапе аргументов против у функциональных менеджеров не возникает.*

Далее я говорю, что, если у вас несколько scrum-команд, необходимо определить технические стандарты, а также нацелить команду на устранение технических препятствий как внутри нее, так и между командами. *Наш функциональный менеджер все еще счастлив.*

Далее я говорю, что кому-то нужно будет выявить пробелы в знаниях для разработки соответствующих программ обучения и развития — как на техническом, так и на предметном уровнях. *И здесь пока нет возражений.*

Затем я отмечаю, что нам также потребуется кто-то с соответствующими умениями и квалификацией, чтобы помогать привлекать новых разработчиков, когда мы будем формировать новые команды или нам понадобится кого-то заменить. *По-прежнему никаких возражений.*

Наконец, я говорю функциональным менеджерам, что они могут бросить те неприятные задачи по распределению работ, которые они «вынуждены» делать с низким КПД, и сосредоточиться на том, что они

действительно любят делать! Типичный ответ на это: *«Звучит классно! Когда начинаем?»*

Я рассматриваю этот структурный сдвиг как отход от командования и шаг к формированию сообществ практикующих специалистов. Если эта трансформация пройдет успешно, я не стану возражать против того, чтобы они сохранили позиции функциональных менеджеров.

БУДЕМ РЕАЛИСТАМИ

То, что Scrum предписывает только три роли, не значит, что все остальные — лишние или избыточные. Изменение организационной структуры и массовый отказ от традиционных позиций, таких как менеджер проекта, не только нереалистичны, но и во многих случаях неоправданны. Взглянув непредвзято, мы увидим альтернативы, с помощью которых эти традиционные роли могут быть переосмыслены в рамках истинной модели Scrum, без ее искажения.

Принимая существующий функционал, например управление сложными проектами, мы сможем преодолеть барьер, препятствующий внедрению Scrum, особенно в крупных и традиционных компаниях.

ЗАКЛЮЧЕНИЕ

Три лайфхака этой главы посвящены тактике, инструментам и советам для интеграции стейкхолдеров в scrum-среду. Давайте вспомним, что мы рассмотрели.

Лайфхак 25. Впечатление создает реальность

- важность гибких рабочих отношений со спонсором / спонсорами проекта;
- «одностраничник» — полезный способ сформировать видение продукта;
- советы, благодаря которым спонсоры проекта будут чувствовать себя вовлеченными и оставаться в курсе дел.

Лайфхак 26. Мастера и лорды

- отличие scrum-мастера от chief scrum-мастера;
- основные функции chief scrum-мастера;
- основные функции scrum-мастера.

Лайфхак 27. Трансформация менеджеров в матричной структуре

- почему командно-ориентированная структура предпочтительнее типичной сбалансированной матрицы или функциональных моделей;

- потенциальные возможности для традиционных менеджеров проектов, которые не хотят принимать на себя ни одну из ролей Scrum;
- эволюция роли функциональных менеджеров.

Глава 10

ГЛОБАЛЬНЫЕ УРОКИ

Когда вы напряженно работаете и неумолимо летите вперед, иногда бывает очень трудно увидеть лес за деревьями. В этой заключительной главе сделаем шаг назад и посмотрим на более широкий контекст, чтобы в полной мере оценить то, что на самом деле важно.

Следующие три лайфхака — три последних глобальных урока.

Лайфхак 28 «Оценка развертывания Scrum» исследует выбор макрометрик, фокусируясь на способах оценки развивающегося Scrum и agile-навыков в вашей организации.

Лайфхак 29 «Сосредоточьтесь на главном» объясняет важность самоорганизации для раскрытия потенциала вашей команды.

Наконец, лайфхак 30 «Последний уровень» привлекает внимание к трем фундаментальным понятиям: прозрачности, инспекции и адаптации.

ЛАЙФХАК 28. ОЦЕНКА РАЗВЕРТЫВАНИЯ SCRUM

Если вы прочитали лайфхак 19, то, как и было обещано, здесь вторая часть. Если вы этого не сделали — не беспокойтесь, вы можете читать эти лайфхаки независимо, в любой последовательности. В любом случае большая разница между этим лайфхаком и «Метриками, которые имеют значение» (см. лайфхак 19) заключается в том, что теперь мы готовы подняться на более высокий уровень и обсудить метрики с перспективы развертывания Scrum, а не на уровне scrum-проекта. Тот же совет — используйте метрики для пользы, а не во вред — по-прежнему актуален.

НАСКОЛЬКО МЫ AGILE?

Я иногда слышу, как коллеги по отрасли обсуждают, что их проектная команда «примерно на 85% Agile», или они могут сказать что-то вроде «Мы используем Scrum на 50% его возможностей». Когда я слышу подобные заявления, хочется спросить: «Что, черт возьми, это значит?»

Давайте рассмотрим первое утверждение: «Мы на 85% Agile». Правда? Итак, означает ли это, что если вы сделаете что-то чуть по-другому, возможно начнете внедрять еще несколько практик, то скоро станете на 100% Agile? Увы, вы не можете выполнить что-либо больше, чем на 100%, поэтому, когда вы достигнете этого рубежа, полагаю, вам останется только поддерживать то, что вы делаете, верно? А вот и нет! Я уже говорил это раньше, и вас не нужно убеждать в том, что 100-процентного agile-совершенства никогда не достичь! Всегда будет что-то, что можно сделать лучше. Учитывая, что Scrum в основе своей подразумевает непрерывное улучшение, совершенство в принципе недостижимо. Классифицировать вашу гибкость в процентах просто не имеет смысла.

Я отвечу на второе утверждение (использование Scrum на 50%) кратко: вы либо делаете Scrum, либо нет. Это двоичная система (см. рис. 10.1). Частичная реализация Scrum обсуждалась в лайфхаке 2 через сравнение с шахматами — вы не сможете играть без всех фигур на доске. Точно так же вы не можете использовать Scrum на 50% — если это так, скорее всего, это что-то похожее на Scrum, но точно не Scrum.

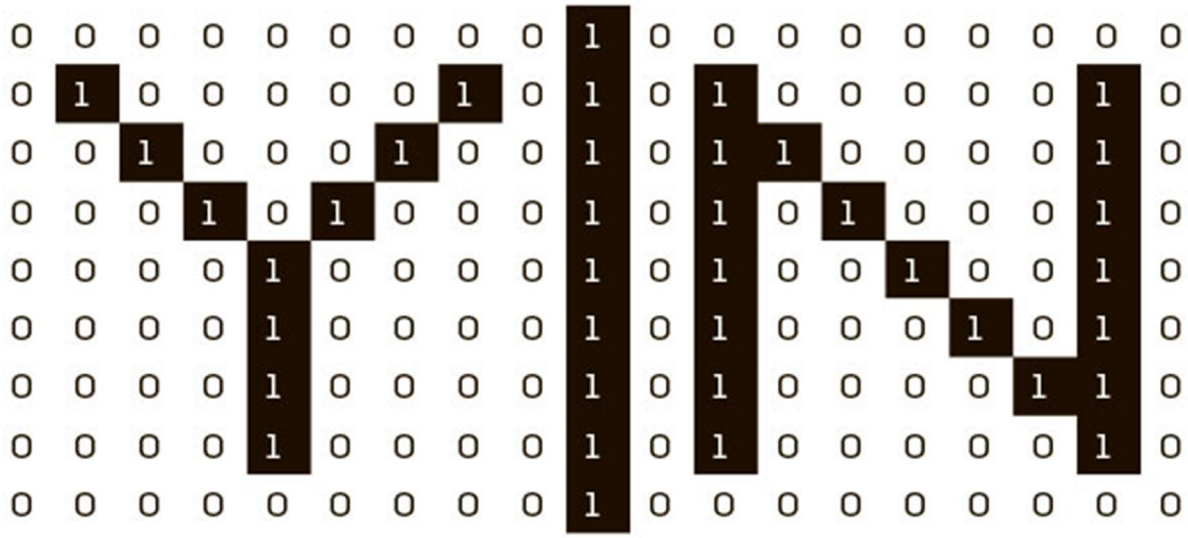

Рис. 10.1. Практика Scrum является бинарной: вы либо работаете по Scrum, либо нет — частичного Scrum не существует

ЛЮДИ ЛЮБЯТ ИЗМЕРЯТЬ

Одержимость измерениями, возможно, начинается тогда, когда наши гордые родители делают первую отметку на стене, отмеряя, насколько мы подросли. Люди любят считать, и мы любим оценивать наш прогресс по самым разным причинам. Это не так плохо, если оценка оправдана: например, если нужно измерить непрерывный процесс и то, что Scrum смог предоставить (особенно по сравнению с первоначальными потенциальными преимуществами, изложенными в лайфхаке 1). Оценка также может быть отличным стимулом для команды, дающим каждому ее участнику ощущение движения вперед и прогресса, — так же как цвет

пояса в боевых искусствах. Конечно, для некоторых цвет пояса важнее искусства, однако для большинства он демонстрирует их улучшения и признание их упорной работы, и это совсем не плохо.

Существует две причины, по которым вам может потребоваться оценка успешности развертывания Scrum:

1. Чтобы решить, следует ли вам продолжать использовать Scrum.
2. Чтобы оценить прогресс вашей команды за время scrum-путешествия.

НУЖНО ЛИ ПРОДОЛЖАТЬ?

Как вы оцениваете, было ли ваше первоначальное развертывание Scrum успешным? Ну вы можете положиться на массив цифр в электронной таблице, сравнивая свои проекты до и после Scrum. Но мне не нравится этот подход (по крайней мере, сам по себе), в первую очередь потому, что Scrum не механика. Он настолько зависит от людей и культуры, что даже при фантастических количественных результатах его внедрение может привести к таким потрясениям, что сделает несчастными слишком многих, а это плачевно скажется на долгосрочном существовании Scrum.

Мне особенно нравится подход, который помогает обогатить любую количественную обратную связь, — это простой индивидуальный опрос в команде, о котором я впервые прочитал в статье scrum-тренера Габриэль Бенефилд101. Чтобы оценить эффективность новаторского глобального развертывания Scrum в Yahoo!, Benefield и Deemer, использовали простой опрос, основанный на следующих шести критериях:

- Сколько команда сделала за 30 дней.
- Ясность целей — что команда должна была поставить.
- Сотрудничество и кооперация внутри команды.
- Бизнес-ценность того, что команда произвела за 30 дней.
- Количество потраченного времени, работы впустую, неправильного использования циклов.
- Общее качество и «правильность» того, что произвела команда.

Все участники опроса имели возможность сопоставить критерии со следующими ответами:

- со Scrum намного хуже;
- со Scrum хуже;
- со Scrum так же;
- со Scrum лучше;
- со Scrum намного лучше.

Результаты этих опросов четко показали позитивное влияние Scrum с точки зрения коллектива.

ИЗДЕРЖКИ ПРОТИВ ПРЕИМУЩЕСТВ

Независимо от чудесных результатов, которые вы надеетесь извлечь из опросов, помните, что они не должны быть единственным индикатором того, ждет ли Scrum успех в вашей организации. В отличие от изолированного пилотного проекта, Scrum не работает в вакууме. Переход от контролируемого испытания к широкому внедрению вызывает ряд новых и часто существенных препятствий, которые отныне необходимо учитывать. Это могут быть размещение кросс-функциональных команд, пересмотр принципов мотивации, изменение планировки офиса, а также пересмотр процедур подписания документов и форм договоров с клиентами — и это лишь часть трудностей.

Суровая реальность такова: в некоторых организациях первоначальное развертывание Scrum может только выявить ограничения, которые будут препятствовать успешному внедрению Scrum в рамках всей компании. Возможно, она не готова, не хочет или не может устранить их.

СТАНОВИТСЯ ЛИ ВАМ ЛУЧШЕ?

Вернемся к хорошим новостям. Допустим, Scrum у вас жив и процветает. Вполне естественно, вам хочется определить: вы стоите на месте или у вас прогресс.

Чтобы оценить это, вам понадобится эталон, привязанный либо к вашей прошлой эффективности, либо к вашей результативности в сравнении с другими (особенно конкурентами).

К счастью для всех нас, Майк Кон из компании Mountain Goat Software и Кенни Рубин из компании Innolution сняли бо́льшую часть проблем, создав веб-сайт Comparative Agility. Вот как они это объясняют:

В Comparative Agility мы предполагаем, что agile-команды и организации всегда стремятся стать лучше своих конкурентов и самих себя в прошлом. Иными словами, нет никакого святого Грааля или «высшей отметки» для оценки достижений. Не существует заранее заданных параметров для лучшего в своем роде или «agile-зрелости пятого уровня». Оценка в Comparative Agility выставляется по результатам опроса в сравнении с другими наборами ответов. Так с помощью Comparative Agility можно сравнить команду, проект или организацию по следующим параметрам:

- общий набор собранных ответов;
- ответы компаний из одной отрасли;
- ответы по аналогичным проектам (например, коммерческое программное обеспечение, веб-сайты и т. д.);

- ответы по проектам с подобным опытом перехода на Agile.

75 вопросов для оценки Comparative Agility разделены на 7 категорий и 32 характеристики. Семь категорий представляют широкую классификацию изменений, которые следует ожидать от команды или организации по мере становления Agile. Вот они:

- работа в команде;
- требования;
- планирование;
- технические практики;
- качество;
- корпоративная культура;
- создание знаний.

Каждая категория состоит из трех-шести параметров и определяется по набору вопросов для оценки команды (для каждой характеристики отдельно)[102].

Опрос удобен для пользователя и дает возможность в увлекательной манере оценить относительный прогресс.

Я уверен в таких инструментах, как оценка Comparative Agility, которую создали и поддерживают ведущие эксперты. Тем не менее должен предупредить вас: если сравнение с другими является единственным критерием, который вы используете, по сути, вы позволяете другим оценивать ваш прогресс, что может привести к стагнации. Например, если вы проходите опрос и понимаете, что ваши дела сравнительно хороши на фоне других компаний, вы можете оказаться в плену ложного чувства безопасности. Во-первых, ваши конкуренты, возможно, не прошли опрос. Во-вторых, что еще хуже, вы можете почувствовать себя царем горы, наслаждаться своим положением и потерять темп, убрав ногу с педали непрерывного улучшения, что никогда и никому не идет во благо.

БУДЬТЕ ПРОЩЕ

Если вы действительно не хотите ничего усложнять, задайтесь тремя вопросами:

1. Стали ли ваши клиенты счастливее — они остаются лояльными и покупают больше продуктов?
2. Стали ли участники вашей команды счастливее — они бегут с корабля или больше улыбаются?
3. Стали ли ваши стейкхолдеры счастливее — они более расслаблены и позволяют команде выполнять свою работу, а не занимаются микроменеджментом?

Метрика, которая действительно должна вас интересовать, — это скорость внедрения Scrum. Дает ли Scrum такие замечательные результаты, что теперь он начинает широко распространяться по организации, преодолевая пропасть и проникая в отделы, не связанные с программным обеспечением? Для меня это и есть успех!

РАСПРОСТРАНЯЯ ХОРОШИЕ НОВОСТИ

В Австралии широко распространено мнение, что мы, австралийцы, не любим праздновать успех — эта черта, верите или нет, даже имеет название: tall-poppy syndrome[103]. Простота и скромность порой восхищают, однако при внедрении изменений празднование успеха имеет решающее значение для поддержания импульса, воодушевления и поощрения тех, кто принял на себя риски. В таком случае я говорю: «Забудьте о скромности, пусть весь мир узнает, как хорошо идут дела!» Проводите регулярные встречи команды, чтобы обсудить прогресс. Приглашайте на них главных стейкхолдеров. Разошлите письма по электронной почте, как описано в лайфхаке 25, разбавив сухую статистику достижений юмором и веселыми историями. Пойте о своих успехах с крыш! Мне все равно, как вы это сделаете, но вы должны праздновать непрерывное улучшение и позволить команде и компании разделить триумф.

ЛАЙФХАК 29. СОСРЕДОТОЧЬТЕСЬ НА ГЛАВНОМ

Подростком я был одержим игрой в футбол (или соккер — для моих приятелей из Австралии и друзей в Северной Америке).

Во время обеденного перерыва в школе наша группа выходила на поле и играла чрезвычайно напряженные товарищеские матчи. Мы быстро разбивались на две команды и с головой уходили в игру. Когда я оглядываюсь назад на эти приятные игры, я поражаюсь той удивительной химии, которая, казалось, автоматически возникала между нами. Каким-то образом мы сразу выясняли, на какой позиции нам хотелось бы войти в игру. Без униформы мы могли определить, кто в какой команде, просто постоянно переговариваясь.

Самоотверженность была невероятной (учитывая, что мы бились за нашу честь), мы действовали на удивление слаженно, и, казалось, каждый знал, где окажутся остальные в следующую минуту. Это происходило почти телепатически — если кто-то из защитников не успевал вернуться к своим воротам, остальная команда бежала назад, чтобы помочь (даже если они не играли в защите). Но лучше всего было товарищество — побеждая или проигрывая, каждый из нас всегда поддерживал другого.

Мне повезло: у меня была возможность играть в сборной штата на национальном уровне. Мои ожидания перед первым подобным турниром были невероятно высоки, ведь я должен был сыграть с лучшими и против лучших из лучших со всей страны! У нас были шикарная униформа, суперталантливые профессиональные игроки, опытный тренер с большим количеством впечатляющих планов и огромная личная мотивация (в национальную сборную попадут игроки по итогам этого турнира). Итак, как все прошло? Мы отыграли как звезды, которыми себя считали? Нет, очень посредственно! Почему? Вы спросите, что пошло не так? Ну мы ссорились и указывали на малейшую ошибку. Наш тщательный план на игру провалился, стоило только одному игроку получить травму. Мы плохо знали друг друга, поэтому не могли использовать сильные стороны друг друга. Командная работа была забыта — каждый старался показать себя. Из-за позиционной игры подмога не поспевала к тем, кто оказывался в меньшинстве.

Жесткий подход к игре, основанный на указаниях и контроле, в сочетании со звездами на поле оказался гораздо менее эффективным, чем наши самоорганизующиеся матчи в обеденный перерыв.

Как эта история соотносится с нашими scrum-командами? Читая об этом в конце книги, вы уже понимаете, какой объем работы выполняет scrum-мастер. За ворохом повседневных задач очень трудно сосредоточиться на святом Граале — создании по-настоящему самоорганизующейся команды, сродни крутым футбольным командам, которыми мы, школьники, становились в обеденный перерыв.

ОБЪЯСНЯЯ САМООРГАНИЗАЦИЮ

Давайте теперь формализуем термин «самоорганизация» (ироничный выбор слов, я знаю), поскольку часто скептики неверно толкуют его как нечто больше похожее на бедлам! Мне нравится описание Кеннета Рубина[104]:

Самоорганизация — эмерджентное свойство сложной адаптивной системы, пронизывающее ее снизу доверху. В таких системах объекты взаимодействуют друг с другом по-разному, и эти взаимодействия регулируются простыми локализованными правилами, работающими с учетом постоянной обратной связи. Эти типы систем обладают любопытными характеристиками, такими как невероятная прочность и генерирование поразительных инноваций.

Над командой разработчиков нет центра принятия решений, ей не указывают, как выполнять работу. Вместо этого кросс-функциональная команда организовывает сама себя наиболее подходящим способом, чтобы выполнить работу. Самоорганизация чрезвычайно сильная вещь. Как показал мой пример с футболом, примеры можно находить снова и снова, просто наблюдая за природой. Посмотрите на колонию

муравьев, улей или стаю птиц, и вы увидите самоорганизующееся поведение. Лисса Адкинс[105] приводит замечательный пример из мира развлечений, описывая свое изумление при наблюдении за игрой струнного квартета:

Четверо исполнителей сели на свои места и, не подав друг другу никакого сигнала, не сделав даже глубокого вдоха, начали играть. Во время игры они без посторонней помощи управляли музыкой и подстраивались друг под друга. Они не нуждались в дирижере.

РАБОЧАЯ СРЕДА И ГРАНИЦЫ

Всякий раз, когда вы пытаетесь объяснить преимущества самоорганизации скептику, не игнорируйте отсылки на границы. Скептики, проводящие параллели между самоорганизующимися командами и группой безмозглых цыплят, должны понимать, что ответственные участники команды лучше всех могут определить, как выполнять свою работу и когда, по их мнению, она будет выполнена. Руководитель, который любит раздавать указания, на самом деле не настолько компетентен, чтобы делать это, по сравнению с командой, реально выполняющей работу. Навязывая свои условия, он добивается лишь одного — иллюзии контроля над организацией.

При этом участники команды не оказываются как по волшебству в состоянии свободной самоорганизации. Как любое растение, самоорганизацию нужно выращивать и культивировать в конкретной среде с четкими границами, иначе оно выйдет из-под контроля и перемахнет через забор соседа. Эти границы не появляются сами по себе и не остаются заброшенными. Именно scrum-мастер должен постоянно взаимодействовать с руководством, чтобы «создать достаточно точек контроля с целью не допустить превращения нестабильности, двусмысленности и напряженности в хаос»[106].

Эти точки контроля, или границы, могут быть самыми разными (см. рис. 10.2). Давайте рассмотрим пять важнейших из них.

Рис. 10.2. Будь то футбольная или scrum-команда, самоорганизация приводит к лучшим результатам и большему удовольствию!

Правила Scrum

Правила прежде всего. Хотя нет никакой scrum-полиции, которая выпрыгнет из-за офисной кофеварки и оштрафует вас за несоблюдение scrum-правил, крупный штраф может оказаться хорошим вариантом, если ни правила, ни дисциплина не соблюдаются. Если вы хотя бы раз видели, что происходит после ошибки футбольного судьи, то понимаете: нечто подобное может возникнуть и в scrum-команде, если правила становятся неоднозначными или часто меняются. При строгом применении (небольшого) набора scrum-правил у самоорганизующейся команды возникают четкие границы, что позволяет ей сосредоточиться на своих задачах.

Состав команды

Участники команды редко сами решают, кто вольется в их ряды, даже если им предстоит работать вместе, чтобы самоорганизоваться. Если вы получите дисфункциональную команду неподходящих друг другу людей или неправильное сочетание навыков, то самоорганизация пойдет псу под хвост (см. лайфхак 5). Даже при правильном подборе участников в команде будут возникать конфликты. Scrum-мастеру

предстоит помогать руководителям набирать команду, чтобы обеспечить природную склонность к самоорганизации и совместной работе.

Рабочая среда

В лайфхаке 3 мы обсуждали рабочую среду. Очевидно, до тех пор пока вы не создадите условия, обеспечивающие нахождение всех членов команды рядом, будет очень сложно достичь самоорганизации (хотя это все еще будет посильной задачей). Ограниченное, но открытое пространство, способствующее естественному и регулярному общению, создает физические границы, позволяющие самоорганизующейся команде легко контактировать без искусственных, формализованных каналов связи.

Корпоративная культура

Если люди стремятся скрывать свои ошибки и уличать других, будьте уверены, шансы достичь статуса самоорганизующегося объекта находятся где-то между нолем и нулем. Scrum-мастер и руководство отвечают за предоставление команде безопасной гавани, в которой людям было бы комфортно работать вместе (см. лайфхак 24). Подобная корпоративная культура также призвана способствовать коллективному желанию команды постоянно искать возможности для улучшения и апробировать новые идеи без страха потерпеть неудачу.

Требования

Основополагающий принцип Scrum гласит: владелец продукта (через бэклог продукта) определяет, что следует разрабатывать и в каком порядке (это не происходит по усмотрению команды разработчиков). Кроме того, согласованные критерии готовности (см. лайфхак 11) устанавливают четкие критерии качества, гарантируя, что ожидания владельца продукта и команды разработчиков будут выровнены.

Убедитесь, что разработчики понимают важность ограничений, присущих бэклогу продукта, и критериев готовности. Именно это предоставит команде свободу самоорганизовываться и вырабатывать требования оптимальным способом.

БЕСКОНЕЧНАЯ РОЛЬ

Меня очень тревожат прекраснодушные восклицания вроде этого: «Как великие scrum-мастера, мы всегда должны стремиться сделать себя устаревшими», ведь после того, как команда достигла самоорганизации, scrum-мастер становится лишней фигурой. Это чепуха, и вот почему.

Во-первых, изменения неизбежны. Как феноменальная спортивная команда не останется в неизменном составе навсегда, точно так же и блестящую самоорганизующуюся scrum-команду ждут перемены. Люди

идут дальше (и уходят из команды), берут отпуска, компании подвергаются реструктуризации, что приводит к разделению команд. Там, где возможны изменения, scrum-мастер необходим.

Во-вторых, трудности непредсказуемы. Никто не знает, какие проблемы, возникшие из ниоткуда, команда не сможет решить, независимо от того, насколько она самоорганизована. Кроме того, есть препятствия, которые должна решать вовсе не команда, а scrum-мастер, чтобы максимум времени уходило на разработку.

Наконец, совершенство недостижимо. Ничто в мире не является идеальным, всегда можно что-то улучшить, даже у зрелых самоорганизующихся команд. Точная настройка и помощь командам в непрерывном улучшении намного сложнее (но потенциально более полезны), чем быстрые улучшения, которые легко проследить для менее зрелых команд.

ЛАЙФХАК 30. ПОСЛЕДНИЙ УРОВЕНЬ

Да, я знаю, вы не раз и не два слышали, что Scrum «трудный и разрушительный»[107]. Уверен, многие из вас испытали это на собственной шкуре.

Не могу не согласиться с этим утверждением, ведь Scrum, безусловно, отягощен множеством условий для трансформации. Необходимо провести немало изменений, например: выстроить сплоченность кросс-функциональной команды; обеспечить полномочия владельца продукта; объединить высокоэффективные самоорганизующиеся команды; обеспечить, чтобы главные стейкхолдеры поддерживали системные изменения.

Однако я склонен смотреть на Scrum иначе. Для меня он абсолютно безошибочен и всегда беспроигрышен. Звучит странно, не правда ли? Что ж, позвольте объяснить.

ГЛЯДЯ В ЗЕРКАЛО

Помните, в лайфхаке 12 я обсуждал, что возможность постоянно инспектировать и адаптировать работу, выполняемую в спринте, похожа на то, как команда часто смотрит в зеркало, чтобы обнаружить мелкие проблемы на ранней стадии? Перенесем этот принцип на макроуровень. Посмотрим правде в глаза: у вашего scrum-проекта может ничего не выйти. Он может провалиться по ряду причин. Однако, если вы будете дисциплинированны, будете следовать правилам Scrum и придерживаться ключевых практик, в худшем случае Scrum станет для вас зеркалом, помогающим обнаружить сбои, приводящие к срыву проекта. Если вы можете определить их, это уже достижение! Прямо как со здоровьем. Если вы плохо себя чувствуете, разве вы не захотите

узнать, что с вами не так, чтобы выбрать план действий и вылечиться? Scrum похож на выдающегося доктора, способного выявлять и диагностировать проблемы, о которых вы даже не подозревали.

ВЫБЕРИ СВОЕ ПРИКЛЮЧЕНИЕ

Пожалуйста, помните, что Scrum — это легкий фреймворк с несколькими ключевыми правилами и ограниченным набором практик. В этой книге я, безусловно, не пытаюсь изложить закон об обязательной реализации Scrum. Я всего лишь предлагаю несколько интерпретаций и привожу разные мнения, основываясь на конкретном опыте. Решите ли вы прислушаться к моим рекомендациям или нет, это не повлияет на то, что вы делаете в Scrum. Пока вы остаетесь в рамках фреймворка, можете выбрать приключение по своему усмотрению. Подобные книги просто путеводители. Все, что вам нужно для отпуска, — паспорт, немного денег, понимание правил путешествия и знание местных законов. За пределами этого вы можете делать все, что вам нравится. Путеводители необязательны, но они, безусловно, могут помочь!

ЭКСПЕРИМЕНТ

В лайфхаке 24 я предложил несколько советов, как помочь пессимистам преодолеть страх перемен. Это настолько важно, что я повторю еще раз со всей серьезностью в этом последнем лайфхаке. Есть только три слова, которые вы должны вынести из этой книги:

ПРОЗРАЧНОСТЬ

ИНСПЕКЦИЯ

АДАПТАЦИЯ

Эти три столпа эмпирического управления процессами составляют основу Scrum, и, если твердо их уяснить, вы всегда будете в выигрыше. Людям не нравятся перемены по многим причинам. Прежде всего потому, что им кажется, что, приняв новое, они лишат себя пространства для маневра. Но вот в чем дело: Scrum — открытая система. Для того, что не работает, есть очень простое правило: инспекция и адаптация. В открытой и прозрачной системе проверять будет легко. А регулярно проверяя, вы сможете эффективно адаптироваться. Адаптация может включать корректировку, возврат к старому подходу или попытку внедрить что-то совершенно новое. Принятие Scrum следует рассматривать как последовательность небольших экспериментов (действий по итогам ваших обзоров и ретроспектив), составляющую масштабный эксперимент (возможно, изначально запущенный пилот Scrum). Если Scrum интерпретировать именно так, никаких огорчений не будет, ведь в итоге благодаря прозрачности, инспекции и адаптации будет найден наиболее подходящий путь, чтобы вывести вашу команду и всю компанию на новый уровень результативности. Как сказал

теоретик системного анализа Ричард Бакминстер Фуллер: «Каждый раз, проводя новый эксперимент, человек всегда узнает больше. Он не может узнать меньше»108.

НЕ ПОЧИВАЙТЕ НА ЛАВРАХ

Некоторые из вас могут чувствовать себя счастливыми: вы сыграли важную роль в запуске Scrum, он работает, и все улыбаются от уха до уха. Время сидеть сложа руки и ждать наступления хороших времен, правильно? Неправильно! Вспомните лайфхак 4! Безусловно, вы можете быть довольны текущими достижениями, но вы никогда не должны чувствовать, что все выполнено и завершено. Помните: совершенства не существует, поэтому всегда будет что улучшить. Кроме того, в динамичной организации ничто и никто не стоит на месте. Трудности могут появиться в любой момент, участники команды могут двигаться дальше или меняться, а отделы — объединяться или разделяться. Вы должны видеть всю среду как постоянно меняющуюся, неважно, насколько спокойной она может казаться в конкретный момент.

Движение требует мягкой направляющей силы, способной позитивно реагировать на изменения, разглаживая складки на ткани команды, чтобы они не стали глубокими морщинами. Поскольку на всех уровнях происходит ротация сотрудников, вам придется организовать постоянное обучение внутри компании (я говорю не только о тех, кто участвует в разработке программного обеспечения). Ваша конечная цель — убедить каждого сотрудника вашей организации в том, что «гибкость должна стать бизнес-стратегией, а не просто оставаться тем, что делают IT-специалисты»109. Если Agile станет частью ДНК и корпоративной культуры всей организации, благодаря эффекту домино реализация scrum-проектов станет гораздо более естественной.

ПРЕВОСХОДЯ ОЖИДАНИЯ

Забавно, но когда вы погружаетесь в Scrum, то можете ощутить захватывающий побочный эффект — Scrum проникает в другие сферы вашей жизни, становясь способом мышления и поведения.
В лайфхаке 11 я упомянул, что использовал Scrum с его артефактами и активностями, чтобы справляться с домашними обязанностями после рождения нашего первого ребенка. И я не остановился на этом. Меня захватил дух Scrum. Я стараюсь работать (и играть) в ритмичном устойчивом темпе. Я сосредотачиваюсь на выполнении задачи, будь то домашние обязанности или работа в целом, а не тщетно жонглируя пятью вещами одновременно (моя старая привычка). Мне больше не нужны трехлетние жизненные планы — я позволяю своему будущему реализовываться по гибкому сценарию. Сегодня проявления авторитарного поведения на работе или дома заставляют меня вздрагивать, хотя в прошлом я вполне мог быть тем командиром,

который раздавал указания и контролировал их выполнение. Теперь же я стремлюсь использовать принципы убеждения (которые принял как scrum-мастер), близкие боевому искусству айкидо. В этом виде спорта вместо того, чтобы применять силу самому, вы используете агрессию противника против него самого. Теперь я спрашиваю себя, как я могу получить согласие других, не возвращаясь к авторитарному давлению и жестким командам. Также я регулярно провожу ретроспективы о самом себе — обычно утром по дороге на работу, не дожидаясь 31 декабря, чтобы перечислить все нереалистичные и быстро забываемые новогодние обещания.

ПОДВЕДЕМ ИТОГИ

Отправляясь в свое scrum-путешествие, я ожидал, что оно изменит работу моих команд к лучшему. Но, конечно, я не думал, что это может изменить мои принципы и поведение в целом. Тем не менее именно это и произошло. Scrum не только сделал меня лучше как коллегу и лидера, он сделал меня лучше как человека (я действительно верю в это).

Теперь моя цель — вывести трансформационные возможности Scrum далеко за пределы программного обеспечения, и я надеюсь, вы присоединитесь ко мне.

Благодарю, что позволили поделиться с вами своими мыслями, и желаю безопасного и приятного scrum-путешествия.

БЛАГОДАРНОСТИ

Со смирением я признаю, что недооценил усилия, необходимые для превращения моих расплывчатых мыслей в книгу. Оглядываясь назад, я все еще не могу поверить, как все в конечном итоге сложилось! Написание книги требует особой сосредоточенности, систематической работы и широты взглядов. Но самое главное, оно требует помощи. Поддержка приходила ко мне в разных видах и формах, и без нее не было бы книги. Много людей делали все возможное и невозможное, чтобы помочь книге стать такой, какой вы ее видите сегодня. И я чрезвычайно благодарен всем, кто протянул мне руку помощи на этом пути.

Давайте начнем с самого начала. В первую очередь хочу поблагодарить моего главного помощника, Колина Тана — делового партнера, арт-дизайнера, лучшего друга и первого, кто редактировал книгу. Он первым убедил меня поделиться scrum-мыслями с миром, без него вы бы не читали эту книгу — вот так. Уверен, вы согласитесь, что его творческий подход добавил уникальное, креативное измерение этой книге, которое слова сами по себе не могут создать.

Я хотел бы выразить глубокую признательность Супермену, также известному как Майк Кон. Я называю его Суперменом по нескольким причинам. Сейчас все знают, какое огромное положительное влияние он оказал на scrum-мир своими фундаментальными прорывными книгами и участием в сообществе, но немногие знают, что он также является чемпионом по пауэрлифтингу и может жать от груди 250 кг! Я не шучу! Оставив в стороне обсуждение сверхспособностей, скажу, что приглашение написать книгу для серии Майка является таким карьерным достижением, которое будет невероятно трудно превзойти.

Спасибо Крису Гузиковски, Оливии Базеджо и Крису Зану из издательства Pearson, которые помогли мне пройти через незнакомый мир издательского дела и терпеливо отвечали на все мои вопросы. Я благодарю вас за все. Кроме того, я хотел бы поблагодарить Кэрол Лальер и Элизабет Райан, которые проделали замечательную работу, собрав все вместе.

Я хотел бы сказать спасибо остальным авторам серии Signature: Лиссе Адкинс, Юргену Апелло, Лайзе Криспин, Джанет Грегори, Клинтону Кейту, Роману Пихлеру и Кенни Рубину не только за то, что они приняли меня в команду, но и за то, что вдохновляли задолго до написания этой книги.

Особенно я хотел бы поблагодарить Кенни Рубина за все ценные советы, которые он дал мне, завершив фантастическую книгу Essential Scrum в то же время, когда я приступил к своей.

Теперь к рецензентам: я всегда буду благодарен за то время, которое вы выкроили из своего напряженного графика, чтобы поделиться мнением и высказать свои мысли. Прежде всего я говорю спасибо Сесилу Голдштейну, своему отцу. Могу поспорить, ты считал, что дни проверки домашних заданий давно прошли. Тем не менее ты благополучно вернулся в строй, твои замечания были, как всегда, ценны.

Я также хотел бы выразить признательность тем, кто давал мне полезную обратную связь, пока писалась эта книга: Кевину Остину, Джоэл Бэнкрофт-Коннорс, Жереми Беназра, Чарльзу Брэдли, Марио Куеве, Питу Димеру, Райану Дорреллу, Каролайн Гордон, Дагу Джейкобсу, Рону Джеффризу, Мартину Кернсу, Джой Келси, Ричарду Каупинсу, Арику Когану, Венкатешу Кришнамурти, Джону Мэддену, Кейну Мару, Йенсу Мейдаму, Николасу Малдуну, Брайану О’Доновану, Майклу Рембаху, Мэтту Роаднайту, Питеру Сэддингтону, Лайзе Шуп, Крейгу Смиту, Хьюберту Смитсу, Михаэлю Штанге, Рене Траутон и Джейсону Ипу.

Я благодарю все scrum-сообщество и более обширное agile-сообщество за то, что вы поддерживаете такую удивительную культуру открытости и всегда готовы делиться идеями. Я искренне надеюсь, что эта книга будет способствовать позитивному влиянию, которое сообщество оказывает на преобразование мира труда.

Наконец, хочу поблагодарить мою удивительную жену Кармен, сделавшую все от нее зависящее и освободившую для меня каждый отрезок времени Pomodoro110, чтобы я писал в тисках хаоса, проходя наш первый круг родительства! Ты действительно Scrummy Mummy! И спасибо маленькой Эми, позволявшей мне высыпаться и в итоге писать хотя бы полусвязно. Но в основном просто за то, что ты служишь удивительным источником вдохновения и придаешь смысл моей жизни.

ОБ АВТОРЕ

Илан Голдштейн — известный австралийский agile-эксперт с более чем десятилетним практическим опытом. Он всемирно признанный сертифицированный scrum-тренер (CST), работает со стартапами, лидерами рынка, государственными учреждениями и публичными компаниями по всему миру, помогая им повысить свою гибкость с помощью Scrum. Илан Голдштейн регулярно выступает на конференциях, является приглашенным преподавателем в нескольких университетах и основателем AxisAgile — ведущей компании — поставщика обучающих продуктов и консалтинговых agile-услуг. Кроме этого, Илан — учредитель национальной некоммерческой организации Scrum Australia, призванной развивать и обогащать scrum-сообщество Down Under.

Илан — преданный практик Scrum, наслаждавшийся временем, которое он проводил как scrum-мастер, разработчик и владелец продукта (конечно, не одновременно, а в проектах разных компаний и отраслей). Он прекрасно осознает, что требуется для превращения теории в практику, и щедро делится этими знаниями (как в этой книге, так и при обучении Scrum, которое проводит по всему миру).

Илан — обладатель нескольких профессиональных сертификатов, дополняющих «боевые раны», которые он получил, добиваясь многочисленных scrum-побед на «поле боя». К ним относятся:

- сертифицированный scrum-тренер (Certified Scrum Trainer, CST);
- сертифицированный scrum-профессионал (Certified Scrum Professional, CSP);
- сертифицированный scrum-мастер (Certified Scrum Master, CSM);
- сертифицированный владелец продукта (Certified Scrum Product Owner, CSPO);
- профессионал по управлению проектами (Project Management Professional, PMP);
- сертифицированный agile-практик (Agile Certified Practitioner, PMI-ACP).

Он живет в Сиднее, в Австралии, с женой Кармен и дочерью Эми и свободное время отдает волонтерству для Compeer — международной

некоммерческой организации, помогающей людям с психическими заболеваниями проходить терапию и поддерживающей их на пути к выздоровлению. Дополнительно про автора вы можете прочитать на www.axisagile.com. Илана можно найти в «Твиттере» как @ilagile, а также связаться с ним по электронной почте ilan@axisagile.com.

ПРИМЕЧАНИЯ

1. Дилберт (*англ.* Dilbert) — название серии комиксов и имя главного героя — инженера IT-компании, который прозябает в офисном рабстве, продирается через бессмысленный набор ценностей и миссию компании, страдает от недальновидного руководства и безуспешно пытается создать креативную команду. Первая публикация состоялась 16 апреля 1989 года, комикс Скотта Адамса выходил в 65 странах на 25 языках. По его мотивам был снят анимационный сериал. *Прим. ред.*

2. Генри Лоуренс Гантт (*англ.* Henry Laurence Gantt) — соратник отца научного менеджмента Фредерика Тейлора, он изучал менеджмент на примере строительства кораблей и предложил полосатую диаграмму для визуализации плана-графика работ по проекту. *Прим. ред.*

3. Кеннет Рубин тренирует и обучает Scrum и agile-разработке, помогая сотрудникам компаний эффективно и экономично разрабатывать программные продукты. Являясь сертифицированным инструктором, Кеннет обучил гибкой разработке по методике Scrum более 18 тысяч человек. Он тренировал специалистов из более чем 200 компаний. *Прим. ред.*

4. *Rubin K.S.* Essential Scrum: A Practical Guide to the Most Popular Agile Process. Ann Arbor: Addison-Wesley Professional, 2012; издана на русском: *Рубин К.С.* Основы Scrum. Практическое руководство по гибкой разработке ПО. М. : Вильямс, 2016.

5. Фреймворк (*англ.* framework — каркас, структура) — набор базовых элементов и правил, на которых строится тот или иной процесс. *Прим. ред.*

6. https://www.scrumalliance.org/ScrumRedesignDEVSite/media/Scrum-AllianceMedia/Files%20and%20PDFs/Learn%20About%20Scrum/Core-Scrum.pdf.

7. *Schwaber K., Sutherland J.* The Scrum Guide. 2011. Downloadable at www.scrum.org.

8. *Deemer P., Benefield G., Larman G., Vodde B.* The Scrum Primer, Version 2.0. 2010. www.scrumprimer.com.

9. *Rubin K.S.* Essential Scrum: A Practical Guide to the Most Popular Agile Process. Ann Arbor: Addison-Wesley Professional, 2012; издана на русском: *Рубин К.С.* Основы Scrum. Практическое руководство по гибкой разработке ПО. М. : Вильямс, 2016.

10. *Schwaber K.* Scrum Fails? // Ken Schwaber's Blog: Telling It Like It Is. 2011, April 7. Доступно по ссылке: http://kenschwaber.wordpress.com/2011/-04/07/scrum-fails/.

11. *Takeuchi H., Nonaka I.* The New New Product Development Game // Harvard Business Review. 1986. Доступно по ссылке: https://hbr.org/1986/-01/the-new-new-product-development-game.

12. *Cohn M.* Introduction to Scrum Methodology. 2007, April 12. Презентация доступна по ссылке: http://scrumalliance.org/resources/47.

13. *Rubin K.S.* Essential Scrum: A Practical Guide to the Most Popular Agile Process. Ann Arbor : Addison-Wesley Professional, 2012; издана на русском: *Рубин К.С.* Основы Scrum. Практическое руководство по гибкой разработке ПО. М. : Вильямс, 2016.

14. «Водопад», или waterfall, — метод управления проектами, в котором все этапы работы идут последовательно один за другим: ни один из этапов не должен быть пропущен, а следующий этап не начинается, пока не закончится предыдущий. *Прим. ред.*

15. *Schwaber K.* Scrum Fails? // Ken Schwaber's Blog: Telling It Like It Is. 2011, April 7. Доступно по ссылке: http://kenschwaber.wordpress.com/2011/-04/07/scrum-fails/.

16. В текущем спринте они тестируют результат работы программистов из предыдущего спринта. *Прим. ред.*

17. *Schwartz T.* Why Appreciation Matters So Much // Harvard Business Review. 2012, January 23. Доступно по ссылке: http://blogs.hbr.org/-schwartz/2012/01/why-appreciation-matters-so-mu.html.

18. *Spolsky J.* Smart and Gets Things Done. Joel Spolsky's Concise Guide to Finding the Best Technical Talent. Berkeley : Apress, 2007.

19. *DeMarco T., Lister T.* Peopleware: Productive Projects and Teams. Dorset House, 1999. Издано на русском: *Демарко Т., Листер Т.* Человеческий фактор. Успешные проекты и команды. М. : Символ-Плюс, 2005.

20. В соответствии с правилом «20% времени» каждый сотрудник Google имел право 20% рабочего времени посвящать сторонним проектам, не связанным с его основным функционалом. *Прим. ред.*

21. *Пинк Д.* Драйв. Что на самом деле нас мотивирует. М. : Альпина Паблишер, 2013.

22. *Greenleaf R.K.* The Servant as Leader. Westfield: Greenleaf Center for Servant Leadership, 2008.

23. *Rubin K.S.* Essential Scrum: A Practical Guide to the Most Popular Agile Process. Ann Arbor: Addison-Wesley Professional, 2012.

24. *Owsinski B.* The Studio Musician's Handbook (Music Pro Guides). New York : Hal Leonard, 2009.

25. В соответствии с переводом «Руководства по Скраму» (The Scrum Guide) (2017), размещенном по адресу https://www.scrumguides.org/docs/-scrumguide/v2017/2017-scrum-guide-russian.pdf.

26. На русском языке: *Юрген А.* Agile-менеджмент. Лидерство и управление командами. М. : Альпина Паблишер, 2019.

27. *Owsinski B.* The Studio Musician's Handbook (Music Pro Guides). New York : Hal Leonard, 2009.

28. *Cohn M.* Succeeding with Agile: Software Development Using Scrum. Addison-Wesley, 2009. На русском языке издана: *Кон М.* Scrum. Гибкая разработка ПО. Описание процесса успешной гибкой разработки программного обеспечения с использованием Scrum. М. : Вильямс, 2016.

29. *Rubin K.S.* Essential Scrum: A Practical Guide to the Most Popular Agile Process. Ann Arbor: Addison-Wesley Professional, 2012.

30. *Кон М.* Scrum. Гибкая разработка ПО. Описание процесса успешной гибкой разработки программного обеспечения с использованием Scrum. М. : Вильямс, 2016.

31. *Shore J., Warden S.* The Art of Agile Development. O'Reilly Media, 2007.

32. *Poppendieck M., Poppendieck T.* Leading Lean Software Development: Results Are not the Point. Boston : Addison Wesley, 2009.

33. *Пихлер Р.* Управление продуктом в Scrum. Agile-методы для вашего бизнеса. М. : Манн, Иванов и Фербер, 2016.

34. *Бек К.* Экстремальное программирование. Разработка через тестирование. М. : Питер, 2017.

35. *Кон М.* Scrum. Гибкая разработка ПО. Описание процесса успешной гибкой разработки программного обеспечения с использованием Scrum. М. : Вильямс, 2016.

36. *Пихлер Р.* Управление продуктом в Scrum. Agile-методы для вашего бизнеса. М. : Манн, Иванов и Фербер, 2016.

37. *Cohn M.* Introduction to Scrum Methodology. A downloadable presentation from the Scrum Alliance website. 2007, April 12. http://scrum-alliance.org/resources/47.

38. Имеется в виду при парном программировании. *Прим. ред.*

39. Падающая сборка — процесс сборки проекта, во время которого произошли ошибки, проект не пересобрался на сервере, и на сервере осталась старая версия проекта. *Прим. ред.*

40. «Прогрессирующее откровение» — одна из богословских концепций, согласно которой господь давал свое откровение постепенно, раскрывая его через каждого следующего своего пророка и посланника (от Ветхого Завета к Новому). *Прим. ред.*

41. *Кон М.* Пользовательские истории. Гибкая разработка программного обеспечения. М. : Вильямс, 2018.

42. *Kniberg H.* Lean from the Trenches: Managing Large-Scale Projects with Kanban. Pragmatic Bookshelf, 2011.

43. *Keith C.* Agile Game Development with Scrum. Boston : Addison-Wesley, 2010.

44. Боевой сервер — сервер, на котором приложение исполняется для конечных пользователей. *Прим. ред.*

45. Smoke-тесты — минимальный набор тестов на явные ошибки, чтобы убедиться, что критически важные функции работают согласно ожиданиям. *Прим. ред.*

46. Бесплатный калькулятор Майка Кона доступен по ссылке: www.mountaingoatsoftware.com/tools/velocity-range-calculator.

47. Про Angry Birds можно многое узнать по адресу http://en.wikipedia.org/-wiki/Angry_Birds.

48. *Grenning J.* Planning Poker Party (the Companion Games) // James Grenning's Blog. 2009, February 6. www.renaissancesoftware.net/blog/-archives/36.

49. *Crispin L., Gregory J.* Agile Testing: A Practical Guide for Testers and Agile Teams. Addison-Wesley, 2009. На русском книга была издана: *Криспин Л.*, *Грегори Д.* Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд. М. : Вильямс, 2016.

50. *Crispin L., Gregory J.* Agile Testing: A Practical Guide for Testers and Agile Teams. Addison-Wesley, 2009. На русском книга была издана: *Криспин Л.*, *Грегори Д.* Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд. М. : Вильямс, 2016.

51. *Crispin L., Gregory J.* Agile Testing: A Practical Guide for Testers and Agile Teams. Addison-Wesley, 2009. На русском книга была издана: *Криспин Л.*, *Грегори Д.* Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд. М. : Вильямс, 2016.

52. «Слон в комнате» — английская идиома, означающая очевидную правду / обстоятельства / факты, которые невозможно не заметить, но все старательно делают вид, что не замечают. *Прим. пер.*

53. *Кон М.* Scrum. Гибкая разработка ПО. Описание процесса успешной гибкой разработки программного обеспечения с использованием Scrum. М. : Вильямс, 2016.

54. Интуитивное тестирование — вид тестирования, который выполняется без подготовки, без определения ожидаемых результатов и без проектирования тестовых сценариев. *Прим. ред.*

55. «Горилла»-тестирование — высоконагруженное тестирование одного конкретного модуля, чтобы определить предел его работы. *Прим. ред.*

56. *Crispin L., Gregory J.* Agile Testing: A Practical Guide for Testers and Agile Teams. Addison-Wesley, 2009. На русском книга была издана: *Криспин Л.*, *Грегори Д.* Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд. М. : Вильямс, 2016.

57. Оракул — это эвристический механизм, который помогает определить проблему. *Прим. пер.*

58. *Shore J., Shane W.* The Art of Agile Development. Pragmatic Guide to Agile Software Development. O'Reilly Media, 2007.

59. *Бек К.* Экстремальное программирование. Разработка через тестирование. М. : Питер, 2017.

60. *Fowler M.* Continuous Integration. From Martin Fowler's website. 2006, May 1. http://martinfowler.com/articles/continuousIntegration.html.

61. *Fowler M.* Continuous Integration. From Martin Fowler's website. 2006, May 1. http://martinfowler.com/articles/continuousIntegration.html.

62. *Mar K.* Scrum 101 — Scrum and Extreme Programming (XP). July 13 2012. www.youtube.com/watch?v=Pav4YxhsQbc.

63. *Crispin L., Gregory J.* Agile Testing: A Practical Guide for Testers and Agile Teams. Addison-Wesley, 2009. На русском книга была издана: *Криспин Л.*, *Грегори Д.* Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд. М. : Вильямс, 2016.

64. *Crispin L., Gregory J.* Agile Testing: A Practical Guide for Testers and Agile Teams. Addison-Wesley, 2009. На русском книга была издана: *Криспин Л.*, *Грегори Д.* Гибкое тестирование. Практическое руководство для тестировщиков ПО и гибких команд. М. : Вильямс, 2016.

65. *Jeffries R.* Which End of the Horse? // XProgramming.com: An Agile Software Development Resource. 2010, December 24. http://-xprogramming.com/articles/which-end-of-the-horse/.

66. Более подробную информацию о FitNesse см.: http://en.wikipedia.org/-wiki/Fitnesse.

67. Более подробную информацию о Selenium см.: http://en.wikipedia.org/-wiki/Selenium_(software).

68. *Кон М.* Scrum. Гибкая разработка ПО. Описание процесса успешной гибкой разработки программного обеспечения с использованием Scrum. М. : Вильямс, 2016.

69. *Shore J., Warden S.* The Art of Agile Development. O'Reilly Media, 2007.

70. Мок-объекты — объекты, которые заменяют реальный объект в условиях тестирования. *Прим. пер.*

71. *Фарли Д., Хамбл Д.* Непрерывное развертывание ПО. Автоматизация процессов сборки, тестирования и внедрения новых версий программ. М. : Вильямс, 2016.

72. *Чапел Х., Брукс Ф.* Мифический человеко-месяц, или Как создаются программные системы. М. — СПб. : Символ-Плюс, 2007. С. 20.

73. Подробнее об альтернативной scrum-диаграмме сгорания релиза Майка Кона можно узнать на сайте www.mountaingoatsoftware.com/-scrum/alt-releaseburndown/.

74. Имеется в виду «чтобы достичь целей спринта». *Прим. ред.*

75. *Yip J.* It's Not Just Standing Up: Patterns for Daily Standup Meetings. 2011, August 29. URL: http://martinfowler.com/articles/itsNotJustStanding-Up.htm.

76. *Schwaber K.* Agile Project Management with Scrum (Microsoft Professional). Microsoft Press, 2004.

77. *Keith C.* Agile Game Development with Scrum. Addison-Wesley, 2010.

78. *Silverman R.E.* No More Angling for the Best Seat; More Meetings Are Stand-Up Jobs // Wall Street Journal. 2012, February 2.

79. *Conway M.* Why Do Committees Invent? // Datamation. 1968. Vol. 14. No 4. P. 28–31.

80. Симуляционная среда — тестовая среда, в которой проверяют работоспособность программного обеспечения перед выгрузкой нового функционала в производство (или на «боевые» сервера). Симуляционная среда должна быть максимально приближена по условиям к «боевой». *Прим. ред.*

81. Чтобы узнать больше о Microsoft PowerPoint, перейдите по ссылке: http://en.wikipedia.org/wiki/Microsoft_PowerPoint.

82. Чтобы узнать больше о Apple Keynote, перейдите по ссылке: http://-en.wikipedia.org/wiki/Keynote_(presentation_software).

83. Деплой — развертывание, помещение исполняемого кода на «боевой» сервер, где он будет работать. *Прим. ред.*

84. Affinity mapping — одна из техник проведения мозгового штурма. *Прим. ред.*

85. *Дерби Э., Ларсен Д.* Agile-ретроспектива. Как превратить хорошую команду в великую. М. : Издательство Дмитрия Лазарева, 2017.

86. *Cohn M.* Succeeding with Agile: Software Development Using Scrum. Addison-Wesley, 2009.

87. *Пихлер Р.* Управление продуктом в Scrum. Agile-методы для вашего бизнеса. М. : Манн, Иванов и Фербер, 2016. С. 116–117.

88. *Senor D., Singer S.* Start-up Nation: The Story of Israel's Economic Miracle. Grand Central Publishing, 2009. Издано на русском: *Сенор Д., Сингер С.* Нация умных людей. История израильского экономического чуда. М. : Карьера Пресс, 2011.

89. Английское издание вышло в 2013 г. и содержит устаревшие сведения. Население Израиля в 2019 г. превысило 9 миллионов человек. *Прим. ред.*

90. *Moses A.* Brain Drain: Why Young Entrepreneurs Leave Home // Sydney Morning Herald. 2012. May 18.

91. *Bosworth A.* Facebook Engineering Bootcamp // Facebook Engineering's Notes. 2009, November 20. www.facebook.com/note.php?note_id=177577-963919.

92. *Cohn M.* Succeeding with Agile: Software Development Using Scrum. Addison-Wesley, 2009.

93. *Fowler G.A., Morrison S* eBay Attempts to Clean Up the Clutter // Wall Street Journal. 2010, October 31.

94. *Кон М.* Scrum. Гибкая разработка ПО. Описание процесса успешной гибкой разработки программного обеспечения с использованием Scrum. М. : Вильямс, 2016.

95. *Кон М.* Scrum. Гибкая разработка ПО. Описание процесса успешной гибкой разработки программного обеспечения с использованием Scrum. М. : Вильямс, 2016.

96. Сотрудников компании можно подразделить на функциональные команды — например, разработчиков, тестировщиков или scrum-мастеров. Chief scrum-мастер берет на себя HR-функционал: помогает составлять индивидуальные планы развития scrum-мастеров, развиваться им карьерно и профессионально; разрабатывать систему вознаграждений scrum-мастеров; обучать scrum-мастеров; нанимать и увольнять и т. д. *Прим. ред.*

97. *Tuckman B.W.* Developmental sequence in small groups // Psychological Bulletin. 1965. Vol. 63(6). P. 384–399.

98. *Maher R.* Increasing Team Productivity: A Project Focus Creates Waste and Leaves Value on the Table // Scrum.org Whitepaper retrieved from the Scrum.org website. www.scrum.org/Portals/0/Documents/Community%20-Work/Increasing%20Team%20Productivity.pdf.

99. *Rubin K.S.* Essential Scrum: A Practical Guide to the Most Popular Agile Process. Ann Arbor : Addison-Wesley Professional, 2012; издана на русском: *Рубин К.С.* Основы Scrum. Практическое руководство по гибкой разработке ПО. М. : Вильямс, 2016.

100. *Deemer P.* Manager 2.0: The Role of the Manager in Scrum. Доступно по ссылке: www.goodagile.com/resources/roleofthemanager10.pdf.

101. *Benefield G.* Rolling Out Agile in a Large Enterprise // Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008).

102. *Cohn M., Rubin K.* What is Comparative Agility? // Comparative Agility. 2010. Доступно по ссылке: www.comparativeagility.com/overview.

103. Tall-poppy syndrome (*австр. слэнг* «синдром высокого мака») — склонность критиковать успешных людей. *Прим. пер.*

104. *Rubin K.S.* Essential Scrum: A Practical Guide to the Most Popular Agile Process. Ann Arbor : Addison-Wesley Professional, 2012; издана на русском: *Рубин К.С.* Основы Scrum. Практическое руководство по гибкой разработке ПО. М. : Вильямс, 2016.

105. *Adkins L.* Coaching Agile Teams: A Companion for Scrum-Masters, Agile Coaches, and Project Managers in Transition. Boston : Addison-Wesley, 2010; издана на русском: *Адкинс Л.* Коучинг agile-команд. Руководство для scrum-мастеров, agile-коучей и руководителей проектов в переходный период. М. : Манн, Иванов и Фербер, 2017. С. 307–308.

106. *Takeuchi H., Nonaka I.* The New New Product Development Game // Harvard Business Review. 1986. Доступно по ссылке: https://hbr.org/1986/-01/the-new-new-product-development-game.

107. *Schwaber K.* Scrum Is Hard and Disruptive. Доступно по ссылке: www.controlchaos.com/storage/scrum-articles/Scrum%20Is%20-Hard%20and%20Disruptive.pdf.

108. *Fuller R.B.* Operating Manual for Spaceship Earth / Ed. by J. Snyder. Lars Müller Publishers, 2008.

109. *Kearns M.* Agile and Portfolio/Program Management. Session presented at Agile Australia 2012, Melbourne.

110. Метод «помидора» — техника управления временем, предложенная Франческо Чирилло в конце 1980-х годов. Предполагает разбиение задач на 25-минутные отрезки, называемые «помидоры», с короткими перерывами между ними. *Прим. ред.*

НАД КНИГОЙ РАБОТАЛИ

Главный редактор *Артем Степанов*

Шеф-редактор *Ренат Шагабутдинов*

Ответственный редактор *Анна Красова*

Литературный редактор *Елена Ефремова*

Арт-директор *Алексей Богомолов*

Дизайн обложки *Наталья Майкова*

Верстка *Вячеслав Лукьяненко*

Корректоры *Светлана Липовицкая, Надежда Петрив*

ООО «Манн, Иванов и Фербер»

mann-ivanov-ferber.ru

Электронная версия книги подготовлена компанией Webkniga.ru, 2020