Содержание

Предисловие научного редактора

Привет, друзья!

Меня зовут Илья Павличенко, я занимаюсь скрамом и аджайл-разработкой в течение последних девяти лет. Скрам — моя жизнь и моя любовь. Являюсь сертифицированным тренером от компании Scrum.org, которую основал Кен Швабер, помогаю компаниям становиться гибкими.

Когда мне предложили курировать перевод этой книги, я бегло пробежал по манускрипту и растерялся: скрам эволюционировал за последние 15 лет, и его современное определение в «Руководстве по скраму» конфликтовало с отдельными частями этой книги. Тогда мы связались с Кеном и получили от него разрешение на внесение правок, чтобы привести текст в соответствие с современным руководством. Теперь вы держите в руках старую-новую книгу, в которой учтены последние изменения в скраме.

Интерес к фреймворку скрам в России и мире просто огромный. Большое количество компаний от маленьких стартапов до больших корпораций используют его в качестве секретного оружия. Убежден, что скрам — это настоящее и как минимум ближайшее будущее компаний, которые хотят стать адаптивными. Если вы хотите научиться быстро менять направление разработки своих

продуктов и сервисов, выстраивать доверительные отношения с клиентами и обходить конкурентов, то эта книга для вас.

Я рекомендую прочитать книгу, потому что она несет в себе дух скрама, раскрывая его ценности и основные принципы. Они не меняются со временем: в основе скрама лежит гуманистическая психология, эмпирический контроль и самоорганизация. Скрам — это новый управленческий подход для создания компаний нового типа, в которых люди могут увлеченно работать, развиваться, при этом доставляя продукты высочайшей ценности и качества на рынок.

Большую пользу от прочтения книги получат собственники компаний, менеджеры среднего и высшего звена. Скрам — это новый подход к управлению бизнесом, соответствующий вызовам и трендам XXI века.

Книга также будет полезна скрам-мастерам и аджайл-коучам. Она должна изучаться ими буквально через лупу: Кен на личном примере показывает действия скрам-мастера в запутанных ситуациях, открывая трансформационный смысл этой роли. Кен не боится ошибаться: он открыто признает ошибки и вместе с читателями проводит работу над ними.

Я желаю вам приятного чтения и даже немного завидую, потому что вас ожидают удивительные открытия и несколько увлекательных часов чтения, проведенных вместе с Кеном и его книгой.

Scrum ON!

Илья Павличенко,
скрам-мастер, коуч ACC,
первый в России сертифицированный скрам-тренер
от Scrum.org и первый в мире русскоязычный LeSS-тренер

Предисловие Майка Кона

На самом деле мой новый босс не был придурком, но в тот момент именно так и казалось. Мы разрабатывали новое программное обеспечение (ПО) для крупных колл-центров компании. Я сказал, что нам понадобится 12 месяцев, а он решил дать мне только четыре, чтобы новым ПО не обязательно можно было бы начать пользоваться, но чтобы оно было готовым к выводу в промышленную среду за 30 дней после соответствующего уведомления. Таким образом, через четыре месяца мне придется поддерживать программное обеспечение в состоянии 30-дневной готовности к релизу. Мой начальник понимал, что не все функции появятся через четыре месяца, он просто хотел получить как можно больше и как можно быстрее.

Мне нужно было найти процесс, который позволил бы нам сделать это. Изучив о процессах разработки ПО все, что получилось найти, я остановился на ранних работах Кена Швабера о скраме. За прошедшие после этого проекта годы я использовал скрам для создания как коммерческих продуктов, так и программного обеспечения для внутреннего использования, консалтинговых проектов, проектов по стандарту ISO 90011* и других. Каждый из них уникален, но все они были срочными и критически важ-

* ISO 90011 — международный стандарт, содержащий требования к системам менеджмента качества. — *Здесь и далее, за исключением специально оговоренных случаев, прим. пер.*

ными для организации. Скрам идеально подходит для таких проектов: требования часто меняются или они неизвестны, поэтому их невозможно уточнить. Скрам позволяет командам преуспеть в таких условиях.

В этой книге Кен пишет, что скрам трудно применять. Трудно не из-за того, что вы делаете, а из-за того, чего вы не делаете. Если вы — менеджер проекта, вам, возможно, будет не хватать некоторых привычных инструментов. В скраме нет диаграмм Ганта и отчетов о потраченном времени, вы не назначаете задачи программистам. Вместо этого вы изучаете несколько простых правил скрама и то, как использовать частые циклы инспекции и адаптации, чтобы быстрее создавать более ценное программное обеспечение.

Кен Швабер создал скрам вместе с Джеффом Сазерлендом. В этой книге вы узнаете о многих скрам-проектах, в которых участвовал Кен. Он часто выступает на профессиональных конференциях. Если вы когда-либо слышали, как он говорит, то знаете, что он не стесняется в выражениях. Эта книга такая же: Кен рассказывает как об успехах, так и о неудачах проектов. Его цель — научить нас делать проекты успешными, поэтому он приводит примеры для подражания и ситуации, которых лучше избегать. Эта книга ясно отражает опыт Кена в наставничестве скрам-команд и преподавании курсов для скрам-мастеров по всему миру. С помощью множества историй Кен делится с нами десятками усвоенных им уроков.

Эта книга — прекрасное руководство для тех, кто хочет улучшить процесс разработки программного обеспечения. Я настоятельно рекомендую ее!

Майк Кон,
сертифицированный тренер скрам-мастеров,
один из основателей Agile Alliance и Scrum Alliance,
автор книг о скраме и пользовательских историях

Предисловие от Мэри Поппендик: почему скрам работает

Допустим, я добираюсь из Чикаго в Бостон самолетом. До и во время полета капитан судна получает инструкции от авиадиспетчерской службы. Мы взлетаем по команде и следуем по заданному маршруту. В ходе полета компьютеры будут с точностью до минуты предсказывать время приземления в Бостоне. Если что-то меняется — скажем, плотность воздуха, — пилот должен получить разрешение на переход на другую высоту. При приближении к аэропорту пилоту сообщают, на какую полосу садиться и у каких ворот парковаться.

Если я отправлюсь в Бостон на машине, то смогу поехать когда захочу и как захочу. Я не знаю точного времени прибытия и, скорее всего, не буду планировать, по какому маршруту поеду и где остановлюсь на ночь. В пути я следую правилам дорожного движения: останавливаюсь на красные сигналы светофоров, придерживаюсь стиля вождения другого региона, соблюдаю ограничения скорости, передвигаюсь вместе с потоком. В автомобиле я — независимый агент, принимающий решения в своих собственных интересах в соответствии с правилами игры.

Я не перестаю удивляться, как тысячи тысяч людей каждый день путешествуют на машинах, достигают своих целей в рамках простых правил дорожного движения, без центра управления или диспетчерской службы. Меня также удивляет, что если я захочу отправить посылку, то могу ввести запрос на сайте почтовой службы и водитель прибудет к моей двери к указанному времени. Диспетчер не направляет водителя в каждый дом: водитель получает постоянно обновляемый список адресов и временных окон и должен самостоятельно составить маршрут так, чтобы забрать все посылки вовремя.

Для обеспечения доставки на следующий день достаточно центральной системы диспетчеризации, которая планирует маршрут водителя единожды в начале дня. Но невозможно заранее спланировать маршрут, если клиенты в любое время дня оставляют запросы на отправку посылки сегодня же. Системы центрального управления и диспетчеризации по мере увеличения комплексности выходят из строя. Самоотверженно пытаясь заставить их работать, некоторые люди применяют более жесткий подход — и это действительно помогает, но лишь на время и не в каждом случае. Таксопарки регулируют новые запросы через диспетчерские центры. Некоторые службы доставки закрепляют водителей в районах, отправляют им запросы и позволяют самостоятельно определять оптимальный маршрут на основе текущей ситуации. Однако в долгосрочной перспективе выигрывают те, кто осознает необходимость перехода к системе независимых агентов, действующих в соответствии с набором правил.

Чем более комплексной является система, тем выше вероятность сбоя центрального управления. По этой причине компании децентрализуются, а правительства отменяют регулирование. Отказ от контроля над независимыми агентами — это проверенный временем подход к решению комплексных проблем. Чем выше комплексность проекта, тем острее необходимость делегировать принятие решений независимым агентам, непосредственно выполняющим работу. Скрам предлагает проторенную дорожку для перехода от централизованных диспетчеризации и управления расписанием к отдельно работающим командам.

Еще одна причина успешной работы скрама заключается в том, что он значительно сокращает цикл обратной связи между заказ-

чиком и разработчиком, между списком пожеланий и их реализацией, между инвестициями в продукт и их возвратом. Опять же, существенную роль здесь играет комплексность. Когда система проста, нетрудно заранее определить, что делать. Но, имея дело с постоянно меняющейся рыночной экономикой и идущими вперед технологиями, получение новых знаний через короткие циклы исследований и проверки гипотез — проверенный подход к решению проблем.

Мы давно знаем этот подход: проводим различные маркетинговые кампании и выясняем, какой подход работает; моделируем поведение транспортного средства во время проектирования автомобиля, чтобы обнаружить оптимальный наклон капота и лучшее распределение веса. Практически все подходы к улучшению процессов используют некоторую версию цикла Деминга–Шухарта для изучения проблемы, проведения экспериментов с решением, измерения полученных результатов и применения проверенных улучшений. Мы называем это «принятием решений на основе фактов» и знаем, что это работает намного лучше, чем предиктивные подходы.

Скрам основан на спринтах — коротких циклах обучения длительностью до одного месяца, которые подтверждают бизнес-гипотезы. Если все уже известно и нечего открывать, то, возможно, нам не следует использовать скрам. Однако если нам нужно учиться, то настойчивость скрама в предоставлении завершенного инкремента, добавляющего бизнес-ценность, помогает нам учиться быстрее. Преимущество завершенных полноценных инкрементов заключается в том, что мы точно знаем, какую ценность для бизнеса приносят наши эксперименты. Частичные ответы часто вводят нас в заблуждение, заставляя думать, что подход будет работать, хотя при более тщательном рассмотрении мы убедимся, что в действительности он не работает. Скрам побуждает нас тестировать и интегрировать наши эксперименты, а затем выпускать ПО в промышленную среду, проходя полный цикл обучения каждый спринт.

Скрам фокусируется на предоставлении не просто инкремента бизнес-ценности, а на предоставлении бизнес-ценности с наивысшим приоритетом, определенным заказчиком или владельцем продукта. Владелец продукта и команда обсуждают ценность

требований для бизнеса, а затем команда определяет, что сможет сделать в течение следующего спринта. При этом короткая петля обратной связи внутри спринта становится петлей обратной связи для бизнеса: скрам часто и рано проверяет, будет ли разрабатываемая система приносить ценность и как именно будет выглядеть эта ценность. Это позволяет последовательно создавать программную систему, приносящую ценность в соответствии с нашим текущим актуальным и основанным на фактах мнением.

Еще одна причина успешной работы скрама заключается в том, что он раскрывает интеллектуальный потенциал сотрудников. Зачастую, когда что-то идет не так, вокруг есть люди, которые знали о потенциальной проблеме, но почему-то их идеи не были учтены. Например, когда при возвращении на Землю 1 февраля 2003 года взорвался космический шаттл «Колумбия», многие предполагали, что инженеры знали о возможных проблемах, но не смогли добиться серьезного отношения руководства NASA к своим опасениям.

По словам Гэри Конвиса, экс-президента Toyota Motor Manufacturing Kentucky, роль менеджеров в здоровой, процветающей рабочей среде заключается в том, чтобы «формировать организацию не силой воли или диктата, а, скорее, собственным примером, коучингом, пониманием и помощью другим в достижении их целей»*.

Скрам превращает участников небольших команд в менеджеров своей судьбы. Мы знаем, что несем ответственность за собственный выбор и обязательно найдем способ добраться в Бостон, если будем самостоятельно выбирать маршрут. Мы станем объезжать ремонтные работы и избегать пробок в часы пик, принимать решения на лету, адаптируясь к независимым решениям других водителей. Аналогичным образом скрам-команды принимают вызов, а затем совместно выясняют, как действовать. Они обходят препятствия такими творческими способами, которые не могли быть спланированы центральным контрольно-диспетчерским центром.

Если размер команд способствует активному вовлечению каждого участника и они чувствуют, что контролируют свою судьбу,

---

* Gary Convis, “Role of Management in a Lean Manufacturing Environment”, “Learning to Think Lean”, August 2001, SAE International. — *Прим. авт*.

то опыт, идеи и опасения отдельных участников будут максимально использованы, а не проигнорированы. Когда участники команды разделяют общую цель, в которую все верят, они находят способ, как ее достичь. Когда команды понимают бизнес-ценность и берут на себя ответственность предоставить ее своим клиентам, когда они свободны решать, как выполнять задачи, когда им предоставляются все необходимые ресурсы, они добиваются успеха.

Гэри Конвис отмечает, что устойчивый успех Toyota обусловлен «взаимосвязанным набором трех основных элементов: философскими основами, управленческой культурой и техническими инструментами. Философские основы включают в себя ориентацию на клиента, акцент в первую очередь на людей, приверженность постоянному совершенствованию. Культура управления основана на нескольких факторах, включая развитие и поддержание чувства доверия, обязательство прежде всего вовлекать тех, кого ситуация затрагивает непосредственно, командную работу, равное и справедливое отношение ко всем и, наконец, принятие решений на основе фактов и анализа влияния действий в долгосрочной перспективе»*.

Скрам успешно работает по тем же причинам. Его философские основы направлены на расширение возможностей команды разработчиков и удовлетворение потребностей клиентов. Его управленческая культура основана на помощи другим в достижении их целей. Его технические инструменты направлены на принятие решений, основанных на получаемых в процессе обучения фактах. Обладая всеми этими характеристиками, скраму трудно не преуспеть.

Мэри Поппендик,
Poppendieck LLC

---

* Gary Convis, “Role of Management in a Lean Manufacturing Environment”, “Learning to Think Lean”, August 2001, SAE International. — *Прим. авт.*

Благодарности

Огромное спасибо моей дочери Кэри Швабер, которая превратила мои слова в ясно изложенный текст, а также Майку Кону и Мэри Поппендик за их незаменимую помощь в поддержании лаконичности и сфокусированности этой книги.

Вступление

Я представляю вам скрам — самый парадоксальный и вызывающий вопросы процесс управления комплексными проектами. С одной стороны, скрам обезоруживающе прост и легок для изучения: в нем немного практик, артефактов и правил. С другой стороны, простота скрама может быть обманчива, потому что скрам — не предписывающий подход. Он не описывает, в каких обстоятельствах и что делать. Скрам используется для работы в комплексном окружении, когда невозможно предсказать, что произойдет дальше. Соответственно, скрам просто предлагает фреймворк и набор практик, поддерживающих прозрачность происходящего. Благодаря этому команды, использующие скрам, точно знают, что происходит, и могут вносить коррективы на ходу, чтобы проект достигал выбранных целей.

Здравый смысл — это комбинация знаний, опыта, умеренности, смекалки и интеллекта. Использующие скрам люди применяют здравый смысл всякий раз, когда обнаруживают, что работа отклоняется от пути, ведущего к желаемым результатам. И все же большинство из нас так привыкли использовать предписывающие процессы — те, которые говорят: «Сделай это, затем сделай это, а затем сделай вот это», — что мы научились пренебрегать здравым смыслом и просто ждать инструкций.

Я написал эту книгу, чтобы помочь людям понять, как использовать скрам для решения комплексных задач. Вместо описания

процесса, ролей, артефактов, правил и практик скрама представляю вам набор реальных рабочих ситуаций, в которых люди используют скрам для разрешения комплексных проблем и выполнения комплексной работы. В некоторых из этих примеров действующие лица используют скрам правильно, и рассматриваемый проект в конечном итоге достигает своих целей. В других случаях люди сопротивляются скраму, и их проекты оказываются в итоге менее успешными.

Для тех, кто сопротивлялся, скрам не был интуитивно понятен. Я постарался разобраться, как такое может быть возможно, ведь скрам — очень простой процесс управления комплексными проектами. В сравнении со многими традиционными подходами к управлению проектами скрам практически не требует усилий. По крайней мере, сначала я так думал.

Большинство ответственных за управление проектами людей обучались предопределенному подходу к управлению проектами, в котором используются подробные планы, диаграммы Ганта и расписания. А скрам — полная противоположность. В отличие от традиционных инструментов, которые пытаются побороть естественное течение проекта, скрам демонстрирует менеджерам, как оптимально вести проект по курсу, который изменяется на ходу. Я слышал, что путешествие по кривой обучения начинается с момента, когда новичок должен все продумывать шаг за шагом, и заканчивается, когда он может выполнять новую работу неосознанно. Это особенно верно в отношении скрама, потому что люди, погруженные в традиционные методы управления, должны отучиться от многих привычных практик и инструментов.

Недавно я помог одной компании по разработке программного обеспечения начать использовать скрам. Первоначально компания планировала выпустить два релиза в течение следующих 12 месяцев. Однако благодаря успешному использованию скрама большая часть функциональных возможностей этих двух релизов была готова уже через пять месяцев. Посетив подразделение разработки, я узнал, что сотрудники работали по выходным и ночами, лишь бы добавить в релиз еще больше функциональности. Несмотря на то что инженеры трудились чрезвычайно продуктивно, маркетинг все еще ругал их за то, что они поставляли недостаточно

и не выполняли «обязательства». Разработчики чувствовали себя виноватыми за то, что не делали всего, что просил маркетинг, поэтому вредили своей личной жизни. Эта патология сохранялась, несмотря на выполнение 12-месячной работы двух релизов за пять месяцев. Старые привычки побороть очень сложно.

Еще одно существенное отличие скрама лучше всего описать, подумав о том, как строится дом. Покупатель не может переехать в дом, пока тот не будет полностью завершен. Предположим, что существует такой итеративно-инкрементальный подход к строительству, при котором дома строятся по комнатам. Сантехника, электричество и инфраструктура будут заложены в первой комнате, а затем проведены в каждую строящуюся комнату. В этом случае покупатель сможет переехать, как только решит, что завершено достаточное количество комнат. После переезда дополнительные комнаты могут быть построены в последовательности и в зависимости от актуальных потребностей покупателя. Скрам позволяет покупателям создавать программное обеспечение аналогичным образом. Как только развернута инфраструктура, компоненты системы постепенно доставляются покупателям, чтобы они могли пользоваться частями системы уже на ранних этапах цикла разработки. По мере реального использования системы покупатель может определить, какие следующие части и в каком порядке будут построены, и начинать использовать эти части по мере их готовности. Если покупатели удовлетворены уже реализованным подмножеством функциональности, они могут даже отказаться от создания всей системы в предполагаемом изначально виде.

Раньше я учил людей теории, практикам и правилам скрама. Теперь я стараюсь передать им ощущения, которые испытываешь, когда скрам применяется корректно. Я учу их, как распознавать, когда дела идут хорошо, а когда — плохо. Я даю упражнения и темы для обсуждений, которые позволяют достичь озарений, чтобы люди представляли, как ощущается скрам. Вы не знаете, как думает и что чувствует другой человек, пока не окажетесь на его месте, попав в похожую ситуацию. Точно так же вы не сможете полностью понять скрам, пока не станете применять его самостоятельно. Тем не менее, прочитав эту книгу, вы получите живое представление и начнете понимать, что чувствуют люди, применяющие скрам в своей организации.

Как следует читать эту книгу, являющуюся, по сути, сборником примеров из реального опыта применения скрама? Каждую историю я сопроводил описанием контекста, рассказал, как скрам использовался в этой ситуации, и предоставил некоторые извлеченные уроки. Все примеры сгруппированы в главы по темам:

1. Введение. Наука скрама.
2. Новые управленческие роли.
3. Скрам-мастер.
4. Команда разработки. Создаем порядок из хаоса.
5. Владелец продукта.
6. Планирование проекта в скраме.
7. Отчетность по проекту: поддержание прозрачности.
8. Команда разработки.
9. Масштабирование проектов с помощью скрама.

Если в главе рассматривается уже знакомая по предыдущим главам ситуация, я отсылаю к соответствующему описанию контекста.

В Приложении А перечислены события и правила скрама. Они объединяют скрам в единое целое.

Если вы знакомы со скрамом, но столкнулись с термином, который не совсем понимаете, обратитесь к Приложению Б «Определения».

Если вы незнакомы со скрамом, вам следует прочитать первую главу «Введение. Наука скрама», в которой кратко описаны теория скрама, процесс, практики, артефакты, роли и события.

Приложение В «Ресурсы» содержит список книг и сайтов, к которым вы можете обратиться, чтобы лучше понять скрам.

Приложения Г «Контракты с фиксированной ценой и фиксированной датой» и Д «Модель зрелости возможностей создания ПО (CMMI)» — белые вороны этой книги. Они содержат материалы, которые помогут вам использовать скрам в довольно уникальных обстоятельствах, не рассмотренных в примерах, составляющих основную часть этой книги.

Глава 1

Введение. Наука скрама

Продуктовая разработка — комплексная деятельность. Возможно, для вас это не новость, ведь в мире существует множество разнообразных комплексных процессов и явлений. Многие из них происходят сами собой, например превращение угля в алмазы под давлением. Некоторые комплексные процессы допускают неточность, например ежедневная дорога от дома до работы.

О комплексности большинства явлений мы просто не догадываемся или не задумывались, однако невозможно не заметить комплексность разработки ПО. Во-первых, результат этого процесса эфемерен и состоит лишь из сигналов микросхем, которые управляют аппаратурой. Во-вторых, для создания этого результата используется крайне непостоянное сырье: требования пользователей к программной системе, которую они не видели, взаимодействие еще не созданной программы с другими программами посредством входящих и исходящих сигналов и взаимодействие с самыми комплексными организмами нашей планеты — людьми. В-третьих, сам процесс разработки полностью интеллектуален, а все промежуточные результаты — материальная форма представления мыслей, возникавших в ходе этого процесса.

Эта книга — именно об этом невероятно трудном процессе создания программного обеспечения. В этой главе я кратко расскажу о скраме — фреймворке, который повышает вероятность успеха разработки ПО. Он специально создан для того, чтобы

добывать полезные программные продукты из комплексных проблем. За последние 10 лет скрам успешно использовался в тысячах проектов сотен организаций. Он основан на теории управления производственными процессами, в которой используются такие концепции, как самоорганизация и эмергентность*.

Эта книга — о скрам-мастерах, руководителях и менеджерах скрам-проектов. Скрам-мастера сопровождают команду, демонстрируют лидерство, применяют коучинг, обучают команду корректному применению скрама, чтобы справляться с постоянно возникающими в проекте трудностями. Учитывая рассмотренные ранее особенности процесса разработки ПО, комплексность всегда в избытке, и преодолеть ее невозможно без усердной работы, гибкого ума и храбрости.

В этой главе рассказывается, как эмпиризм и скрам применяются в управлении комплексными процессами, включая управление проектами разработки программных систем. Фраза «Скрам помогает управлять проектом разработки ПО» не означает, что проект будет строго следовать плану, а результат будет идентичен ожидаемому. Скорее, скрам управляет процессом разработки программного обеспечения так, чтобы работа приводила к наиболее ценному результату.

Эмпирический процесс управления

Комплексными называются проблемы, поведение которых непредсказуемо. Более того, не только сами эти проблемы непредсказуемы, но и способы доказательства их непредсказуемости невозможно предсказать математически. Другими словами, статистика работы этих процессов никогда не приведет к пониманию, какой математической моделью можно их описать. Определить подходящую статистическую выборку можно только путем такого сильного обобщения элементов этих процессов, при котором результаты станут неактуальными для людей, пытающихся понять эти процессы или управлять ими.

---

* Наличие у системы свойств, не присущих ее элементам и возникающих в процессе функционирования системы. Например, способность автомобиля передвигаться или способность команды поставлять программный продукт.

Бóльшая часть нашего общества функционирует с помощью процессов, допускающих некоторую неточность. Колебание колес, тряска цилиндров, дрожание тормозов — все это происходит незаметно для нас и не препятствует управлению автомобилем и езде. При сборке машины детали приспосабливаются с достаточной для ее применения точностью. Мы можем управлять многими процессами потому, что точность получаемых результатов соответствует ограничениям наших органов восприятия. Например, собирая шкаф, мне нужно распилить и соединить материалы настолько точно, чтобы результат был приемлем для человеческого глаза. Если моей целью будет лишь функциональность шкафа, то я смогу быть гораздо менее точным.

Но что, если точности, получаемой путем усреднения, недостаточно и мы хотим создать что-то более точное? Что произойдет, если погрешность любого разрабатываемого нами процесса сборки автомобилей окажется слишком большой для наших клиентов и нам нужно будет уменьшить ее? В этих случаях мы должны внимательно пройти по процессу шаг за шагом, на каждом этапе получая подтверждение, что он приводит к продукту достаточной степени точности. Иначе мы должны адаптировать этапы процесса, чтобы вернуть процесс в диапазон допустимой погрешности.

Процесс, который позволяет раз за разом получать результат приемлемого качества, называется *предопределенным (детерминированным) процессом управления*. В случаях, когда из-за комплексной природы промежуточных действий невозможно достичь прогнозируемой определенности процесса, необходимо использовать *эмпирический процесс управления*.

> Предопределенный (теоретический) подход к моделированию уместно применять, когда базовые механизмы, с помощью которых функционирует процесс, достаточно хорошо понятны. Когда процесс слишком сложен для предопределенного подхода, эмпирический подход станет более подходящим выбором.
>
> *Бабатунде Огуннайке и Хармон Рэй. Динамика, моделирование и управление процессами (Process Dynamics, Modeling, and Control)**

* Babatunde A., Ogunnaike E. I. Process Dynamics, Modeling, and Control. Oxford University Press, 1994. P. 364 — *Прим. авт.*

Мы используем предопределенный процесс управления всегда, когда это возможно, потому что с его помощью можем наладить автоматическое производство и выпустить такое количество товара, которому можно будет установить адекватную рыночную цену. В случаях, когда качество получаемого товара неприемлемо для его использования, или цена не соответствует полученному уровню качества, или требуется слишком много усилий по повышению качества для соответствия цене, стоит обратить внимание на управление эмпирическим процессом и согласиться на его более высокую стоимость. В конечном счете создание успешных продуктов с первого раза, используя управление эмпирическим процессом, окажется намного дешевле переделки неудачных продуктов, созданных с применением управления предопределенным процессом.

Любое внедрение управления эмпирическим процессом основывается на трех китах: *прозрачность, инспекция* и *адаптация*. Прозрачность означает, что характеристики процесса, влияющие на результат, должны быть видны и известны тем, кто этот процесс контролирует. Одинаково важно и то, чтобы эти характеристики были видны, и то, чтобы они были правдивы. При управлении эмпирическим процессом нет места обману. Например, если кто-то говорит, что конкретная функциональность отмечена как «готовая», что имеется в виду? В сфере разработки ПО утверждение о готовности функциональности может подразумевать выполнение всех следующих условий: аккуратно написан программный код, произведен рефакторинг, протестированы отдельные компоненты, собран дистрибутив для установки, произведена приемка пользователями. При этом кто-то другой может подразумевать, что программный код был только написан, протестирован и собран в дистрибутив. Если нет единого понимания того, что означает слово «готово», то и от наличия информации о «готовности» конкретной функциональности пользы мало.

Второй кит управления эмпирическим процессом — инспекция. Необходимо достаточно часто инспектировать различные характеристики процесса, чтобы иметь возможность вовремя обнаружить неприемлемые отклонения. При выборе частоты проверок следует учитывать и то, что в результате акта проверки меняются и сами процессы. Интересно, что требуемая частота инспекций

часто превышает толерантность процесса к проверкам, поэтому для надлежащего управления процессом нужно производить проверки с такой частотой, при которой они не вредят процессу. К счастью, обычно это не относится к разработке программного обеспечения.

Третий кит управления эмпирическим процессом — адаптация. Если в ходе проверки инспектор выявляет, что одна или несколько характеристик процесса выходят за допустимые пределы значений и полученный продукт будет неприемлемым, то инспектор корректирует процесс или используемое сырье. Корректировка должна быть произведена как можно быстрее, чтобы свести к минимуму дальнейшее отклонение.

В качестве примера управления эмпирическим процессом давайте рассмотрим проверку программного кода. Код проверяется на соответствие лучшим отраслевым практикам и стандартам кодирования (инспекция). Все, кто участвует в проверке, одинаково понимают эти стандарты и передовые практики (прозрачность). Проверка кода производится всякий раз, когда кто-то понимает, что код части функциональности (или просто часть кода) завершен (частота инспекции). Наиболее опытные разработчики просматривают код (инспекция), и автор кода корректирует его в соответствии с их комментариями и предложениями (адаптация).

Разработка комплексного программного обеспечения

Разрабатывая программное обеспечение, я создаю набор логически взаимосвязанных инструкций, которые посылают сигналы, управляющие аппаратурой и ее взаимодействием с другими аппаратами, людьми или окружающей средой. Уровень точности, необходимый для успешного функционирования ПО, варьируется от невероятного до по-настоящему пугающего. Любой из этих объектов может оказаться комплексным. А когда эти комплексные объекты вступают во взаимодействие, уровень сложности зашкаливает. Рассматривая далее комплексную природу разработки программного обеспечения, давайте ограничимся тремя наиболее важными измерениями: требованиями, технологиями и людьми.

Требования к программному обеспечению бывают и простыми. Допустим, существует всего лишь один клиент, который будет единственным пользователем системы. Он может провести достаточное количество времени с разработчиком, чтобы четко согласовать то, что нужно создать. Предположим, что этот клиент, к несчастью, умирает сразу после подтверждения своих требований. В этом случае его требования остаются неизменными в ходе разработки: нет корректировок, пересмотров, уточнений, модификаций в последнюю минуту. Однако обычно *заинтересованных лиц* (тех, кто заинтересован в программном обеспечении и деталях его реализации) много. У них разные потребности, эти потребности трудно однозначно сформулировать, и они часто меняются. В большинстве случаев клиенты начинают понимать, чего хотят, только когда другие люди демонстрируют им, как поняли их желания. Их требования такие комплексные не только из-за своей неоднозначности, но и из-за постоянной изменчивости.

Технологии тоже бывают простыми, но такие редко используются в разработке программного обеспечения. Проектам по разработке ПО можно дать такое определение — это применение передовой, часто ненадежной технологии для решения бизнес-задач и достижения конкурентных преимуществ. К тому же обычно одновременно используется более одной технологии, каждая из них — комплексная, а друг с другом они взаимодействуют посредством еще более комплексных интерфейсов.

На рис. 1.1 вертикальная ось показывает комплексность требований, а горизонтальная — комплексность технологии. Пересечение этих двух видов комплексности определяет общий уровень комплексности проекта. Почти все проекты разработки программного обеспечения сегодня — комплексные. А некоторые даже хаотичны, но в них невозможно работать, пока часть сложностей не будет устранена.

Третье измерение комплексности — люди, разрабатывающие программное обеспечение. У всех разные интеллектуальные способности, навыки, опыт, точки зрения, взгляды, убеждения и предрассудки. В зависимости от качества и количества сна, состояния здоровья, погоды, соседей и отношений в семье каждый человек каждое утро просыпается в настроении, непохожем на вчерашнее. Затем эти разные и переменчивые люди начинают работать вместе,

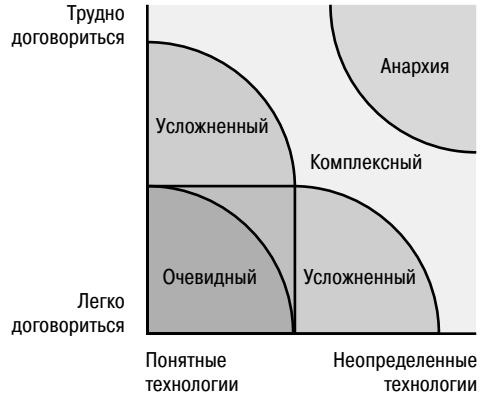

Рис. 1.1. Схема оценки комплексности проекта

и уровень комплексности зашкаливает. Принимая во внимание третье измерение комплексности — людей — в дополнение к технологиям и требованиям, я считаю, что последний «простой» проект случился в 1969 году, когда один человек из отдела обработки заказов в Sears Roebuck попросил меня отсортировать несколько карточек и создать отчет на IBM 360/20. С тех пор все становится более беспорядочным. Скрам помогает решать проблему комплексности проектов разработки программного обеспечения с помощью эмпирического процесса, включая обязательные требования его прозрачности, инспекции и адаптации, а также с помощью набора простых практик и правил, которые описаны в следующих разделах.

Скелет и сердце скрама

Все практики скрама держатся на скелете итеративно-инкрементального процесса. Он изображен на рис. 1.2. Нижний круг представляет собой итерацию разработки программного обеспечения. Итерации следуют одна за другой. Содержание работы в каждой итерации основывается на списке требований. В результате каждой итерации получается инкремент продукта. Верхний круг представляет собой ежедневную инспекцию в ходе итерации: участники команды разработки собираются вместе для инспекции работы, выполненной каждым за день, и для принятия мер по адаптации дальнейшей работы. Цикл итераций завершается, когда проект перестает финансироваться.

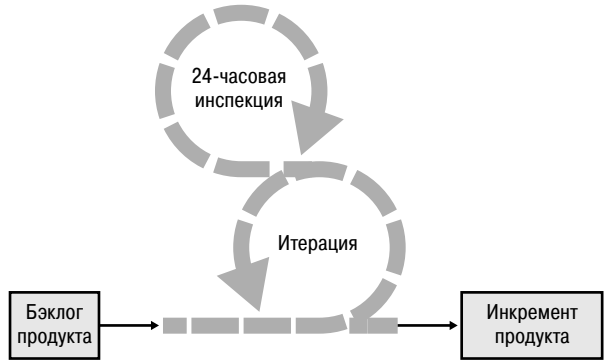

Рис. 1.2. Скелет скрама

Рассмотрим устройство скелета. В начале итерации команда разработки анализирует, что она должна сделать. Затем выбирает требования, которые сможет к концу итерации превратить в инкремент готовой к поставке* функциональности. В ходе итерации команда прилагает для этого все усилия, а любые заинтересованные лица не отвлекают команду до конца итерации. По завершении итерации команда разработки демонстрирует созданный за итерацию инкремент продукта, чтобы заинтересованные лица могли произвести его инспекцию и соответствующие адаптации в проекте могли быть произведены своевременно.

Сердцем скрама является итерация. Команда изучает требования, рассматривает доступные технологии и оценивает свои собственные возможности и навыки. Затем сообща определяет, как она будет создавать запланированную функциональность. Сталкиваясь с новыми трудностями и неожиданностями комплексного окружения, команда разработки ежедневно изменяет свой подход к работе. Команда выясняет, что нужно реализовать за итерацию, и самостоятельно выбирает лучший способ сделать это. Этот творческий процесс — основа продуктивности в скраме.

Реализовать этот скелет итеративно-инкрементального процесса в скраме позволяют три роли. В следующих разделах я дам краткий обзор этих ролей, а затем опишу процесс скрама и его

* «Готовая к поставке функциональность», также встречается «потенциально поставляемая функциональность» — по решению владельца продукта может быть установлена в промышленную среду или иным образом предоставлена пользователям.

артефакты. Приложение А содержит список событий и правил, а Приложение Б — определения терминов скрама. Более подробную информацию о скраме можно найти в Приложении В «Ресурсы», а также в книге «Гибкое управление разработкой ПО по скраму» (Agile Software Development with Scrum).

Роли скрама

В скраме есть только три роли: владелец продукта, команда разработки и скрам-мастер. Вся ответственность в части управления проектом разделена между этими ролями. Владелец продукта отвечает за представление интересов всех лиц, заинтересованных в проекте, в создаваемой программной системе. Чтобы добиться и затем получать регулярное финансирование проекта, владелец продукта описывает первоначальные общие требования проекта, определяет цели по возврату инвестиций (ROI) и поддерживает план релизов в актуальном состоянии. Список требований называется *бэклогом продукта*. Владелец продукта несет ответственность за поставку заинтересованным лицам максимальной ценности: гарантирует, что наиболее ценная функциональность будет реализована в первую очередь. Это достигается за счет регулярного пересмотра порядка задач в бэклоге так, чтобы в следующую итерацию были взяты самые ценные требования.

Команда разработки отвечает за создание функциональности. Команда является самоуправляющейся, самоорганизующейся и кросс-функциональной*. Она несет ответственность за организацию своей работы и за решения о том, как в рамках итерации превратить часть бэклога продукта в инкремент потенциально поставляемой функциональности. Участники команды несут коллективную ответственность за успех каждой итерации и проекта в целом.

Скрам-мастер отвечает за организацию процесса скрама, за обучение скраму всех участников проекта, за то, чтобы каждый следовал правилам и практикам скрама, за внедрение скрама

* «Кросс-функциональные команды обладают всеми необходимыми компетенциями для выполнения работы и не зависят от людей, которые не входят в команду», — «Руководство по скраму», 2017. Все участники команды вместе обладают набором знаний и навыков, необходимых для реализации продукта от идеи до доставки пользователю.

так, чтобы он и соответствовал культуре компании, и позволял получать ожидаемые преимущества.

Люди, выполняющие эти роли, должны быть преданы проекту. Остальные тоже могут быть заинтересованы в проекте, но они «не на крючке». Скрам четко различает эти две группы и гарантирует, что те, кто несет ответственность за проект, имеют полномочия делать все необходимое для его успеха, а те, кто не несет такой ответственности, не могут вмешиваться в работу первых. В этой книге я называю эти группы людей «цыплятами» и «поросятами» соответственно. Названия пришли из старого анекдота. Цыпленок и поросенок идут по дороге. Цыпленок говорит поросенку: «Хочешь открыть вместе со мной ресторан?» Поросенок поразмыслил и отвечает: «Да, пожалуй. Как ты хотел бы назвать его?» «Яичница с беконом!» — отвечает цыпленок. Поросенок останавливается и, вздохнув, отвечает: «Знаешь, я передумал. Я не хочу открывать такой ресторан с тобой, потому что мне придется отдать себя проекту целиком, а тебе — лишь поучаствовать!»

Это важное различие, поэтому скрам так настойчив в требовании полной прозрачности: всегда должно быть понятно, кто на крючке, а кто просто назойливый советчик. Кто несет ответственность за возврат инвестиций, а кто имеет долю в ROI, но не отвечает за него. Кому придется превратить новую технологию в функциональность, а кто — лишь беспокойный «адвокат дьявола». Скрам проводит явную черту между цыплятами и поросятами, чтобы положить конец неразберихе, и тем самым повышает производительность и создает продуктивный импульс.

Процесс скрама

Скрам-проект начинается с описания видения системы, которую предстоит разработать. Вначале это видение может быть нечетким, выраженным в маркетинговых, а не системных и технических терминах. Однако с развитием проекта оно становится яснее и конкретнее. Владелец продукта отвечает перед инвесторами проекта за разработку и поставку описанной в видении продукта функциональности, которая позволит добиться максимального возврата инвестиций. Владелец продукта создает соответствующий план, в том числе и бэклог продукта — список функциональных

и нефункциональных требований, необходимых для реализации описанной в видении системы. Элементы бэклога продукта располагаются в порядке убывания ценности: в верхней части списка расположены наиболее ценные или прибыльные элементы. Бэклог продукта разделен на части — предполагаемые релизы. Такой упорядоченный бэклог продукта — лишь отправная точка, поскольку его содержимое, порядок и группировка в релизы обычно изменяются уже в момент начала проекта. Изменения бэклога продукта отражают изменчивость бизнес-требований и то, насколько быстро или медленно команда разработки превращает элементы бэклога в работающую функциональность.

Вся работа выполняется в *спринтах*. Каждый спринт — итерация длиной максимум в один месяц — начинается с *планирования спринта* — события, на котором владелец продукта и команда разработки совместными усилиями определяют, что будет сделано за следующий спринт. Владелец продукта выбирает из бэклога продукта наиболее ценные элементы и рассказывает команде, какой результат хочет получить в конце спринта. В свою очередь команда сообщает владельцу продукта, какие элементы из запрошенной им функциональности она сможет разработать за спринт. Считается, что спринт стартовал, как только началось планирование спринта. Планирование спринта должно длиться не более восьми часов. Событие ограничено по времени, чтобы не затягивать обсуждение и не вступать в бесполезные споры о том, что можно сделать в течение спринта. Цель планирования — договориться и начать работу, а не обдумывать ее снова и снова.

Планирование спринта состоит из двух частей. В течение первых четырех часов владелец продукта рассказывает команде о наиболее важных задачах, расположенных в верхней части бэклога, а команда разработки расспрашивает его о содержании задач, их сути, назначении и целях. Прояснив необходимые детали, команда выбирает только те элементы бэклога продукта, которые сможет разработать за спринт, то есть превратить в готовый к выпуску инкремент продукта. Она обещает владельцу продукта, что сделает для этого все от нее зависящее. В течение вторых четырех часов планирования спринта команда разработки создает план спринта. Поскольку команда сама отвечает за организацию своей работы, ей необходим предварительный план, чтобы начать разработку

задач спринта. Все взятые в спринт элементы бэклога продукта и необходимые для их реализации подзадачи составляют *бэклог спринта*. В течение спринта могут добавляться дополнительные подзадачи.

Ежедневный скрам — это 15-минутное событие команды, которое проводится каждый день в ходе спринта. Его цель — ежедневно синхронизировать информацию о состоянии работы всех участников команды, а также запланировать встречи, которые помогут команде выполнить задачи из бэклога спринта. Для этого участники могут использовать формат трех вопросов, которые позволяют сфокусироваться на цели спринта и командной ответственности:

1. Что я сделал с момента окончания предыдущего ежедневного скрама, чтобы помочь команде достичь цели спринта?
2. Что я планирую сделать до следующего ежедневного скрама, чтобы помочь команде достичь цели спринта?
3. Какие препятствия могут помешать мне и команде достичь цели спринта?

В конце спринта проводится *обзор спринта* — событие, в ходе которого команда разработки демонстрирует любым заинтересованным лицам результаты выполненной за спринт работы. Обзор спринта должен длиться не более четырех часов. Это неформальное событие, цель которого — совместно обсудить разработанную командой функциональность и определить, над чем нужно работать в следующих спринтах.

После обзора спринта и перед планированием следующего спринта скрам-мастер проводит с командой *ретроспективу спринта*. Событие должно длиться не более трех часов. Скрам-мастер побуждает команду анализировать и улучшать процесс разработки в рамках фреймворка скрама. Это необходимо, чтобы в следующем спринте повысить эффективность работы и получать большее удовольствие от нее.

Все перечисленные регулярные события (планирование спринта, ежедневный скрам, обзор спринта, ретроспектива спринта) являются формальными возможностями для инспекции и адаптации, на которых основано управление эмпирическим процессом по скраму. Схема процесса скрама представлена на рис. 1.3.

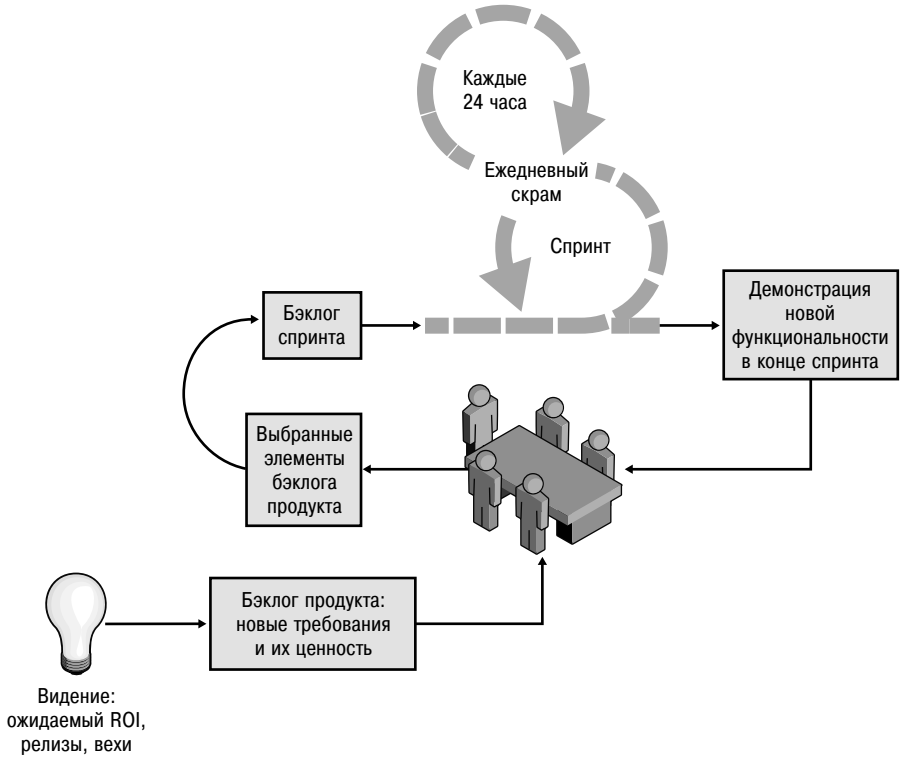

Рис. 1.3. Обзор процесса скрама

Артефакты скрама

Используемые в скраме артефакты описаны в следующих разделах.

Бэклог продукта

Требования к системе или продукту, разрабатываемому в рамках проекта или нескольких проектов, перечислены в бэклоге продукта. Владелец продукта несет ответственность за содержание, порядок расположения элементов и доступность бэклога продукта. Бэклог продукта никогда не полон и к моменту планирования проекта содержит лишь первичное представление о требованиях. Бэклог динамичен: по мере развития продукта, окружающей среды, понимания заинтересованными лицами того, каким дол-

жен быть полезный конкурентный продукт, изменяется и его бэклог. Пока существует продукт, существует и бэклог продукта. На рис. 1.4 показан пример бэклога «Система управления продуктом в скраме». Он оформлен в виде таблицы.

Бэклог	Первоначальная оценка	Коэффициент сложности	Скорректированная оценка	Осталось работы до завершения						
				1	2	3	4	5	6	7
Задачи				256	209	193	140	140	140	140
Выбрать проект или создать новый	3	0,2	3,6	3,6	0	0	0	0	0	0
Шаблон бэклога для новых проектов	2	0,2	2,4	2,4	0	0	0	0	0	0
Создать таблицу бэклога продукта с форматированием	3	0,2	3,6	3,6	0	0	0	0	0	0
Создать таблицу бэклога спринта с форматированием	3	0,2	3,6	3,6	0	0	0	0	0	0
Отобразить древовидную структуру элементов бэклога продукта, релизов, спринтов	2	0,2	2,4	2,4	0	0	0	0	0	0
Спринт 1	13	0,2	15,6	16	0	0	0	0	0	0
Создать новое окно с шаблоном бэклога продукта	3	0,2	3,6	3,6	3,6	0	0	0	0	0
Создать новое окно с шаблоном бэклога спринта	2	0,2	2,4	2,4	2,4	0	0	0	0	0
Окно с графиком сгорания элементов бэклога продукта	5	0,2	6	6	6	0	0	0	0	0
Окно с графиком сгорания элементов бэклога спринта	1	0,2	1,2	1,2	1,2	0	0	0	0	0
Отобразить древовидную структуру элементов бэклога продукта, релизов, спринтов	2	0,2	2,4	2,4	2,4	0	0	0	0	0
Отобразить график сгорания для выбранного спринта или релиза	3	0,2	3,6	3,6	3,6	0	0	0	0	0
Спринт 2	16	0,2	19,2	19	19	1,2	0	0	0	0
Автоматический пересчет значений и итогов	3	0,2	3,6	3,6	3,6	3,6	0	0	0	0
Обновить график сгорания на главной странице при внесении изменений в бэклог продукта в другом окне	2	0,2	2,4	2,4	2,4	2,4	0	0	0	0
Скрыть / автоматически повторно отобразить окно сгорания	3	0,2	3,6	3,6	3,6	3,6	0	0	0	0
Добавить функцию «Итого» для спринта… добавляет строку с суммой по спринту	2	0,2	2,4	2,4	2,4	2,4	0	0	0	0

Окончание рис. 1.4

Бэклог	Первоначальная оценка	Коэффициент сложности	Скорректированная оценка	Осталось работы до завершения						
				1	2	3	4	5	6	7
Задачи				**256**	**209**	**193**	**140**	**140**	**140**	**140**
Добавить функцию «Итого» для релиза ... добавляет итоговую строку по релизу	1	0,2	1,2	1,2	1,2	1,2	0	0	0	0
Функции «Владелец» и «Исполнитель», опциональные столбцы	2	0,2	2,4	2,4	2,4	2,4	0	0	0	0
Распечатать графики сгорания	1	0,2	1,2	1,2	1,2	1,2	0	0	0	0
Спринт 3	14	0,2	16,8	17	17	17	0	0	0	0
Дублировать незавершенные задачи бэклога, не изменяя суммарные значения «Итого»	5	0,2	6	6	6	6	6	6	6	6
Возможность делать заметки	6	0,2	7,2	7,2	7,2	7,2	7,2	7,2	7,2	7,2
Функция отображения «что — если» для релиза на графике сгорания	15	0,2	18	18	18	18	18	18	18	18
Возможность отображения тренда на графике сгорания	2	0,2	2,4	2,4	2,4	2,4	2,4	2,4	2,4	2,4
Возможность публикации всего проекта, публикация в виде веб-страниц HTML	11	0,2	13,2	0	0	13	13	13	13	13
Будущие спринты	39	0,2	46,8	34	34	47	47	47	47	47
Релиз 1				85	70	65	47	47	47	47

Рис. 1.4. Бэклог «Система управления продуктом в скраме»

Эта таблица — бэклог «Система управления продуктом в скраме» по состоянию на март 2003 года. Я был владельцем этого продукта. Строки таблицы — элементы бэклога. Они разделены подзаголовками «Спринт» и «Релиз». Например, все строки выше «Спринта 1» представляют собой задачи, реализованные в первом спринте, а строки между подзаголовками «Спринт 1» и «Спринт 2» были реализованы во втором спринте. Обратите внимание, что строка «Отобразить древовидную структуру бэклога продукта, релизов, спринтов» дублируется в спринтах 1 и 2. Это потому, что задача в строке 10 не была завершена в спринте 1 и была перенесена в спринт 2. Если бы после завершения первого спринта я решил, что ценность этого требования ниже ценности задач второго, третьего или других спринтов, то я мог бы переместить эту строку еще ниже в списке задач.

Рассмотрим столбцы таблицы с бэклогом продукта:

1) описание элемента бэклога продукта;
2) начальная оценка;
3) коэффициент сложности. Некоторые особенности проекта снижают производительность команды. Этот коэффициент позволяет учесть это в оценке;
4) скорректированная оценка, получаемая применением коэффициента сложности к начальной оценке;
5) остальные столбцы представляют спринты, в течение которых реализуются требования бэклога продукта. Добавляя новый элемент в бэклог, мы указывали начальную оценку в столбец текущего спринта. Пример применения этого правила вы можете увидеть только в строке «Возможность публикации всего проекта, публикация в виде веб-страниц HTML», об этой задаче мы не задумывались до третьего спринта, а остальные элементы этого бэклога я и разработчики создали еще до начала первого спринта проекта.

График сгорания показывает объем работы, оставшийся к определенному моменту времени. Он изображен на рис. 1.5. Это отличный способ увидеть корреляцию между объемом оставшейся работы и прогрессом команды (или нескольких команд) проекта в сокращении этого объема. Пересечение линии тренда оставшейся работы и горизонтальной оси показывает наиболее вероятное на текущий момент время завершения работы. Это позволяет понять, «что будет, если» мы, желая получить больше функциональности, добавим в релиз дополнительные элементы, или «что будет, если» мы, стремясь выпустить функциональность раньше, уберем часть элементов бэклога из релиза. График сгорания — это столкновение реальности с планом: того, что уже сделано и за какое время, с тем, что мы надеемся сделать.

Элементы бэклога продукта, которые возьмут в работу в будущих спринтах, описаны довольно общими словами. Я не тратил время на их анализ и более точную оценку, потому что команда разработки не приступит к реализации этого функционала в ближайшее время. Более того, существует множество других требований к этому продукту, но они еще не продуманы и поэтому

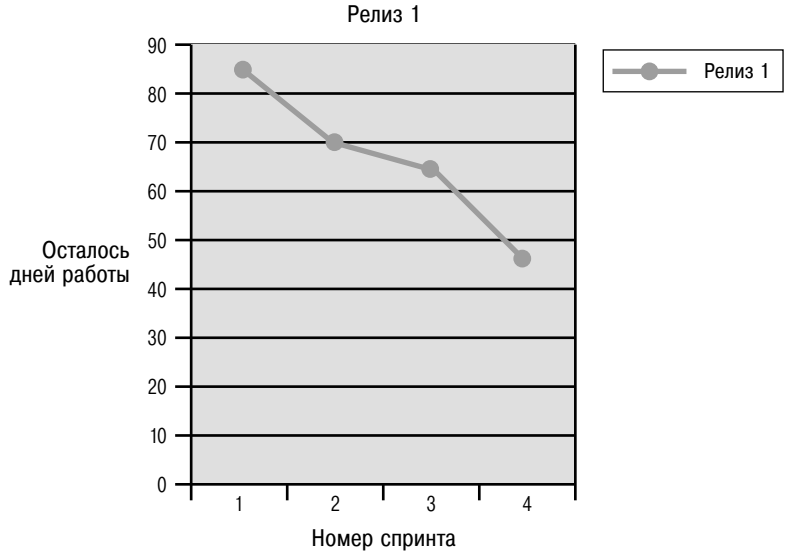

Рис. 1.5. График сгорания

не упомянуты в бэклоге. Когда у меня будет время или желание продолжить проработку бэклога, я добавлю в него больше элементов. Эта конкретная таблица — пример требований к новому продукту, поэтому я могу отложить детализацию и уточнение бэклога до момента, когда команда начнет превращать эти элементы в работающую функциональность.

Бэклог спринта

Из бэклога продукта команда разработки выбирает элементы, которые она за один спринт сможет превратить в потенциально поставляемый инкремент продукта. Во второй части планирования спринта команда для каждого элемента определяет задачи, необходимые для его реализации. Набор выбранных элементов и запланированных задач вместе составляют бэклог спринта. Каждая задача должна занимать от 4 до 16 часов. Задачи длительностью более 16 часов считаются просто контейнерами для меньших задач, которые еще не были надлежащим образом определены. Только команда разработки может изменить состав бэклога спринта.

Бэклог спринта — очень наглядная и в любой момент времени доступная картина работы, которую команда планирует выполнить за спринт. Пример бэклога спринта показан на рис. 1.6. Строки таблицы — задачи из бэклога спринта, столбцы — дни спринта максимальной длительности в один месяц. Как только задача определена, в ячейке на пересечении строки задачи и столбца дня спринта исполнитель указывает, сколько еще часов необходимо для завершения задачи.

Потенциально готовый к поставке инкремент продукта

Скрам требует, чтобы в результате каждого спринта появлялся инкремент продукта — чтобы команды создавали новую функциональность, а не только вносили исправления в существующую. Этот инкремент должен быть потенциально поставляемым в промышленную среду, поскольку владелец продукта может решить использовать новую функциональность сразу же. Это означает, что созданная функциональность должна:

- состоять из хорошо написанного, логично и понятно структурированного программного кода, встроенного в исполняемый файл;
- быть тщательно протестирована;
- быть описана в пользовательской документации или в файлах справки.

Перечисленные требования — определение «готового» инкремента продукта.

Если создаваемый в ходе спринта инкремент продукта будет использоваться в более регламентированной среде, компания-разработчик обычно определяет дополнительные требования к продукту в виде стандартов или соглашений. Например, Управление по санитарному надзору за качеством пищевых продуктов и медикаментов министерства здравоохранения и социальных служб США утверждает все продукты, которые будут использоваться в жизненно важных условиях в медицинских учреждениях. В рамках процесса утверждения Управление проверяет, что требования к продукту являются адекватными, полными и непосредственно

Бэклог Описание задачи	Автор	Ответственный	Статус (Не начато, В процессе, Завершено)	Осталось часов работы											
				1	2	3	4	5	6	7	8	9	10	11	12
Встретиться для обсуждения целей и функционала для спринтов 3–6	Даниэлла	Даниэлла/ Сью	Завершено	20	0	0	0	0	0	0	0	0	0	0	0
Убрать расчеты из Crystal Reports	Джим	Алан	Не начато	8	8	8	8	8	8	8	8	8	8	8	8
Получить данные KEG		Том	Завершено	12	0	0	0	0	0	0	0	0	0	0	0
Проанализировать данные KEG — Заголовок		Джордж	В процессе	24	24	24	24	12	10	10	10	10	10	10	10
Проанализировать данные KEG — Посылка		Тим	Завершено	12	12	12	12	12	4	4	4	0	0	0	0
Проанализировать данные KEG — Закладная		Джош	В процессе							12	10	10	10	10	10
Проанализировать данные KEG — Контактное лицо		Даниэлла	В процессе	24	24	24	24	12	10	8	6	6	6	6	6
Проанализировать данные KEG — Услуги		Алан	В процессе	24	24	24	24	12	10	10	10	10	10	10	10
Создать базу данных		Барри/ Дейв	В процессе	80	80	80	80	80	80	60	60	60	60	60	60
Проверить размер базы данных KEG		Тим	Не начато												
Посмотреть на данные KEG на диске G:\		Дейв	В процессе	3	3	3	3	3	3	3	3	3	3	3	3
Подтвердить соглашение с KEG		Сью	Не начато												
Подтвердить доступность сотрудников KEG		Том	Не начато	1	1	1	1	1	1	1	1	1	1	1	1
Переключить JDK на версию 1.3.1. Прогнать все тесты		Алан	Не начато	8	8	8	8	8	8	8	8	8	8	8	8
Хранить файлы PDF в структуре		Джеки	Завершено	8	0	0	0	0	0	0	0	0	0	0	0
TopLink. Не удается избавиться от преобразователя NetScape		Ричард	Завершено	4	0	0	0	0	0	0	0	0	0	0	0
Создать хранилище тестовых данных		Барри	В процессе	10	10	10	10	10	10	10	10	8	8	8	8
Перенести приложение и базу данных в Qual (включая Crystal)		Ричард	Завершено	4	4	4	4	4	4	4	0	0	0	0	0
Настроить среду Crystal		Джош	Завершено	2	2	2	2	1	1	1	0	0	0	0	0

Окончание рис. 1.6

Бэклог Описание задачи	Автор	Ответственный	Статус (Не начато, В процессе, Завершено)	Осталось часов работы											
				1	2	3	4	5	6	7	8	9	10	11	12
Протестировать приложение в Qual		Сью	В процессе												20
Определить цель спринта, необходимую для решения в 2002 году		Линн	В процессе	40	40	40	40	40	40	40	38	38	38	38	38
Таблицы источников для импорта		Джош	В процессе												
Разработать типовой процесс обработки исключений при импорте		Джош	В процессе									12	12	12	10
Обрабатывать несколько файлов для импорта на одной странице		Джеки	Отклонено												
Мигрировать сервлет CruiseControl на сервер iWS 6.0 (landcc_7101)		Алан	Не начато	4	4	4	4	4	4	4	4	4	4	4	4
Создать веб-сервер для Qual на диске PF1D8 LTCS		Алан	Завершено	1	0	0	0	0	0	0	0	0	0	0	0
		Даниэлла/ Джордж	В процессе	12	12	12	12	8	8	8	8	8	8	8	8
Обсудить с Томом и Сью ответы на вопросы по данным KEG в письме с темой «Keg, LTO»	Джеки	Даниэлла	Завершено	10	10	10	10	10	8	8	0	0	0	0	0
Сопоставить данные KEG и Active Tables. Также см. № 14	Джеки	Джеки/ Алан	В процессе	50	50	50	50	50	50	50	50	50	50	50	50
Подготовить SQL для импорта таблиц KEG в Active Tables	Джеки	Джордж	В процессе	25	25	25	25	25	25	25	25	25	25	25	25

Рис. 1.6. Бэклог спринта

относящимися к продукту. Чтобы инкремент жизненно важных продуктов мог быть поставлен потребителям, он должен быть потенциально готов к утверждению Управлением, а для этого он должен удовлетворять требованиям стандартов.

Резюме

Итак, мы рассмотрели: теорию, скелет, сердце, роли, процесс и артефакты скрама. Более подробную информацию о том, как, почему и что происходит в скраме, определения терминов и ссылки на дополнительные ресурсы вы найдете в Приложениях. Но учтите, что так же, как мои поездки на велосипеде по Восточному Массачусетсу недостаточны для квалификации Тур де Франс, полученные в первой главе знания недостаточны для управления проектом по скраму. Для этого вам нужна практика и некоторое понимание того, как скрам применяется в реальной жизни. Следующие главы именно об этом — о практике использования скрама.