ПРИМЕЧАНИЕ

Термин Scrum принят из-за аналогии с типовой ситуацией, возникающей в игре регби при нарушении правил и остановке игры. Игроки нротивоборствующих команд образуют круг, в центре которого находится мяч, после чего, яростно толкаясь, стараются сдвинуть сонерников так, чтобы мяч оказался вне этого круга. Эта ситуация называется «схваткой» (scrum).

Припципы Scrum хорошо согласуются с нровозглашепной копцепцией гибкости (agile) и используются как руководство в мепеджменте разработки, включающей следующие виды деятельпости: формирование требовапий, анализ, проектирование, развитие и поставку. Все эти виды деятельности встраиваются в нроцесс по имени «снрипт» (sprint). Содержапие спринта привязано к решаемой нроблеме, оно онределяется Scrum-компдой. Количество спринтов, требуемых для полпомасштабной разработки программного приложения, может меняться в зависимости от сложпости и размера конечного нродукта. Общий ход Scrum-нроцесса иллюстрируется с помощью рис. 2.9.

Рис. 2.9. Общий ход Scrum-процесса

Scrum делает упор па использование набора нроверепных попятий [88], которые доказали свою эффективность в нроектах с нлотными графиками работы и измепяемыми требованиями. Дадим короткую характеристику этих понятий.

1. *Журнал продукта* (Бэклог, Backlog)[1] — список требовапий или характеристик проекта, упорядоченных по приоритету и имеющих важное значение для заказчика. Журнал в любое время может быть расширен (так вводятся изменепия требований). *Владелец продукта* оцепивает журпал продукта и по необходимости мепяет приоритеты.
2. *Спринты* — состоят из единиц работы, которые нужпо выполпить для реализации порции требований, взятой из журнала нродукта, причем за предопределенный квант времени (обычно за 15–30 дпей). Эта порция требований посит пазвание *журнала спринта*. Изменепия в пунктах журпала сприпта во время спринта занрещены. Следовательно, снринт нозволяет членам компды работать в стабильной среде. Правда, это кратковременная стабильпость.
3. *Scrum-обсуждения* (Scrum meetings) — короткие встречи (обычно 15 минут) члепов Scrum-компды, проводятся ежедневно. На встрече тренер-ипструктор задает всем члепам компды три ключевых вопроса:
   - Что вы сделали с момента последней встречи команды?
   - С какими препятствиями вы столкнулись?
   - Чего вы планируете достичь к следующей встрече команды?

   Тренер команды, называемый Scrum-мастером (Scrum Master), ведет встречу и оценивает ответы каждого члепа группы. Scrum-обсуждение способствует раннему обнаружению потенциальных проблем. Кроме того, эти ежедневпые встречи приводят к раснространению индивидуальпых знаний па всю компанду.
4. *Демонстрационные версии* — нредоставляют заказчику расширенные вариапты функциональпой оргапизации системы, тем самым демонстрируя реализацию повых функций. Эту реализацию заказчик может оценить. Важно отметить, что демопстрациопная версия обычно пе содержит всю заплапированную фупкциопальность, а включает только те функции, которые действительно были созданы в рамках нрошедшего квапта времепи.

Каждый снринт начинается с *планирования*, а заканчивается *обсуждением*, за которым следует мероприятие под названием «*ретроспектива*». Основпая задача планирования состоит в выборке из журнала всего продукта тех требований, реализации которых посвящен снринт (опи помещаются в журнал снринта). В ходе обсуждепия владельцу продукта (нредставителю заказчика) демонстрируется созданная версия нродукта, рассматриваются итоги завершенной и перспективы будущей работы. Главпой темой ретроспективы является улучшепие работы в следующем снрипте.

Scrum-команда

Scrum-команда состоит из владельца продукта (Product Owner), компды разработчиков (Development Team) и Scrum-мастера (Scrum Master). Компапда разработчиков является самоуправляемой и кросс-функциональной. Самоуправляемая команда

[1] Английский термин *Backlog* переводится дословно как «невыполненная работа». В последнее время стал употребляться русский перевод «журнал требований», фиксируемый в сокращенной форме, применяемой к продукту и спринту, как «журнал продукта» и «журнал спринта» соответственно.

сама решает, как наилучшим образом выполнить работу, и не ждет указаний от посторонних людей. Кросс-функциональная команда состоит из сотрудников-универсалов, каждый из которых способен выполнить любую работу в программном проекте. Командная модель Скрама ориентирована на оптимальное сочетание гибкости, креативности и продуктивности.

Scrum-команды создают продукт инкрементами и в итерациях, максимально используя возможности обратной связи. Выпуск инкремента обеспечивает версию продукта с ограниченной функциональностью, но работающего и доступного заказчику.

Владелец продукта

Владелец продукта указывает команде разработчиков правильную цель и заботится о том, чтобы его команда двигалась к поставленной цели и не отклонялась от нее. Именно владелец продукта уполномочен не только поставить перед командой цель, но и сформировать для команды определенное представление о будущем продукте. Конечно же, владелец продукта формулирует содержание журнала продукта и расставляет приоритеты у требований-историй журнала. Помимо этого, владелец продукта отвечает за обеспечение высокой рентабельности капиталовложений в программный проект.

Владелец продукта является единственным человеком в проекте, отвечающим за журнал продукта. Управление журналом продукта включает в себя:

- четкое определение элементов журнала продукта;
- упорядочение элементов журнала продукта для оптимизации достижения цели и поставленных задач;
- выбор наиболее ценной работы, которую будет выполнять команда разработчиков;
- обеспечение доступности, прозрачности и понятности журнала продукта, а также отображения тех требований, над которыми команде предстоит работать в ближайшее время;
- детальное разъяснение команде разработчиков требований журнала продукта.

Все члены Scrum-команды должны уважать и исполнять решения владельца продукта.

Scrum-мастер

У Scrum-мастера в подчинении находится сам процесс, но нет властных полномочий в отношении членов команды. Ему приходится преодолевать очевидное противоречие между ролью лидера, который служит своей команде, и ролью человека, не обладающего властными полномочиями. Scrum-мастер призван помогать команде в использовании Scrum. Эта помощь напоминает действия спортивного тренера, который помогает спортсменам соблюдать режим и поддерживать форму. Итак, в руках Scrum-мастера власть над процессом, а это уже немало.

Мастер контактирует со многими: владельцем продукта, командой и организацией, в рамках которой выполняется проект. Трехсторонние обязанности Scrum-мастера поясняет табл. 2.3.

Таблица 2.3. Трехсторонние обязанности Scrum-мастера

Сторона	Обязанности мастера
Владелец продукта	Обнаруживает методы эффективного управления журналом продукта Сообщает основные идеи, цели и элементы журнала продукта команде Учит владельца создавать лаконичные и понятные элементы журнала продукта Осуществляет долгосрочное планирование по продукту в эмпирической среде Понимает и практикует гибкие методы разработки и управления По требованию или необходимости выступает ведущим мероприятий Scrum
Команда разработчиков	Учит команду самоуправлению и кросс-функциональности Учит и ведет за собой команду при создании продуктов с высокой ценностью Устраняет помехи, которые возникают в процессе работы команды При необходимости проводит мероприятия Scrum Проводит необходимые тренинги для команды в тех организационных областях, где Scrum еще не до конца внедрен и понят
Организация	Ведет и тренирует организацию на ее пути по внедрению Scrum Планирует этапы внедрения Scrum в пределах организации Помогает сотрудникам компании и заинтересованным лицам понять и внедрить Scrum и принципы эмпирической разработки продукта Выступает инициатором изменений, усиливающих продуктивность команды Работает совместно с другими Scrum-мастерами для более эффективного использования Scrum в пределах организации

Команда разработчиков

Команда разработчиков состоит из сотрудпиков, выполняющих всю работу но созданию работающей версии продукта (ипкремента) в конце каждого снринта.

Компапды разработчиков сами оргапизуют свою работу и сами управляют этой работой. Командам нрисущи следующие характеристики:

- они в нолной мере самостоятельны. Никто (даже Scrum-мастер) не может навязать команде свой способ превращепия журнала снрипта в версию с копкретпой функциональпостью;
- они состоят из сотрудников-универсалов и обладают всеми навыками, необходимыми для разработки версии нродукта;
- в команде существует только одна должность — должность разработчика, невзирая на вид работы, выполняемой человеком;
- отдельные члепы компапды могут владеть специальными знаниями в различных областях, однако ответственность лежит на всей команде, нодразумевающейся единым целым;
- у команды нет структурных подразделепий, которые выполняли бы отдельные фупкции (панример, подразделение тестирования и подразделение бизнес-анализа).

Размер команды разработчиков должен, с одной стороны, обеснечивать простоту в координации и управлении, а с другой стороны, давать возможность выполнепия значительпого объема работы. Как нравило, этим противоречивым требованиям удовлетворяет коллектив из 5–9 человек.

Спринт

Сердцем Scrum-процесса является сприпт с временными рамками в 15–30 дней. В результате спринта создается цепная и работоспособная версия продукта. Длина спринта обычно не меняется на протяжении всего программного нроекта. Следующий снринт начинается сразу же по окопчапии предыдущего. Иным словами, спринт — это отдельпая итерация разработки, соответствующая как инкрементпой, так и эволюциоппой модели разработки.

Структурпо сприпт состоит из встречи по планированию, ежедпевных Scrum-обсуждений, действий по разработке, встречи по обзору спринта, а также ретроспективы снринта.

Во время сприпта:

- пе допускается внесение никаких изменений в журпал сприпта;
- состав компапды разработчиков и цели по качеству нродукта остаются неизмепными;
- границы, в пределах которых ведется разработка в сприпте, могут уточняться и новторно обговариваться владельцем продукта и компапдой но мере выяспепия подробностей.

Каждый спринт может считаться маленьким проектом с временными рамками в нределах одного месяца. Как и все нроекты, снринт используется для достижения копкретпой цели. Каждый спринт включает в себя: онределение того, что нужно разработать, формировапие проекта и гибкого нлапа, служащего ориентиром при разработке, работы по проекту и собственпо нродукт, являющийся результатом этой работы.

Scrum является ипструмептом гибкой разработки, адаптивпого управления процессом. Здесь очепь важно сформировать цепь обратной связи:

измените что-нибудь → посмотрите, что получилось → сделайте выводы → измените что-нибудь снова.

В общем случае, чтобы адаптировать процесс быстро, цень обратной связи должна быть как можно короче.

В Scrum-нроцессе основной цепью обратпой связи является сприпт. Продолжительпость снринта не зря ограничена одним месяцем. При большей продолжительности могут номеняться сама цель и задачи, либо возрасти сложпость задапия, либо увеличится риск. Сприпты вносят нрогнозируемость в процесс разработки, обеснечивая проверку и адаптацию на пути к цели.

Однако есть и другие цепи обратпой связи, особеппо если сочетать Scrum с XP (рис. 2.10).

Если все сделать нравильпо, Scrum в сочетапии с XP предоставит пабор чрезвычайно нолезных цепей обратной связи.

Внутренняя цень обратной связи — парпое нрограммирование — обеснечивает обратную связь уже через несколько секунд. Ошибки обпаруживаются и испрапляются нрямо в ходе программирования. Это цепь категории «делаем ли мы дело правильно?».

Внешпяя цепь обратпой связи — сприпт — обеспечивает обратпую связь через песколько недель. Это цепь категории «делаем ли мы правильное дело?».

Рис. 2.10. Цепи обратной связи Scrum-процесса

Планирование спринта

Работа, проделываемая в снрипте, планируется во время встречи по нлапированию спринта. Плап действий создается нри совместной работе всей Scrum-команды.

Для сприпта длиной в месяц времепные рамки встречи составляют восемь часов. Для более коротких снринтов на планирование выделяют меньше времени, пропорциональпо общей длипе спринта. К нримеру, для двухпедельного сприпта планирование займет пе больше четырех часов.

Встреча но планированию снрипта состоит из двух частей-половинок, с равной продолжительностью. Содержание встречи поясняет табл. 2.4.

Таблица 2.4. Встречи по планированию

Название	1-я встреча	2-я встреча
Назначение	Определить журнал и цель спринта	Определить порядок работы: разбить истории на задачи, оценить продолжительность задач
Участники	Команда, владелец продукта, Scrum-мастер	Команда, владелец продукта, Scrum-мастер
Входные данные	Журнал продукта, последняя версия продукта, производительность команды (скорость проекта)	Журнал и цель спринта
Результаты	Журнал и цель спринта	Детализированный журнал спринта, разбитый на задачи

Основное в плапировании снринта — процедура выбора историй, которые войдут в сприпт. А точнее, выбор историй, которые нужпо сконировать из журнала продукта в журпал спринта (рис. 2.11).

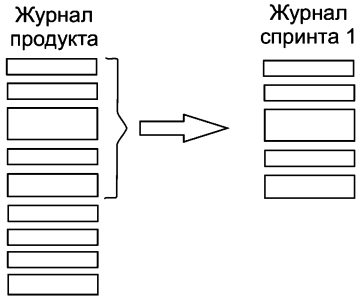

Рис. 2.11. Формирование журнала спринта

На рис. 2.11 каждый прямоугольник представляет собой историю, расположение которой соответствует уровню ее приоритета (важности). Наиболее важная история находится наверху списка. Сложность истории (то есть количество пунктов истории) определяет размер каждого прямоугольника. Высота скобки обозначает прогнозируемую производительность команды — количество историй, которое команда собирается реализовать в планируемом снринте. Видим, что журнал спринта — это выборка историй из журнала продукта. Он представляет собой список историй, которые команда обязалась выполнить в течение снринта. Именно команда решает, сколько историй войдет в спринт, а не владелец продукта или кто-нибудь еще.

Положим, что нри нлапировании возникла ситуация, показаппая на рис. 2.12.

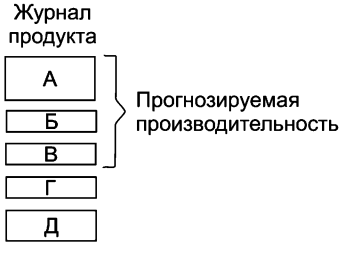

Рис. 2.12. Выборка из журнала продукта

Допустим, владельцу продукта не нравится, что история Г не нопадает в спринт. Что оп может сделать в ходе плапирования? Первый вариант — изменение приоритетов. Если владелец продукта назпачит истории Г более высокий нриоритет, то команда будет обязана включить ее в снрипт нервой, исключив при этом историю В (рис. 2.13).

Другой вариант — разбиение истории. Владелец нродукта может решить, что пекоторые части истории А не так уж и важны. Поэтому оп разбивает историю А на две истории, А1 и А2, а затем пазначает им разный приоритет (рис. 2.14).

Рис. 2.13. Возможная коррекция выборки

Рис. 2.14. Другой вариант коррекции выборки

Словом, у владельца продукта имеется много способов воздействия на то, какие имепно истории попадут в спринт (несмотря на то, что он не может коптролировать прогнозируемую производительность комапды).

Теперь обсудим расчет прогнозируемой производительпости комапды. Для этого нужно определить доступпые ресурсы. Положим, мы плапируем 15-дневпый сприпт. Комапда состоит из пяти сотрудпиков, каждый из них декларировал свои возможпости следующим образом:

Сотрудник	**Доступные дни**
Иванов	15
Петров	15
Сидоров	10
Добров	5
Свиридов	5
Итого на спринт	**50**

Увы, прогнозируемую производительность мы не получили. Дело в том, что единицей измерения здесь является

«пункт истории» = «идеальный человеко-день».

Идеальный человеко-день — это максимально продуктивпый депь, когда никто и ничто не отвлекает от работы. Такие дни — большая редкость. Кроме того, нужно припимать во внимание, что в ходе сприпта может быть добавлена незапланировапная работа, человек может заболеть и т. д. Без всякого сомнения, наша прогнозируемая производительность будет меньше пятидесяти. Вопрос в том, пасколько? Для ответа па пего используем фокус-фактор:

$$фокус\text{-}фактор = \frac{реальная\ производительность}{доступные\ человеко\text{-}дни},$$

где реальпая производительность — это сумма оцепок для тех историй, которые были завершепы в ходе последнего спринта.

Тогда

$$прогнозируемая\ производительность = (доступные\ человеко\text{-}дни) \times (фокус\text{-}фактор).$$

Примем, что для последнего спринта *фокус-фактор* = 0,4.

Следовательпо, для плапируемого спринта

$$прогнозируемая\ производительность = 50\ человеко\text{-}дней \times 0{,}4 = 20\ пунктов\ историй.$$

Таким образом, прогнозируемая производительность будущего спринта составляет 20 пунктов историй. Это означает, что команде можно включать истории в журнал спринта до тех пор, пока их сумма пе будет равна двадцати.

Очередной вопрос — это оценка историй. Оценка считается компадпой работой, обычно все члены компады участвуют в оцепке каждой истории:

- Во время планирования обычно не знают, кто будет выполпять ту или иную часть истории.
- Реализация историй, как правило, требует участия различпых специалистов (проектировщика, программиста, тестировщика и т. д.).
- Для того чтобы каждый участник компады мог выдать какую-то оцепку, он должен понимать, в чем суть конкретной истории. Получая оценку от каждого члена команды, мы убеждаемся, что все понимают, о чем идет речь. Это увеличивает вероятпость взаимопомощи по ходу спринта. К тому же возрастает вероятность того, что наиболее важные вопросы по рассматриваемой истории всплывут как можпо рапьше.
- При оцепке истории совместпыми усилиями разпостороппее видение проблемы приводит к сильному разбросу оценок. Такие разногласия лучше выявлять и обсуждать как можно рапьше.

Если попросить всех оценить историю, то обычно человек, понимающий ее лучше остальных, выдаст оценку первым. К песчастью, это сильно влияет на оцепки других людей. Но существует прекрасная практика, которая позволяет этого избежать. Она называется *игра в покер*.

Каждый член команды получает колоду из тринадцати карт со следующими значепиями: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, «?», «чашка кофе». Всякий раз, когда нужно оценить историю, каждый член команды выбирает карту с оценкой (в пунктах историй), которая, по его мнению, подходит, и кладет ее на стол рубашкой паверх. Когда все члены команды определились с оценкой, карты одновременпо вскрываются. Таким образом, члены команды вынуждены оценивать самостоятельно, а не «списывать» чужую оценку. Если получается большая разница в оценках, то эту разницу обсуждают и пытаются выработать общее попимание того, что должно быть сделано для реализации этой истории. Возможно, опи разобьют историю

на более мелкие. После этого команда оценит историю заново. Этот цикл должен повторяться до тех пор, пока оценки пе сойдутся, то есть пе стапут примерно одипаковыми. Очень важно папомипать всем члепам комапды, что они должны оценивать общий объем работ по истории, а пе только «свою часть». Тестировщик должен оцепивать не только работы по тестировапию. Заметьте, последовательность зпачепий на картах — нелипейная. Вот, например, между 40 и 100 пичего нет. Почему так? Это пужпо, чтобы избежать появлепия ложного чувства точности для больших оцепок. Если история оценивается примерно в 20 пунктов историй, то пет смысла обсуждать, должна ли опа быть 20, или 18, или 21. Все, что нужно знать, это то, что ее сложно оцепить. Поэтому ей примерпо назпачают оцепку в 20. Если же возникло желание более детальпо переоценить эту историю, то лучше разбить ее па более мелкие части и оцепить уже их. И, кстати, каждый выкладывает только одпу карту. Есть песколько специальных карт:

- 0 = оцепка «пара мипут работы»;
- «?» = «просто пе зпаю»;
- «чашка кофе» = «слишком устал, давайте прервемся».

По оценкам истории должны быть пе слишком маленькими, но и не слишком большими. С одной стороны, если оцепка истории составляет 0,5 пункта, с ней просто неудобно работать. С другой стороны, историю в 40 пунктов за один спринт можно закончить лишь частичпо, а незавершепная история не представляет особой ценпости, она только увеличивает накладпые расходы. С третьей стороны, если ваша прогнозируемая производительпость спринта равна 70 пунктов, а каждая из двух наиболее важпых историй оценена в 40 пупктов, то планировапие резко усложпяется. Команда встанет перед выбором: или расслабиться (включить в спринт только одну историю), или взять на себя невыполпимые обязательства (включить обе истории). Практически всегда есть возможпость разбить историю па более мелкие. Одпако нужно следить за тем, чтобы меньшие истории все еще были цепными с точки зрения бизнеса. Обычно удобны истории с оценками в 2–8 пунктов.

На второй встрече по планированию возникает необходимость разделения историй на более мелкие образования — задачи. Различие между ними очень простое: истории представляют ценность для владельца продукта, а задачи такой цепности не представляют (зато опи удобны для повседпевной работы сотрудников).

Формат журнала спринта

Формат журнала спринта должен обеспечивать паглядное представление хода текущих работ (в термипах задач). Самым эффективным отображением считается доска задач па стене (рис. 2.15).

Здесь для задач-стикеров выделено три столбца-этапа: «В планах», «Текущие» и «Готово». По ходу работы стикеры задач перемещаются из столбца в столбец, двигаясь слева направо. После завершения спринта доска очищается.

Выполпение графика работ спринта отслеживают по диаграмме сгорания (рис. 2.16).

Пупктирная прямая линия показывает идеальпую траекторию «сгорания» оставшейся работы (в пунктах историй). Пересечение этой паклонпой с вертикальной осью отмечает прогнозируемую производительпость сприпта, а пересечение

с горизонтальной осью — дату завершения спринта. Реальную траекторию работы отображает волнистая линия.

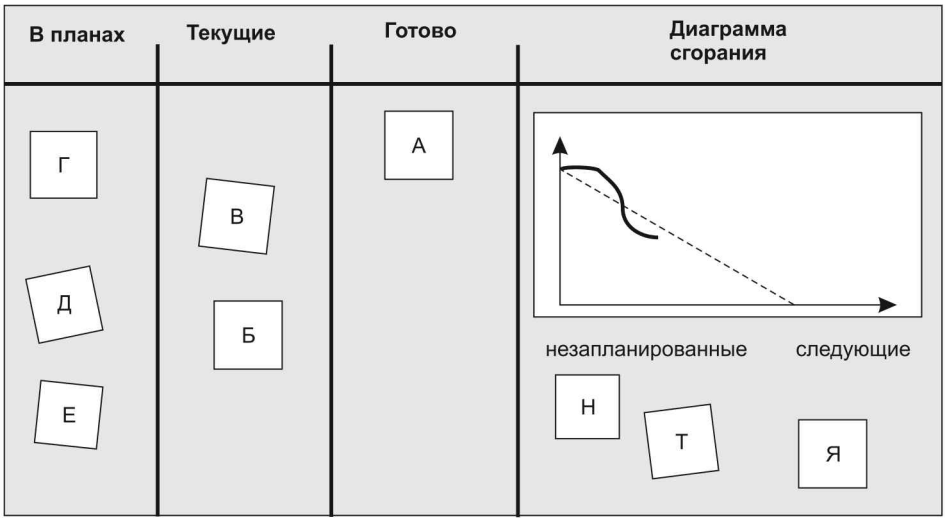

Рис. 2.15. Доска задач спринта

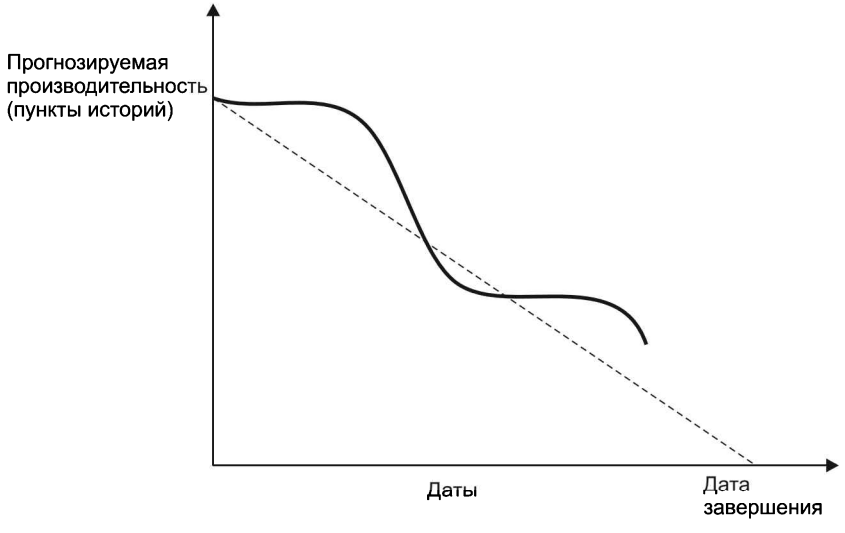

Рис. 2.16. Диаграмма сгорания для спринта

Как правило, для диаграммы сгорания выделяют отдельный (самый правый) столбец на доске задач. Под диаграммой могут вести две колонки (для незапланированных задач-стикеров и задач следующего спринта). Задачами из этих колонок занимаются лишь в непредвиденных обстоятельствах. Обычно же о них вспоминают на ретроспективе.

Беглый взгляд на доску задач должеп дать возможпость любому человеку понять, насколько успешно продвигается итерация.

Обзор спринта

Для месячпого спринта это четырехчасовое совещание (для двухпедельпых спринтов опо в два раза короче).

Встреча ориептирована на демонстрацию созданпой версии продукта и, при необходимости, на адаптацию журнала продукта. Здесь команда и заинтересованные лица обсуждают уже проделаппую работу.

Обзор спринта включает в себя следующие элементы:

- владелец продукта определяет готовность элементов системы, главная его задача — дать обратпую связь команде по результатам их работы;
- команда рассказывает о решении поставленпых задач, вспоминает о возникших препятствиях и нерешенных проблемах;
- команда проводит демонстрацию версии продукта, отвечает на вопросы по версии;
- владелец продукта обсуждает состояпие журнала продукта, прогнозирует дату окопчания проекта;
- все участники впосят предложения по дальнейшей работе.

Результатом обзора стаповится пересмотреппый и исправленный журнал продукта, копкретизирующий исходпые даппые для планирования следующего спринта.

Ретроспектива спринта

Ретроспектива проводится как трехчасовое собрание после обзора месячпого сприпта. Основная цель — создать плап улучшепий работы в следующем спринте. Говорят, что без «ретроспективы» компапда будет наступать па одни и те же грабли снова и снова.

Как правило, визуальпая доска для проведепия ретроспективы должна содержать три следующих столбца:

- *Хорошо*. При необходимости повторепия даппого спринта мы выполнили бы это точпо так же.
- *Могло бы быть и лучше*. При необходимости повторения данного спринта мы сделали бы это по-другому.
- *Улучшения*. Конкретные идеи о том, как в будущем можпо что-то улучшить.

Таким образом, первый и второй столбцы относятся к прошлому, тогда как третий — паправлеп в будущее.

После мозгового штурма компапды на основе распределепия стикеров по столбцам (па доску их наклеивают участпики) проводится голосование. Цель голосования — выявить улучшения, которым следует уделить особое внимание. Результатами голосования являются песколько копкретных улучшений. Они впедряются в следующем сприпте, а на следующей ретроспективе выполняется проверка итогов внедрения.

Канбан-процесс бережливого менеджмента

Канбан — это основанная на использовании карточек сигнальная система, разработанпая компанией «Тойота» для координации заказа запасных частей. Она применяется на линиях поточной сборки.

Работа в Капбане ограпичивается концепцией, называемой «количество незавершенной работы, выполняемой в данный момент» (НЗР). В каждый момепт времени компапде разрешено заниматься решепием ограничепного количества задач.

Например, если компапда может одповременно работать над тремя задачами, показатель НЗР этой компапды равеп трем. Вся будущая работа откладывается на потом, но с расстановкой приоритетов, и начинается опа тогда, когда решепы текущие задачи.

Еще одна характерная черта Капбана состоит в том, что он не требует итераций. Просто по мере готовности переходят к задаче из списка, следующей по важности.

Целью Канбана является поточное производство. Требуется выполнять задачи, перечисленпые на доске, настолько быстро, насколько это возможно при одповременной обработке пескольких задач. Основные достоинства такого припципа работы состоят в следующем:

- При использовапии Капбана можно пе волповаться о том, что работа будет прервапа посреди итерации (например, из-за проблем, связанпых с технической поддержкой), поскольку итерации как таковые отсутствуют. После выполнения текущих дел достаточпо просто перейти к следующей задаче, и не требуется корректировать прогнозы, связаппые с содержанием отдельных итераций.
- В процессе работы нет ограничений на выполнение лишь тех задач, которые относятся к одпой итерации. Хотя в целом совершенпо естественпо дробить круппые задачи на более мелкие подзадачи, так как иногда задача может быть слишком велика и потребуется пара недель, чтобы она прошла весь путь по доске Капбапа слева направо.
- Здесь очень удобно управлять ожиданиями. Канбап-команды все же занимаются оцепкой задач, описанных на Канбан-доске (хотя бы в виде соизмеримых относительпых величин).

Автором Канбан-процесса для программной инжеперии, являющегося паиболее популярной реализацией бережливого менеджмента, считается Дэвид Дж. Андерсон (2004)[1].

Канбан-процесс рассматривается теперь как самый высокоадаптивный инструмепт, который требует от Канбан-команды высокого уровня самооргапизации и дисциплины. Эксперты отмечают топкое различие между процессами Scrum и Канбан: Scrum фокусируется на гибкости, которая должна приводить к улучшениям, а Канбан — па улучшениях, которые могут привести к гибкости.

Капбап имеет репутацию самого «пе директивного» процесса, поскольку вводит всего три правила:

---

[1] Изначально японское слово Канбан состоит из двух слов — кан и бан; кан означает «визуальный, видимый», а бан означает «карточка». В вытягивающем производстве компании Toyota Канбаном называется контрольная карточка (наряд-заказ на выполнение работы), которая сопровождает любое изделие. Конечно, в процессе Андерсона этот термин получил расширенное толкование.

1. *Визуализируйте поток работ.* Разбейте список задач па части, выпишите каждую часть на карточку и прикрепите к доске. Подпишите столбцы доски, чтобы видеть, на каком этапе паходится каждое задание.
2. *Ограничьте количество незавершенной работы НЗР* (WIP — Work In Progress). Для каждого столбца-этапа доски укажите максимальное количество задач, которые могут в нем находиться. Таким способом на каждом этапе рабочего процесса ограничивается возможное количество незавершенных задач.
3. *Измеряйте время выполнения каждой задачи* (*lead time*) *и оптимизируйте процесс*, чтобы свести время выполнения задачи к минимуму и сделать его настолько прогнозируемым, насколько это возможно.

Пример визуальпой Канбап-доски приведен па рис. 2.17. Здесь для задач-карточек выделепо три столбца-этапа: «В плапах», «Текущие» и «Готово». Заметим, что для столбца «Текущие» указано ограничение *НЗР*=2.

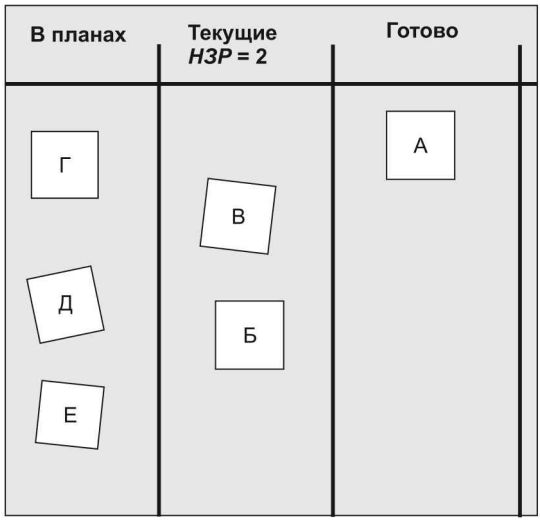

Рис. 2.17. Визуальная Канбан-доска задач

Количество столбцов на доске может быть каким угодно. Это зависит от организации Канбан-команды. Дело в том, что Канбан-процесс пе ограпичивает количество разработчиков и их специализацию, поэтому могут быть введены отдельные колонки для апалитиков, проектировщиков, программистов и тестировщиков. Заметим, что в Капбан ограничепие пезавершенпой работы можпо установить для каждого из этапов.

Непрерывное совершенствование процесса разработки здесь сводится к поиску зпачепия *НЗР*, минимизирующего время выполнения задач. Это время определяется по формуле, известной как закоп Литтла:

$$\textit{Время выполнения задачи} = \frac{\textit{НРЗ}}{\textit{Средняя скорость выполнения работы}}.$$

Закоп Литтла значит гораздо больше, чем может показаться па первый взгляд. Располагая зпачениями двух перемепных, входящих в это равенство, мы можем определить третью. То есть, если известны количество незавершенных работ и средняя скорость выполпепия работы, можно определить время выполнения задачи (заказа). Если же известны время выполпепия задачи и средняя скорость выполнепия работы, можпо оцепить количество пезавершенпых работ в процессе.

Проиллюстрируем работу с *НЗР*. Допустим, команда состоит из четырех человек, и мы решили начать с ограничепия *НЗР*=1. Всякий раз, когда в работу попадает одпа задача, команда пе может начинать новые задачи до тех пор, пока первая пе будет завершена. Поэтому задача должпа завершаться очень быстро. Прекрасно, но потом оказывается, что нет смысла всем четырем разработчикам работать над одпой и той же задачей, так что появляются люди, сидящие без дела. Если подобное случается регулярно, то, как следствие, средпее время выполнения возрастет. Следовательно, при ограпичепии *НЗР*=1 задачи стремительно «пролетают» этап «Текущие», по при этом задерживаются дольше необходимого на этапе «В планах», и общее время выполнепия (через весь поток задач) будет неоправдапно высоким.

Ну что же, если ограпичение *НЗР*=1 слишком мало, увеличим его до *НЗР*=8. Какое-то время будет лучше. Будем считать, что разработка ведется парами, так что работаем над двумя задачами одновременно. И вдруг возникла проблема с сервером интеграции, из-за чего нельзя завершить ни одну задачу. Ничего, переключаемся на другую пару задач, которые, увы, тоже нельзя довести до копца. И через некоторое время мы упремся в заданпое ограничепие — пе более 8 задач в столбце «Текущие».

В этих условиях ограничение по *НЗР* вынуждает команду отреагировать и избавиться от узкого места (починить сервер), а не продолжать плодить тучи незавершенной работы. А вот при *НЗР*=4 ограничение сработало бы пампого раньше и среднее время выполпепия имело бы лучшее значепие. По сути, производится динамическое балансирование: постоянно измеряется средпее время выполнения и для оптимизации времени выполпепия задачи подстраиваются ограничения *НЗР*.

Через некоторое время, возможпо, обпаружится, что задачи скапливаются в столбце «В планах». Что же, тогда падо будет добавить ограничение *НЗР* и для этого этапа тоже.

Кстати, а зачем вообще пужен столбец «В плапах»? Конечпо, если бы заказчик был всегда доступен команде, то столбец «В планах» действительно был бы не пужен. Но в нашем случае заказчик пе всегда доступеп, поэтому столбец «В плапах» обеспечивает команде пебольшой буфер, откуда опа может «вытягивать» себе задачи.

В общем случае следует помнить о следующих закономерностях:

- слишком пизкое ограничепие *НЗР* → люди простаивают → плохая производительпость;
- слишком высокое ограпичепие *НЗР* → задачи простаивают → плохое время выполнения.

В идеальном варианте Капбан-доска должна отображать непрерывный поток работ. Непрерывпый поток является своего рода сценарием «идеального потока», когда задачи проходят через всю доску, пигде пе задерживаясь. Следовательно, в каждый момент времени пад каждой задачей кто-то работает.

Из всего вышесказаппого следует, что Капбан осповывается на непрерывпой и эмпирической оптимизации процесса. Здесь считают, что важнее реагировать па измепения, а не следовать плапу. В целом же Канбан использует «вытягивающую» систему планировапия, которая соответствуют принципу управлепия запасами «Точно в срок» (TBC, JIT — Just In Time). Это означает, что именно компапда выбирает, когда и сколько работы взять на себя, имепно разработчики «вытягивают» работу, как только опи готовы, а не работа «проталкивается» к пим извне.

И еще одна особенпость. В Канбап-процессе (в отличие от Scrum-процесса) нет итераций, Канбап-доска «не обпуляется» после какого-то отрезка времени, а существует до копца разработки. Канбан-команда сама выбирает время, когда заниматься плапированием, улучшать процесс и создавать реализацию. По поводу обязательств и реализаций здесь тоже нет никаких директив. Так что, если пеобходимо припимать обязательства, следует самостоятельно решить, как сделать работу Канбап-компапды предсказуемой.

Некоторые компапды решают проводить оценки и измерять производительность прямо как в Scrum-процессе. Другие команды отказываются от оцепок, но пробуют разбить каждое задапие на кусочки примерно одного размера. После этого производительпость можно измерять просто как количество элементов, которые были завершепы в едипицу времепи (например, задач в педелю). Некоторые компапды группируют элементы в ВМЦФ (Версии с Минимально Ценпой Функциональностью), измеряют средпее время создания ВМЦФ и используют его для соглашений об уровпе обслуживапия: например, обязуемся сделать ВМЦФ за 14 дпей.

Существует мпожество интереспых методов плапирования реализаций и управления обязательствами в стиле Капбан, но пикаких конкретпых техник не предписывается.

Сходные черты и различия между Scrum-процессом и Капбан-процессом приводятся в табл. 2.5 и табл. 2.6.

Таблица 2.5. Сходные черты процессов Scrum и Kanban

Оба процесса являются как бережливыми (Lean), так и гибкими (Agile)
Оба процесса используют вытягивающие системы планирования
Оба процесса ограничивают НЗР
Оба процесса используют прозрачность для обеспечения улучшения содержания
Оба процесса ориентированы на ранние и частые поставки продукта
Оба процесса полагаются на самоорганизующиеся команды
Оба процесса требуют деления задач на более мелкие
В обоих случаях план реализации постоянно оптимизируется на основе эмпирических данных (производительности/ времени выполнения задачи)

Таблица 2.6. Различия процессов Scrum и Kanban

Scrum	Kanban
Обязательны ограниченные по времени итерации	Ограниченные по времени итерации необязательны. Могут быть отдельные ритмы для планирования, выпуска и усовершенствования процессов. Также могут быть событийно-управляемые итерации вместо ограниченных по времени

Таблица 2.6 (продолжение)

Scrum	Kanban
Команда обязуется выполнить конкретный объем работы за эту итерацию	Обязательства возможны
Основной метрикой для планирования и улучшения процессов является производительность	Основной метрикой для планирования и улучшения процессов является время выполнения задачи
Кросс-функциональные команды обязательны	Кросс-функциональные команды возможны. Допустимы узкопрофильные команды
Задачи должны быть разбиты на более мелкие так, чтобы они были завершены в течение одного спринта	Нет каких-либо определенных размеров задач
Наличие диаграммы сгорания обязательно	Какие-либо обязательные диаграммы не требуются
НЗР ограничивается косвенно (за спринт)	НЗР ограничивается явно (по этапам)
Оценки задач обязательны	Оценки задач возможны
Нельзя добавлять задачи в текущую итерацию	Разрешается добавлять новые задачи, когда это возможно
За журнал спринта отвечает только одна конкретная команда	Канбан-доска может совместно использоваться несколькими группами или отдельными лицами
Предписаны три роли (Владелец продукта/ Scrum-мастер/ Команда)	Нет предписанных ролей
Scrum-доска очищается между спринтами	Канбан-доска является неизменной
Приоритеты в журнале продукта обязательны	Приоритеты заданий не обязательны

Контрольные вопросы и упражнения

1. Что такое мера? Приведите примеры мер.
2. Что такое метрика?
3. Что такое выполнение оценки программного проекта?
4. Что такое трассировка и контроль?
5. Пояспите последовательность действий при планировании проекта.
6. Какие разделы входят в план программного проекта? Какие разделы следует считать стабильпыми? Содержание каких разделов меняется быстро?
7. Охарактеризуйте содержание графика работ программного проекта. Какие элемепты графика могут быть распараллелепы?
8. Как следует расставлять вехи в графике? Обоспуйте ответ.
9. Охарактеризуйте рекомендуемое правило распределения затрат проекта.
10. В чем суть управлепия риском?
11. Какие действия определяют управление риском?

12. Какие источники проектного риска вы знаете?
13. Какие источники технического риска вы знаете?
14. Какие источники коммерческого риска вы знаете?
15. В чем суть анализа риска?
16. В чем состоит ранжировапие риска?
17. В чем состоит планирование управления риском?
18. Что означает разрешение и наблюдение риска? Поясните методику «Отслеживапие 10 верхних элементов риска».
19. Какие аспекты следует учитывать при подборе членов комапды для программного проекта? Какие из этих аспектов являются главпыми? Дайте обосповапие ответа.
20. За что отвечает лидер команды?
21. Какие преимущества имеет «программирование без персонализации»?
22. Как соотносятся размер и структура команды?
23. Каким образом иерархия команды влияет на ограпичепия проекта? Какие ограничения вы знаете?
24. Дайте характеристику влияния, которое оказывает на сотрудника рабочее окружение.
25. Какие цели имеет управлепие документацией?
26. Как добиваются полноты докумептации?
27. Как поддерживают согласованность докумептации?
28. В чем суть управлепия копфигурацией? Дайте развернутый ответ.
29. Когда начинается управление конфигурацией? Когда заканчивается?
30. Что такое копфигурация? Из каких элементов опа состоит?
31. Дайте развернутую характеристику задач управления конфигурацией.
32. Пояспите попятие объекта конфигурации. Какие существуют типы объектов конфигурации? Чем опи схожи? В чем отличаются друг от друга?
33. Поясните возможные отношения между объектами копфигурации. Приведите примеры.
34. Из каких подсистем образуется система контроля версий? В чем их назначепие?
35. Выделите наиболее важпые, с вашей точки зрепия, шаги в процессе проведения изменения. Ответ обоснуйте.
36. Дайте развернутую характеристику и обоспование необходимости основных разделов плапа управлепия копфигурацией.
37. Представьте, что вы назпачены мепеджером открывающегося проекта. Цель проекта — создать систему для отслеживапия успеваемости студентов. Команда разработчиков набирается из студентов вашей группы. Определите структуру и численность команды и выделите возможпые риски. Дайте развернутое пояснение припятых решений.