Простое руководство по Теории и Практике Скрама

Версия 2.0

Пит Димер
GoodAgile
www.goodagile.com

Габриель Бенефилд
Evolve
www.evolvebeyond.com

Крэг Ларман
www.craiglarman.com

Бас Водди
Odd-e
www.odd-e.com

Примечание для читателей: В сети много лаконичных описаний Скрама, а это руководство для начинающих помогает достичь следующего уровня в понимании практики. Оно не является конечным шагом в изучении Скрама; командам, которые рассматривают возможность перехода на Скрам, рекомендуется вооружиться книгой *Скрам. Гибкое управление продуктом и бизнесом* Кена Швабера или Scrum. *Гибкая разработка ПО* Майка Кона, также воспользоваться многими превосходными вариантами обучения и коучинга по Скраму, которые доступны; больше информации вы можете найти на сайте scrumalliance.org. Выражаем благодарность Кену Швабер у, Джеффу Сазерленду и Майку Кону за их щедрый вклад.

Последняя версия Scrum Primer (Азбуки Скрама) расположена по адресу:
http://www.infoq.com/minibooks/Scrum_Primer

Переводы можно найти по адресу: http://www.scrumprimer.org/

Перевод осуществлён Кротовым Артёмом и Романом Лапаевым.

За Пределами Традиционной Разработки

Традиционная разработка программного обеспечения с однофункциональными подразделениями, с запаздывающей и слабой обратной связью, предварительным прогнозным планированием на начальном этапе и последовательным переходом начиная с анализа и до тестирования не очень успешна в сегодняшнем нестабильном мире. Такой подход задерживает получение обратной связи, обучение и потенциальный возврат от инвестиций из-за отсутствия реального работающего программного обеспечения до конца игры, вызывая отсутствие прозрачности, отсутствие возможности улучшать, снижение гибкости, увеличение технических и бизнес рисков.

Альтернатива - кросс-функциональные команды с итеративной разработкой - также существуют уже несколько десятилетий, но не так широко распространены, как традиционная модель.

Скрам объединяет проверенные концепции разработки продуктов в простую структуру, в том числе: настоящие команды, кросс-функциональные команды, самоуправляемые команды, короткие петли обратной связи в полных циклах разработки и снижение стоимости изменений. Эти концепции повышают гибкость и обратную связь, обеспечивают более раннюю окупаемость инвестиций и снижают риски.

Обзор

Скрам - фреймворк разработки, в котором кросс-функциональные команды разрабатывают продукты или проекты в итеративно-инкрементальном стиле. Он структурирует процесс разработки в циклы, называемые **Спринтам**. Эти итерации не могут быть длиннее четырёх недель (обычно две недели), и идут одна за одной без остановки. Спринты *ограничены по времени* - они заканчиваются в определённые даты, независимо от того, была ли закончена работа или нет, и *никогда не продлеваются*. Обычно Скрам-команды выбирают длину Спринта один раз, а затем придерживаются её до того, как внедрят улучшения и смогут её сократить. В начале каждого Спринта *кросс-функциональная* **Команда** (около 7 человек) выбирает элементы (требования пользователей) из приоритизированного списка. Команда договаривается об общей цели, в которую они верят и могут поставить в конце Спринта, что-то материальное и что будет действительно ”готово”. Во время Спринта новые элементы не могут быть добавлены; Скрам принимает изменения только в следующий Спринт, но текущий короткий Спринт предназначен для достижения небольшой, четкой и относительно стабильной цели. Каждый день Команда собирается ненадолго, чтобы провести инспекцию прогресса и адаптировать следующие шаги, необходимые для завершения оставшейся работы. В конце Спринта Команда проводит его обзор вместе с заинтересованными лицами, демонстрирует то, что готово. Люди получают обратную связь, которая может быть учтена в следующем Спринте. Скрам подчёркивает, что работающий продукт должен быть по-настоящему “готов“ в конце Спринта; в случае с разработкой программного обеспечения это означает, что система интегрирована, полностью протестирована, содержит документацию для конечных пользователей и потенциально готова к поставке. Ключевые роли, артефакты и события приведены на Иллюстрации 1.

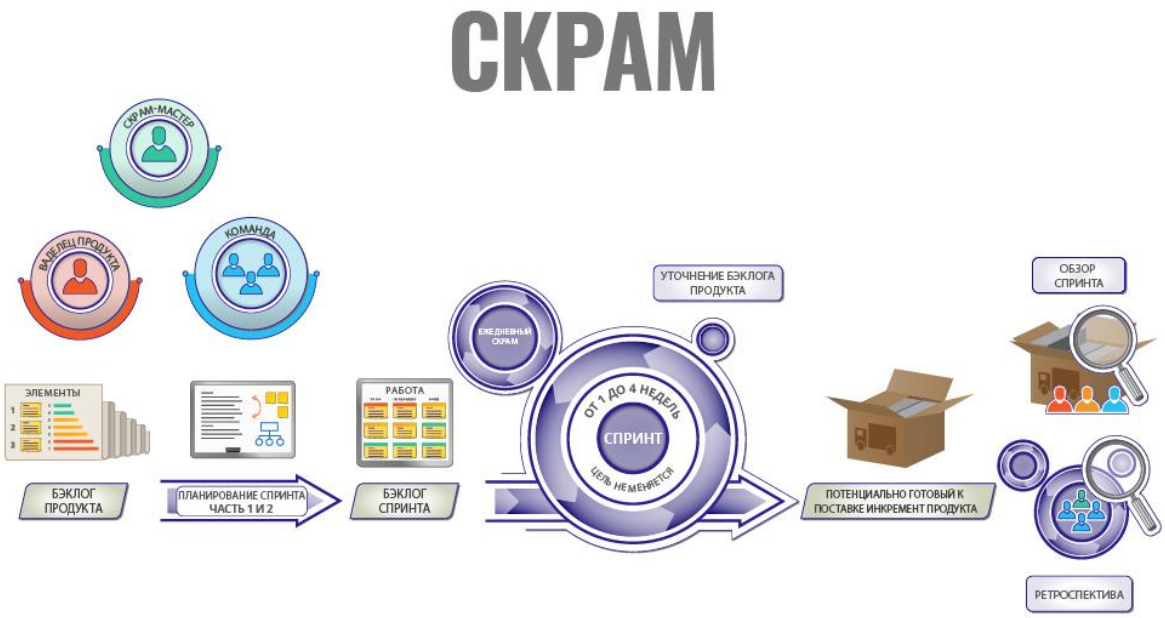


Иллюстрация 1. Обзор Скрама

Важная вещь в Скраме - “инспекция и адаптация”. Поскольку разработка неизбежно связана с обучением, инновациями и неожиданностью, Скрам подчеркивает что нужно вести разработку небольшими шагами, тем самым регулярно инспектировать как конечный продукт, так и эффективность текущих практик, а затем адаптировать продуктовые цели и эти процессные практики. *Повторяйте это бесконечно.*

Роли

В Скраме всего три роли: Владелец Продукта, Команда и Скрам-мастер. Вместе они известны, как Скрам-команда.

Владелец Продукта (Product Owner, PO) несёт ответственность за максимизацию возврата на инвестиции (ROI) путём определения новой функциональности для продукта, перевода её в приоритизированный список, решая, что должно быть сверху этого списка на следующий Спринт, и непрерывной реприоритизации и прояснения этого списка. Владелец продукта несёт ответственность за доходы и расходы (PNL, profit and loss) продукта, если это коммерческий продукт. В случае внутреннего продукта в компании Владелец продукта не несёт ответственность за ROI в смысле коммерческого продукта (который генерирует доход), но на нём всё же лежит ответственность за максимизацию ROI в смысле выбора - каждый Спринт - самых ценных элементов. На практике, ‘ценность’ является нечётким термином, поэтому на приоритизацию могут влиять: желание удовлетворить ключевых клиентов, соответствие стратегическим целям, снижение рисков, улучшение и ряд других факторов. В некоторых случаях Владелец Продукта и клиент один и тот же человек; это обычная практика для внутренних приложений. В других случаях клиентами могут быть миллионы человек с разными потребностями, поэтому тут роль Владелец Продукта похожа на должность Менеджера Продукта (Product Manager, PM) или Менеджера по Маркетингу (Product Marketing Manager) в разных организациях. Однако Владелец Продукта — это нечто отличное от традиционного Менеджера Продукта, потому что он активно и часто взаимодействует с Командой, приоритизирует, учитывает мнения всех заинтересованных лиц (stakeholders, SH), и делает обзор результатов каждый Спринт, а не делегирует решения по разработке Менеджеру Проекта (Project Manager, PM). Важно отметить, что в Скраме один и только один человек, который играет эту роль - и несёт полную ответственность за продукт - Владелец Продукта, и он(а) ответственны за ценность работы; хотя этому человеку не обязательно работать в одиночку.

Команда (Team)**, официально называемая Командой Разработки** (Development Team, DT)**,** создаёт такой продукт, на какой указывает Владелец Продукта: приложение или веб-сайт, например. Команда в Скраме "кросс-функциональна" - она содержит все виды экспертизы, необходимые для поставки потенциально готового продукта каждый Спринт - она "самоорганизованная" (самоуправляемая), с высокой степенью автономии и ответственности. Команда решает, как много элементов (из предложенного Владельцем Продукта набора) взять в разработку в Спринте, и как лучше всего это сделать.

Каждый член Команды просто *член команды*. Заметьте, что в группе, которая внедряет Скрам, нет четких должностей; здесь нет бизнес-аналитика, нет администратора баз данных, нет архитектора, нет тимлида, нет UX/UI-дизайнера, нет программиста. Они работают вместе в течение каждого Спринта любым способом, подходящим для достижения цели, которую они поставили перед собой.

Поскольку есть только *члены команды*, Команда не только является кросс-функциональной, но также демонстрирует *множественное обучение* (multi-learning): каждый человек, безусловно, имеющий более сильные стороны в определённой области, также продолжает изучать и другие направления. Каждый человек имеет первичные, вторичные, и даже третичные навыки, что означает "следуй туда, где есть работа" (go to where the work is); он берёт на себя задачи в менее знакомых ему областях, чтобы помочь завершить элемент полностью. Например, человек с первичными навыками в дизайне мог бы иметь вторичный навык в автоматизации тестирования; кто-то с первичными навыками в написании технической документации мог бы также помочь с анализом и написанием кода.

Команда в Скраме состоит из 7 плюс/минус 2 людей, и для программного продукта Команда могла бы включать людей с навыками аналитики, разработки, тестирования, проектирования интерфейсов, проектирования баз данных, архитектуры, документации, и так далее. Команда развивает продукт и предоставляет идеи Владельцу Продукта, как сделать продукт лучше. В Скраме, Команды более продуктивны и эффективны, если все её члены на 100 процентов выделены на работу в одном продукте в течение всего Спринта; Команда избегает многозадачности в нескольких продуктах или проектах, чтобы предотвратить увеличение их стоимости из-за рассеянного внимания и переключение контекста. Стабильные команды достигают высокой продуктивности, поэтому избегайте смены членов Команды. Продуктовые группы с большим количеством людей могут организоваться в несколько Команд, каждая из которых может концентрироваться на разной функциональности продукта, с тесной координацией их совместных усилий. Как только одна команда начинает постоянно делать всю работу (планирование, анализ, программирование и тестирование) в задачах, ориентированных на конечного клиента, то такую Команду называют *фиче-командой*.

Скрам-мастер (Scrum Master, SM) помогает продуктовой группе изучить и внедрить Скрам для получения бизнес ценности. Скрам-мастер делает всё, что в его власти, чтобы помочь Команде, Владельцу Продукта и организации быть успешными. Скрам-мастер *не является* руководителем членов Команды или менеджером проекта, тимлидом или их представителем. Вместо этого, Скрам-мастер *служит* Команде; он(а) защищает Команду от внешнего воздействия, помогает устранить препятствия и внедрить современные инженерные практики. Он(а) учит, тренирует и наставляет Владельца Продукта, Команду и всю остальную организацию в правильном использовании Скрама. Скрам-мастер — *коуч* и *учитель*. Скрам-мастер удостоверяется, что все (включая Владельца Продукта и менеджмент) понимают принципы и практики Скрама, и он помогает вести организацию через зачастую трудные изменения, необходимые для успеха в гибкой разработке ПО. Поскольку Скрам делает заметными множественные препятствия и угрозы эффективности Команды и Владельца Продукта, то важно иметь вовлечённого Скрам-мастера, энергично работающего над разрешением данных проблем, иначе Команде и Владельцу продукта будет сложно добиться успеха. Скрам-мастер должен быть выделенной ролью на постоянной основе, однако в маленькой Команде член команды может играть его роль (неся при этом меньшую нагрузку из регулярной работы). Великие Скрам-Мастера могут иметь любой опыт или специализацию в прошлом: Инженерия, Дизайн, Тестирование, Продуктовый Менеджмент, Менеджмент Проектов или Менеджмент Качества.

Скрам-мастер и Владелец Продукта не могут быть одним и тем же человеком, потому что их фокусы настолько различны, что часто их совмещение ведёт к путанице и конфликтам. Одни из нежелательных результатов совмещения этих ролей проявляется в микроменеджменте (micro-managing) Владельца Продукта, что противоречит самоуправляемости команд, которую требует Скрам. В отличие от обычного менеджера Скрам-мастер не говорит людям что делать и не назначает им задачи - он фасилитирует [англ. facilitate, помогает чему-то случиться, прим. переводчика] процесс, поддерживают Команду в её самоорганизации и самоуправлении. Если в прошлом Скрам-мастер занимал руководящую должность в Команде, он должен в значительной степени изменить своё мышление и стиль взаимодействия, чтобы Команда была успешна в Скраме.

Замечание: В Скраме в принципе нет роли менеджера проекта. Потому что она не нужна; традиционные обязанности менеджера проекта разделены и распределены между тремя ролями в Скраме, в большей степени Команде и Владельцу Продукта, нежели Скрам-мастеру. Работа по Скраму вместе с менеджером проекта указывает на полное непонимание Скрама, что приводит к конфликту ответственности, неясным полномочиям, и неоптимальным результатам. Иногда (экс-)менеджер проекта может вступить в роль Скрам-мастера, но успех в этом случае сильно зависит от индивидуальных особенностей, и того, насколько хорошо он понимает фундаментальные отличия между двумя ролями, как в повседневных обязанностях, так и в образе мышления, необходимом для успеха. Хороший способ полностью понять роль Скрам-мастера и начать развивать основные навыки, необходимые для успеха - это посетить тренинг Сертифицированный Скрам-мастер (Certified Scrum Master, CSM) от компании Scrum Alliance.

В дополнение к этим трём ролям также существуют заинтересованные лица, которые делают вклад в успех продукта, включая руководителей, клиентов и конечных пользователей. Некоторые заинтересованные лица, такие как функциональные руководители (например, руководители инженеров), могут обнаружить, что их роль, хотя и не перестаёт быть ценной, меняется при переходе на Скрам. Например:

- они поддерживают Команду, уважая правила и дух Скрама
- они помогают убирать препятствия, которые Команда и Владелец Продукта обнаружили
- они предоставляют свой опыт и экспертизу

В Скраме люди, которые раньше тратили время, играя роль “няни” (раздача задач, подготовка статусных отчётов и другие формы микроменеджмента), могут посвятить его тому, чтобы стать “гуру” или “служителями” (servant) для Команды (обеспечивая наставничество, коучинг, помогая в устранении препятствий, в решении проблем, предлагая творческий вклад и направляя развитие навыков членов Команды). При такой перестановке менеджерам необходимо изменить их стиль руководства; например, использовать Сократовские вопросы, чтобы помочь Команде найти решение проблемы, а не решить, что делать, и передать Команде на исполнение.

Бэклог Продукта

При переходе продуктовой группы на Скрам перед стартом первого Спринта им необходим **Бэклог Продукта** (Product Backlog), приоритизированный (упорядоченный 1, 2, 3, ...) список задач, ориентированных на клиента.

Бэклог Продукта существует (и развивается) на протяжении всего жизненного цикла продукта; это дорожная карта (roadmap) продукта (**Иллюстрации 2 и 3**). В любой момент Бэклог Продукта является единственным, исчерпывающим представлением “всего, что могло бы быть сделано Командой, в порядке приоритета”. Для продукта существует только единственный Бэклог Продукта; это означает, что Владелец Продукта необходим, чтобы принимать решения о приоритетах на всём спектре, предоставляя интересы заинтересованных лиц (включая Команду).

Приоритет	Элемент	Детали (ссылка на Wiki)	Первоначальная оценка	Обновлённая Оценка в Спринте					
				1	2	3	4	5	6
1	Как покупатель, Я хочу положить книгу в корзину (см. наброски UI в wiki)	...	5						
2	Как покупатель, Я хочу удалять книги из корзины	...	2						
3	Улучшить производительность обработки транзакции (см. целевые метрики производительности в wiki)	...	13						
4	Исследовать решение для ускорения проверки кредитной карты (см. целевые метрики производительности в wiki)	...	20						
5	Обновить версию Apache до 2.2.3 на всех серверах	...	13						
6	Диагностировать и исправить ошибки в порядке исполнения скриптов (bugzilla ID 14823)	...	3						
7	Как покупатель, Я хочу создавать и сохранять список желаний	...	40						
8	Как покупатель, Я хочу добавлять и удалять элементы в в моём списке желаний	...	20						

Иллюстрация 2. Бэклог Продукта

Иллюстрация 3. Визуальное Управление: Элементы Бэклога Продукта на стене

Бэклог Продукта включает множество видов **элементов**, в первую очередь новую функциональность для пользователей ("разрешить всем пользователем класть книги в корзину"), но также и *основные* цели по инженерному улучшению (напр., "переписать код системы с C++ на Java"), улучшения (напр. "ускорить наши тесты"), исследовательскую работу ("исследовать решения для ускорения проверки кредитной карты") и, возможно, известные дефекты ("диагностировать и исправить ошибки порядка исполнения скриптов"), если только их немного (системы с большим количеством дефектов обычно имеют отдельную систему управления дефектами).

Элементы Бэклога Продукта могут быть сформулированы в любом понятном и поддерживаемом виде. Вразрез с популярным недопониманием Бэклог Продукта *не* содержит "пользовательские истории" (user stories); он просто состоит из *элементов*. Эти элементы могут быть выражены в формате пользовательских историй, сценариев использования или любом другом формате

описания требований, который группа сочтёт полезным. Но какой бы ни был выбран подход, большинство элементы должны сконцентрироваться на поставке ценности клиентом.

Хороший Бэклог Продукта является ИСЧЕРПЫВАЮЩИМ...

Подробно детализирован. Самые приоритетные элементы наиболее хорошо нормализованы по размеру, прояснены и детализированы, чем остальные с более низким приоритетом, так как они попадут в работу раньше. Например, 10% верхушки Бэклога может состоять из очень небольших, хорошо проанализированных элементов, а остальные 90% из более крупных.

Оценён. Элементы для текущего релиза должны иметь оценку, и, кроме того, следует рассматривать возможность переоценки в каждом Спринте, поскольку все учатся и появляется новая информация. Команда предоставляет Владельцу Продукта оценку своих *усилий* для каждого элемента Бэклога Продукта, также, возможно, оценивая *технические риски*. Владелец Продукта и другие заинтересованные лица предоставляют информацию о ценности продуктовых запросов, которые могут включать получение дохода, снижение издержек, бизнес риски, важность для ряда заинтересованных лиц и т.д.

Актуален. В ответ на обучение и вариативность, Бэклог Продукта постоянно проясняется. Каждый Спринт элементы могут быть добавлены, удалены, изменены, разделены, или у них может измениться приоритет. Поэтому Бэклог Продукта постоянно обновляется Владельцем Продукта, чтобы отразить изменения в потребностях клиентов, новые идеи или сведения, действия конкурентов, возникающие технические препятствия и т. д.

Приоритизирован. Элементы верхушки Бэклога Продукта приоритизированы или упорядочены в порядке 1-N. Обычно, самые высокоприоритетные элементы должны принести наибольшую *отдачу от ваших вложений* [в ориг. идиома “bang for your buck”, прим. переводчика]: больше бизнес-ценности за меньшую стоимость. Другая мотивация увеличить приоритет элемента в том, чтобы *разобраться с высоким риском раньше, чем он разберётся с вами.* [в ориг. игра слов “tackle high risks early, before the risks attack you”, прим. переводчика].

Традиционная разработка обычно не выделяет важность поставки элементов с самой высокой *отдачей от ваших вложений*, но Скрам это делает, поэтому Владельцу Продукта необходимо научиться, как определять “бизнес ценность”. Это такое обучение, в котором Скрам-мастер может помочь Владельцу Продукта. Что значит “бизнес ценность”? Некоторые продуктовые группы используют относительные очки ценности для каждого Элемента Бэклога, которые представляют собой синтез “предположительных” факторов, включая рост доходов, сокращение расходов, предпочтения заинтересованных лиц, рыночные дифференциацию и т.д. Некоторые делают вложения в конкретный элемент для одного или нескольких клиентов, платящих за его разработку, и поэтому определяют ценность на базе точного (краткосрочного) дохода от этого элемента. В других продуктовых группах такая оценка конкретных элементов слишком расфокусирована или гранулярна; они применяют более широкий поход, основанный на бизнес выгоде ("увеличить количество подписок на 10% к 1-му сентября"), в котором ценность создаётся только тогда, когда несколько элементов, способствующих результату, поставляются вместе. В этом случае Владелец Продукта должен определить следующий инкремент Минимально Жизнеспособного Продукта (Minimum Viable Product, MVP).

Оценка затрат обычно производится в относительном виде (отражающем объём работы, сложность и неопределённость) и использует в качестве единицы измерения “очки историй” (story points) или просто “очки” (points).

Это только предложение; Скрам не определяет технику для выражения или приоритизации элементов Бэклога Продукта, а также технику оценки затрат или её единицы измерения.

Обычно в Скраме применяется техника отслеживания того, сколько работы было закончено в каждом Спринте; например: в среднем выполнено работы на 26 очков за Спринт. Зная эту информацию можно прогнозировать дату релиза, к которой будут закончены все задачи, или сколько задач будет закончен к установленной дате, если в среднем это значение остаётся

неизменным. Это среднее называют “скоростью” (velocity). Скорость выражается в тех же единицах, что и оценка элементов Бэклога Продукта.

Элементы в Бэклоге Продукта могут значительным образом отличаться по размеру или затратам. Большие разбиваются на более мелкие во время Уточнения Бэклога Продукта или Планирования Спринта, а маленькие, наоборот, могут быть объединены. Элементы Бэклога Продукта для нескольких следующих Спринтов должны быть небольшими и прояснены достаточно, чтобы быть понятными Команде, тем самым давая возможность прогнозу на Планировании Спринта стать реалистичным; такой размер называется “выполнимый” (actionable).

Важные инженерные улучшения, которые требуют много времени и денег, также должны быть в Бэклогое Продукта, раз уж они могут оказаться одним из вариантов инвестиций бизнеса, и в конечном итоге решения насчёт их должны быть сделаны бизнес-ориентированным Владельцем Продукта. Обратите внимание, что в Скраме команда имеет независимые полномочия в отношении того, сколько элементов из Бэклога Продукта они решают взять в Спринт, поэтому они могут самостоятельно выполнять незначительные инженерные улучшения, поскольку их можно рассматривать, как часть обычных затрат на выполнения бизнес задач, и как то, что требуется разработчику для правильного выполнения своей работы. При этом в каждом Спринте *бóльшая часть* времени Команды обычно должна отводиться целям Владельца Продукта, а не на внутренние инженерные задачи.

Один из мифов о Скраме, что он предостерегает вас от написания исчерпывающей документации; в реальности это решение лежит на Владельце Продукта и Команде, какой уровень детализации требуется, и он может меняться от одного элемента Бэклога Продукта к другому, в зависимости от осведомлённости Команды или других факторов. Укажите, что важно, в минимально необходимом виде - другими словами, не описывайте все возможные детали элемента, просто чётко укажите, что необходимо для его понимания, и дополните это постоянным диалогом между Командой, Владельцем Продукта и заинтересованными сторонами. Элементы Бэклога Продукта с низким приоритетом, над которыми не будут работать в течение ближайшего времени, обычно являются "крупнодроблёными" (большие, с менее ясными требованиями). Высокоприоритетные и детализированные Элементы Бэклога Продукта, которые скоро будут реализованы, обычно содержат больше деталей.

Критерии Готовности

The output of every Sprint is officially called a Potentially Shippable Product Increment. Before starting the first Sprint, the Product Owner, Team, and ScrumMaster have to review what is all needed for a Product Backlog item to be potentially shippable. All activities that are needed in order to ship the product should be included in the definition of Potentially Shippable and therefore should be done during the Sprint.

Unfortunately, when Teams start using Scrum, they are often not able to achieve the goal of delivering a Potentially Shippable Increment every Sprint. This is often because the team lacks in automation or isn’t cross-functional enough (e.g. the technical writers aren’t included in the cross-functional Team yet). Over time, the Team has to improve so they will be able to deliver a Potentially Shippable Product Increment every Sprint, but in order to start, they will need to create a baseline of their existing capabilities. This is recorded in the Definition of Done.

Before the first Sprint, the Product Owner and Team need to agree on a Definition of Done, which is a subset of the activities that are needed for creating a Potentially Shippable Product Increment (for a good Team, it will be the same). The Team will plan their Sprint work according to this Definition of Done.

A good Product Owner will always want the Definition of Done to be as close as possible to Potentially Shippable as that will increase the transparency in the development and decrease *delay and risk*. If the Definition of Done is not equal to Potentially Shippable, then work is delayed until before the release which causes this *risk and delay*. This delayed work is sometimes called *undone work*.

A Scrum Team should continuously improve, which is reflected in extending their Definition of Done.

Планирование Спринта

Summary: A meeting to prepare for the Sprint, typically divided into two parts (part one is "what" and part two is "how").
Participants: Part One: Product Owner, Team, ScrumMaster. Part Two: Team, ScrumMaster, Product Owner (optional but should be reachable for questions)
Duration: Each part is timeboxed to one hour per week of Sprint.

At the beginning of each Sprint, the **Sprint Planning Meeting** takes place. It is divided into two distinct sub-meetings, the first of which is called **Sprint Planning Part One**.

In **Sprint Planning Part One**, the Product Owner and Team review the high-priority items in the Product Backlog that the Product Owner is interested in implementing this Sprint. Usually, these items will have been well-analyzed in a previous Sprint (during Product Backlog Refinement), so that at this meeting there are only minor last-minute clarifying questions. In this meeting, the Product Owner and Team discuss the goals and context for these high-priority items on the Product Backlog, providing the Team with insight into the Product Owner's thinking. Part One focuses on understanding *what* the Product Owner wants and *why* they are needed. At the end of Part One the (always busy) Product Owner may leave although they *must* be available (for example, by phone) during Part Two of the meeting.

In Part One, the Team and the Product Owner may also devise the **Sprint Goal**. This is a summary statement of the Sprint objective, which ideally has a cohesive theme. The Sprint Goal also gives the Team scope-flexibility regarding what they may actually deliver, because although they may have to remove some item (since the Sprint is timeboxed), they should nevertheless commit to delivering something tangible and "done" that is in the spirit of the Sprint Goal.

How big should the items be that are taken on in a Sprint? Each item should be split small enough so that it is estimated to require considerably less than the whole Sprint. A common guideline is that an item is estimated small enough to complete within one fourth or less of a Sprint by the whole Team.

Sprint Planning Part Two focuses on *how* to implement the items that the Team decides to take on. The Team forecasts the amount of items they can complete by the end of the Sprint, starting at the top of the Product Backlog (in others words, starting with the items that are the highest priority for the Product Owner) and working down the list in order. *This is a key practice in Scrum: The Team decides how much work it will complete, rather than having it assigned to them by the Product Owner.* This makes for a more reliable forecast because the Team is making it based on its own analysis and planning. While the Product Owner does not have control over how much the Team signs up for, he or she knows that the items are drawn from the top of the Product Backlog – in other words, the items that he or she has rated as most important. The Team has the ability to lobby for items from further down the list; this usually happens when the Team and Product Owner realize that something of lower priority fits easily and appropriately with the high priority items.

The Sprint Planning Meeting will often last several hours, but no more than four hours for a two-week Sprint – the Team is making a serious forecast to complete the work, and this requires careful thought to be successful. Part One and Part Two are of equal timeboxed lengths; for a two-week Sprint each part is two hours maximum.

Scrum does not define how to exactly do Sprint Planning Part Two. Some teams use their velocity from the previous Sprints to guide how much to aim for. Other teams will use a more fine-grained approach of first calculating their capacity.

When using the capacity approach, the Team, in Sprint Planning Part Two, calculates how much time each team member has for Sprint-related work. Most teams assume that the team members can only focus on Sprint-related work for 4-6 hours per day – the rest of the time goes to email, lunch breaks, facebook, meetings, and drinking coffee . Once the capacity is determined, the Team needs to figure out how many Product Backlog items they can complete in that time, and how they will go about completing them. This often starts with a design discussion at a whiteboard. Once the overall design is understood, the Team decomposes the Product Backlog items into fine-grained work. Before taking the Product Backlog items, the Team may focus on generating tasks for an improvement goal created

in the previous Sprint's Retrospective. Then, the Team selects the first item on the Product Backlog – the Product Owner's highest priority item – and work their way down until they are 'full'. For each item they create a list of work which consists of either decomposed Product Backlog items into tasks or, when the Product Backlog item are so small they would only take a couple hours to implement, simply the Product Backlog item. This list of work to be done during the Sprint is called the **Sprint Backlog** (Figure 4 and Figure 5).

				New Estimates of Effort Remaining at end of Day...					
Product Backlog Item	Sprint Task	Volunteer	Initial Estimate of Effort	1	2	3	4	5	6
As a buyer, I want to place a book in a shopping cart	modify database		5						
	create webpage (UI)		8						
	create webpage (Javascript logic)		13						
	write automated acceptance tests		13						
	update buyer help webpage		3						
	. . .								
Improve transaction processing performance	merge DCP code and complete layer-level tests		5						
	complete machine order for pRank		8						
	change DCP and reader to use pRank http API		13						

Figure 4. Example of one way to create a Sprint Backlog

At the end of the Sprint Planning Meeting, the Team sets a realistic target for what they believe they can deliver by the end of the Sprint. Traditionally, this was called a Sprint Commitment – the team commits to doing the best they can to reach their target. Unfortunately, this was sometimes misinterpreted as a written-in-blood promise rather than the team seriously "going for it." To avoid this confusion, the sprint-target is now called a 'forecast" which is communicated to the Product Owner.

Scrum encourages multi-skilled workers, rather than only "working to job title" such as a "tester" only doing testing. In other words, Team members "go to where the work is" and help out as possible. If there are many testing tasks, then *all* Team members may help. This does not imply that everyone is a generalist; no doubt some people are especially skilled in testing (and so on) but Team members work together and learn new skills from each other. Consequently, during task generation and estimation in Sprint Planning, it is not necessary – nor appropriate – for people to volunteer for all the tasks "they

can do best." Rather, it is better to only volunteer for one task at a time, when it is time to pick up a new task, and to consider choosing tasks that will on purpose involve learning (perhaps by pair work with a specialist). This is one reason for not pre-assigning tasks during Sprint Planning, rather this should be done on an 'as needed' basis during the Sprint.

All that said, there are *rare* times when *John* may do a particular task because it would take far too long or be impossible for others to learn – perhaps John is the only person with any artistic skill to draw pictures. Other Team members could not draw a "stick man" if their life depended on it. In this rare case – and if it is not rare and not getting rarer as the Team learns, there is something wrong – it may be necessary to ask if the total planned drawing tasks that *must* be done by John are feasible within the short Sprint.

Many Teams have a Sprint Backlog in the form of a wall-sized task board (often called a **Scrum Board**) where tasks (written on Post-It Notes) migrate during the Sprint across columns labeled "To Do," "Work In Progress," and "Done." See Figure 5.

Figure 5. Visual Management - Sprint Backlog tasks on the wall

One of the pillars of Scrum is that once the Team sets its target for the Sprint, any additions or changes must be deferred until the next Sprint. This means that if halfway through the Sprint the Product Owner decides there is a new item he or she would like the Team to work on, he cannot make the change until the start of the next Sprint. If an external circumstance appears that significantly changes priorities, and means the Team would be wasting its time if it continued working, the Product Owner or the Team can terminate the Sprint. The Team stops, and a new Sprint Planning meeting initiates a new Sprint. The disruption of doing this is usually great; this serves as a disincentive for the Product Owner or Team to resort to this dramatic decision.

There is a powerful, positive influence that comes from the Team being protected from changing goals

during the Sprint. First, the Team gets to work knowing with absolute certainty that its goal will not change, that reinforces the Team's focus on ensuring completion. Second, it disciplines the Product

Owner into really thinking through the items he or she prioritizes on the Product Backlog and offers to the Team for the Sprint.

By following these Scrum rules the Product Owner gains two things. First, he or she has the confidence of knowing the Team has committed to do its best to complete a realistic and clear set of work it has chosen. Over time a Team can become quite skilled at choosing and delivering on a realistic forecast. Second, the Product Owner gets to make whatever changes he or she likes to the Product Backlog before the start of the *next* Sprint. At that point, additions, deletions, modifications, and re-prioritizations are all possible and acceptable. While the Product Owner is not able to make changes to the selected items under development during the current Sprint, he or she is only one Sprint's duration or less away from making any changes they wish. Gone is the stigma around change – change of direction, change of requirements, or just plain changing your mind – and it may be for this reason that Product Owners are usually as enthusiastic about Scrum as anyone.

Ежедневный Скрам

Summary: Update and coordination between the Team members.
Participants: Team is required; Product Owner is optional; ScrumMaster is usually present but ensures Team holds one.
Duration: Maximum length of 15 minutes.

Once the Sprint has started, the Team engages in another of the key Scrum practices: The **Daily Scrum**. This is a short (15 minutes or less) meeting that happens every workday at an appointed time. Everyone on the Team attends. To keep it brief, it is recommended that everyone remain standing. It is the Team's opportunity to synchronize their work and report to each other on obstacles. In the Daily Scrum, one by one, each member of the Team reports three things *to the other members of the Team*: (1) What has been accomplished since the last meeting?; (2) What will be done before the next meeting?; and (3) What obstacles are in the way?. Note that the Daily Scrum is not a status meeting to report to a manager; it is a time for a self-organizing Team to share with each other what is going on, to help them coordinate. Someone makes note of the blocks, and the ScrumMaster is responsible to help Team members resolve them. There is little or no in-depth discussion during the Daily Scrum, the theme is *reporting* answers to the three questions; if discussion is required it takes place immediately after the Daily Scrum in one or more parallel follow-up meetings, although in Scrum no one is required to attend these. A follow-up meeting is a common event where some or all team members adapt to the information they heard in the Daily Scrum: in other words, another inspect and adapt cycle. For Teams new to Scrum, it is generally recommended *not* to have managers or others in positions of perceived authority attend the Daily Scrum. This risks making the Team feel "monitored" – under pressure to report major progress every day (an unrealistic expectation), and inhibited about reporting problems – and it tends to undermine the Team's self-management, and invite micromanagement. It would be more useful for a stakeholder to instead reach out to the Team following the meeting, and offer to help with any blocks that are slowing the Team's progress.

Tracking Progress during the Sprint

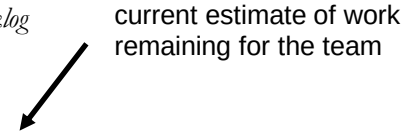
The Team in Scrum is self-managing, and in order to do this successfully, it must know how it is doing. Every day, the Team members update their estimate of the effort remaining to complete their current work in the **Sprint Backlog** (Figure 6). It is also common for someone to add up the effort remaining for the Team as a whole, and plot it on the **Sprint Burndown Chart** (Figure 7 and Figure 8). This graph shows, each day, a new estimate of how much work remains until the Team is finished. Ideally, this is a *downward* sloping graph that is on a trajectory to reach "zero effort remaining" by the last day of the Sprint. Hence it is called a *burndown* chart. And while sometimes it looks good, often it does not; this is the reality of product development. The important thing is that it shows the Team their progress towards their goal, not in terms of how much time was *spent* in the past (an irrelevant fact in terms of *progress*), but in terms of how much work *remains in the future* – what separates the Team from their goal. If the burndown line is not tracking downwards towards completion near the end of the Sprint, then the Team

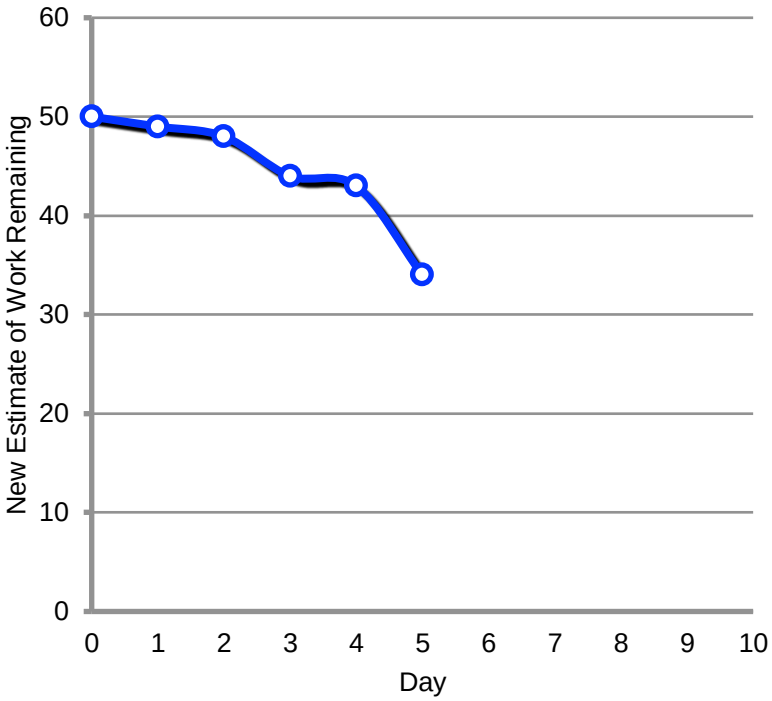
needs to adjust, such as to reduce the scope of the work or to find a way to work more effectively while still maintaining a sustainable pace.

While the Sprint Burndown chart can be created and displayed using a spreadsheet, many Teams find it is more effective to show it on paper on a wall in their workspace, with updates in pen; this "low-tech/high-touch" solution is fast, simple, and often more visible than a computer chart.

				New Estimates of Effort Remaining at end of Day. ..					
Product Backlog Item	Sprint Task	Volunteer	Initial Estimate of Effort	1	2	3	4	5	6
As a buyer, I want to place a book in a shopping cart	modify database	Sanjay	5	4	3	0	0	0	
	create webpage (UI)	Jing	3	3	3	2	0	0	
	create webpage (Javascript logic)	Tracy & Sam	2	2	2	2	1	0	
	write automated acceptance tests	Sarah	5	5	5	5	5	0	
	update buyer help webpage	Sanjay & Jing	3	3	3	3	3	0	
	. . .								
Improve transaction processing performance	merge DCP code and complete layer-level tests		5	5	5	5	5	5	
	complete machine order for pRank		3	3	8	8	8	8	
	change DCP and reader to use pRank http API		5	5	5	5	5	5	
	. . .	. . .	. . .						
		Total	50	49	48	44	43	34	

Figure 6. Daily Updates of Work Remaining on the Sprint Backlog





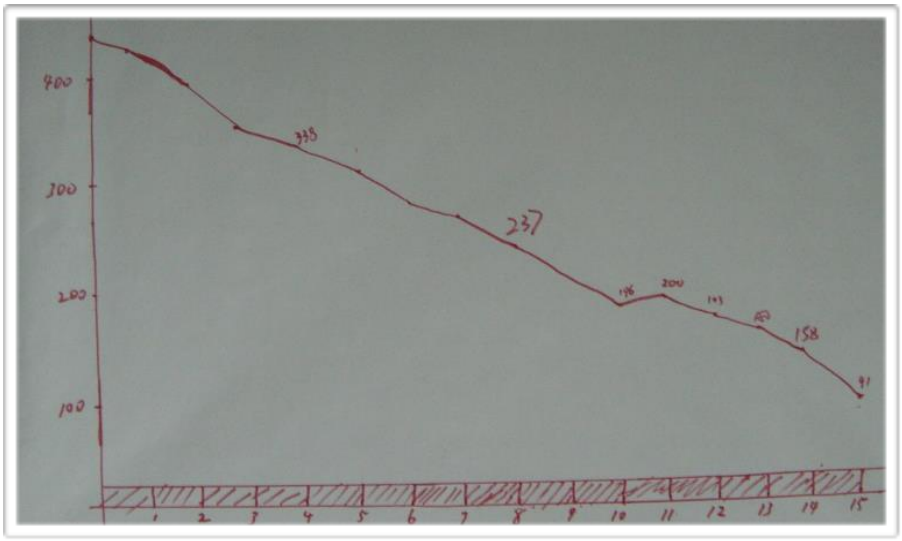


Figure 7. Sprint Burndown Chart

Figure 8. Visual Management: Hand-drawn Sprint Burndown Chart

Уточнение Бэклога Продукта

Summary: Split big items, analyze items, re-estimate, and re-prioritize, for *future* Sprints.
Participants: Team; Product Owner will attend the entire activity if they are the expert who can help with the detailed refinement, otherwise they may attend only a subset to set direction or re-prioritize; others who understand the requirements and can help the Team; ScrumMaster will attend during initial sessions to coach the group to be effective, otherwise may not attend.
Duration: Usually, no more than 10% of the capacity of the Team for the Sprint, though it may be longer for "analysis heavy" items. For example, in a two-week Sprint, perhaps one day is spent on refinement.

One of the lesser known, but valuable, guidelines in Scrum is that some percentage of each Sprint should be dedicated by the whole Team to refining (or "grooming") the Product Backlog to support future Sprints. This includes detailed requirements analysis, splitting large items into smaller ones, estimation of new items, and re-estimation of existing items. Scrum is silent on how this work is done, but a frequently used technique is a focused workshop near the middle or end of the Sprint, so that the Team and Product Owner and other stakeholders can dedicate themselves to this work without interruption.

This refinement activity is *not* for items selected for the current Sprint; it is for items for the future, most likely in the next one or two Sprints. With this practice, Sprint Planning becomes relatively simple because the Product Owner and Scrum Team start the planning with a clear, well-analyzed and carefully estimated set of items. A sign that this refinement workshop is not being done (or not being done well) is that Sprint Planning involves significant questions, discovery, or confusion and feels incomplete; planning work then often spills over into the Sprint itself, which is typically not desirable.

Обзор Спринта

Summary: Inspection and adaption related to the product increment of functionality.
Participants: Team, Product Owner, ScrumMaster. Other stakeholders as appropriate, invited by the Product Owner.
Duration: Timeboxed to one hour per week of Sprint.

After the Sprint ends, there is the **Sprint Review**, where people review the Sprint. Present at this meeting are the Product Owner, Team members, and ScrumMaster, plus customers, users, stakeholders, experts, executives, and anyone else who is interested. For a two-week Sprint it is a maximum length of two hours. Anyone present is free to ask questions and give input.

The Review is often mislabeled the "demo" but that does not capture the real intent of this meeting. A key idea in Scrum is *inspect and adapt*. To see and learn what is going on and then evolve based on feedback, in repeating cycles. The Sprint Review is an inspect and adapt activity for the *product*. It is a time for the Product Owner to learn what is going on with the product and with the Team (that is, a review of the Sprint); and for the Team to learn what is going on with the Product Owner and the market. Consequently, a critical element of the Review is an in-depth *conversation* between the Team and Product Owner to learn the situation, to get advice, and so forth. The review definitely includes using the actual live software that the Team built during the Sprint, but if the focus of the review is only looking at the product rather than having a conversation, there is an imbalance.

The "live software" portion of the Sprint Review is not a "presentation" the Team gives – there is no slideware. It is meant to be a hands-on inspection of the real software running live, for example, in a sandbox development environment. There will be one or more computers in the Review room on which people can inspect and use the live software. Prefer an active session in which real users and the Product Owner do hands-on interaction with the software, rather than a passive-session demo from the Team.

Aim to spend no more than 30 minutes preparing for Sprint Review, otherwise it suggests something is wrong.

Ретроспектива Спринта

Summary: Inspection and adaption related to the process and environment.
Participants: Team, ScrumMaster, Product Owner (optional). Other stakeholders may be invited by the Team, but are not otherwise allowed to attend.
Duration: Timeboxed to 45 minutes per week of Sprint.

The Sprint Review involves inspect and adapt regarding the *product*. The **Sprint Retrospective**, which follows the Review, involves inspect and adapt regarding the *process and environment*. It's an opportunity for the Team to discuss what's working and what's not working, and agree on changes to try. Sometimes

the ScrumMaster can act as an effective facilitator for the Retrospective, but it may be better to find a neutral outsider to facilitate the meeting; a good approach is for ScrumMasters to facilitate each others' retrospectives, which enables cross-pollination among Teams.

There are many techniques for conducting a Sprint Retrospective, and the book *Agile Retrospectives* (Derby, Larsen 2006) provides a useful catalogue of techniques.

Many teams hold retrospectives only focusing on *problems*, and that's too bad. It can lead to people thinking of retrospectives as somewhat depressing or negative events. Instead, ensure that every Retrospective also focus on positives or strengths; there are several books on *appreciative inquiry* that offer more detailed tips.

Retrospectives that always use the same technique of analysis may become boring; therefore, introduce various techniques over time.

Начала Следующего Спринта

Following the Sprint Review, the Product Owner may update the Product Backlog with any new insight –adding new Items, removing obsolete ones, or revising existing ones. The Product Owner is responsible for ensuring that these changes are reflected in the Product Backlog. See Figure 9 for an example of the updated Product Backlog.

Приоритет	Элемент	Детали (ссылка на Wiki)	Первоначальная оценка	Обновлённая Оценка в Спринте					
				1	2	3	4	5	6
1	Как покупатель, Я хочу положить книгу в корзину (см. наброски UI в wiki)	...	5	0	0	0			
2	Как покупатель, Я хочу удалять книги из корзины	...	2	0	0	0			
3	Улучшить производительность обработки транзакции (см. целевые метрики производительности в wiki)	...	13	13	0	0			
4	Исследовать решение для ускорения проверки кредитной карты (см. целевые метрики производительности в wiki)	...	20	20	20	0			
5	Обновить версию Apache до 2.2.3 на всех серверах	...	13	13	13	13			
6	Диагностировать и исправить ошибки в порядке исполнения скриптов (bugzilla ID 14823)	...	3	3	3	3			
7	Как покупатель, Я хочу создавать и сохранять список желаний	...	40	40	40	40			
8	Как покупатель, Я хочу добавлять и удалять элементы в моём списке желаний	...	20	20	20	20			
	...	...	...	...	...	...			
			537	**580**	**570**	**500**			

Figure 9. Updated Product Backlog

There is no down time between Sprints – Teams normally go from a Sprint Retrospective one afternoon into the next Sprint Planning the following morning (or after the weekend).

One of the principles of agile development is "sustainable pace", and only by working regular hours at a reasonable level can Teams continue this cycle indefinitely. Productivity grows over time through the evolution of the Team's practices, and the removal of impediments to the Team's productivity, not through overwork or the compromise of quality.

Sprints continue until the Product Owner decides the product is ready for release. The perfection vision of Scrum is that the product is potentially shippable at the end of each Sprint, which implies there is no wrap up work required, such as testing or documentation. The implication is that *everything* is completely *finished* every Sprint; that you could actually ship it or deploy it immediately after the Sprint Review. However, many organizations have weak development practices, tools and infrastructure and cannot achieve this perfection vision and so there will be the need for a "Release Sprint" to handle this remaining

work. When a "Release Sprint" is needed, it is considered necessary evil and the organization's job is to improve their practices so this is not needed anymore.

Управление Релизами

A question that is sometimes asked is how, in an iterative model, can long-term release planning be done. There are two cases to consider: (1) a new product in its first release, and (2) an existing product in a later release.

In the case of a new product, or *an existing product just adopting Scrum*, there is the need to do initial Product Backlog refinement before the first Sprint, where the Product Owner and Team shape a proper Scrum Product Backlog. This could take a few days or a week, and involves a workshop (sometimes called Initial Product Backlog Creation or Release Planning), some detailed requirements analysis, and estimation of all the items identified for the first release.

Surprisingly in Scrum, in the case of an established product with an established Product Backlog, there should not be the need for any special or extensive release planning for the next release. Why? Because the Product Owner and Team should be doing Product Backlog refinement every Sprint (five or ten percent of each Sprint), continuously preparing for the future. This *continuous product development* mode obviates the need for the dramatic punctuated prepare-execute-conclude stages one sees in traditional sequential life cycle development.

During an initial Product Backlog refinement workshop and during the continuous backlog refinement each Sprint, the Team and Product Owner will do release planning, refining the estimates, priorities, and content as they learn.

Some releases are date-driven; for example: "We will release version 2.0 of our project at a trade-show on November 10." In this situation, the Team will complete as many Sprints (and build as many features) as is possible in the time available. Other products require certain features to be built before they can be called complete and the product will not launch until these requirements are satisfied, however long that takes. Since Scrum emphasizes producing potentially shippable code each Sprint, the Product Owner may choose to start doing interim releases, to allow the customer to reap the benefits of completed work sooner.

Since they cannot possibly know everything up front, the focus is on creating and refining a plan to give the release broad direction, and clarify how tradeoff decisions will be made (scope versus schedule, for example). Think of this as the roadmap guiding you towards your final destination; which exact roads you take and the decisions you make during the journey may be determined en route.

The destination is more important than the journey.

Most Product Owners choose one release approach. For example, they will decide a release date, and will work with the Team to estimate the Product Backlog items that can be completed by that date. The items that are anticipated to be in the current release are sometimes called the *release items*. In situations where a "fixed price / fixed date / fixed deliverable" commitment is required – for example, contract development – one or more of those parameters must have a built-in buffer to allow for uncertainty and change; in this respect, Scrum is no different from other approaches.

Фокус на Продукте или Приложении

For applications or products – either for the market or for internal use within an organization – Scrum moves groups away from the older *project*-centric model toward a *continuous application/product development* model. There is no longer a project with a beginning, middle, and end. And hence, no traditional project manager. Rather, there is simply a stable Product Owner and a long-lived self-managing Team that collaborate in an "endless" series of fixed-length Sprints, until the product or application is retired. All necessary "project" management work is handled by the Team and the Product Owner – who is an internal business customer or from Product Management. It is not managed by an IT manager or someone from a Project Management Office.

Scrum can also be used for true *projects* that are one-time initiatives (rather than work to create or evolve long-lived applications); still, in this case the Team and Product Owner do the project management.

What if there is insufficient new work from one or more existing applications to warrant a dedicated long-lived Team for each application? In this case, a stable long-lived Team may take on items from one application in one Sprint, and then items from another in the next Sprint; in this situation the Sprints are often quite short, such as one week.

Occasionally, there is insufficient new work even for the prior solution, and the Team may take on items from *several* applications during the same Sprint; however, beware this solution as it may devolve into unproductive multitasking across multiple applications. A basic productivity theme in Scrum is for the Team to be *focused* on one product or application for one Sprint.

Основные Трудности

Scrum is not only a concrete set of practices – rather, and more importantly, it is a framework that provides transparency, and a mechanism that allows "inspect and adapt". Scrum works by making visible the dysfunction and impediments that are impacting the Product Owner and the Team's effectiveness, so that they can be addressed. For example, the Product Owner may not really know the market, the features, or how to estimate their relative business value. Or the Team may be unskillful in effort estimation or development work.

The Scrum framework will quickly reveal these weaknesses. Scrum does not solve the problems of development; it makes them painfully visible, and provides a framework for people to explore ways to resolve problems in short cycles and with small improvement experiments.

Suppose the Team fails to deliver what they forecast in the first Sprint due to poor task analysis and estimation skill. To the Team, this feels like failure. But in reality, this experience is the necessary first step toward becoming more realistic and thoughtful about its forecasts. This pattern – of Scrum helping make visible dysfunction, enabling the Team to do something about it – is the basic mechanism that produces the most significant benefits that Teams using Scrum experience.

One common mistake made, when presented with a Scrum practice that is challenging, is to change Scrum. For example, Teams that have trouble delivering might decide to make the Sprint duration extendable, so it never runs out of time – and in the process, ensure it never has to learn how to do a better job of estimating and managing its time. In this way, without coaching and the support of an experienced ScrumMaster, organizations can mutate Scrum into just a mirror image of their own weaknesses and dysfunction, and undermine the real benefit that Scrum offers: Making visible the good and the bad, and giving the organization the choice of elevating itself to a higher level.

Another common mistake is to assume that a practice is discouraged or prohibited just because Scrum does not specifically require it. For example, Scrum does not require the Product Owner to set a long-term strategy for his or her product; nor does it require engineers to seek advice from more experienced engineers about complex technical problems. Scrum leaves it to the individuals involved to make the right decision; and in most cases, both of these practices (along with many others) are well advised.

Something else to be wary of is managers imposing Scrum on their Teams; Scrum is about giving a Team space and tools to manage itself, and having this dictated from above is not a recipe for success. A better approach might begin with a Team learning about Scrum from a peer or manager, getting comprehensively educated in professional training, and then making a decision as a Team to follow the practices faithfully for a defined period; at the end of that period, the Team will evaluate its experience, and decide whether to continue.

The good news is that while the first Sprint is usually very challenging to the Team, the benefits of Scrum tend to be visible by the end of it, leading many new Scrum Teams to exclaim: "Scrum is hard, but it sure is a whole lot better than what we were doing before!"

Приложение A: Дополнительные материалы

There is a lot of material published about Scrum. In this reference section, we would like to point out some additional online material and a couple of books.

Online material:

- The Lean Primer - An introduction to Lean Thinking, an important influence to Scrum. http://www.leanprimer.com
- The Distributed Scrum Primer - Additional tips for teams who aren't co-located. http://www.goodagile.com/distributedscrumprimer/
- The ScrumMaster Checklist - A list of question that good ScrumMasters use. http://www.scrummasterchecklist.org/
- Feature Team Primer - Scaling Scrum with Feature Teams, http://www.featureteams.org
- The Agile Atlas - Core Scrum. ScrumAlliance description of Scrum. http://agileatlas.org/atlas/scrum
- Scrum Guide - Scrum.org description of Scrum. http://www.scrum.org/Scrum-Guides
- Agile Contracts Primer - How to make Scrum-friendly contracts. http://www.agilecontracts.org/

Books:

- Leading Teams - Richard Hackman
- Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum - Craig Larman, Bas Vodde
- Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum - Craig Larman, Bas Vodde
- Agile Project Management with Scrum - Ken Schwaber
- Succeeding with Agile: Software Development using Scrum - Mike Cohn

Приложение Б: Определения

Burn Down
The trend of work remaining across time in a Sprint, a Release, or a Product. The source of the raw data is the Sprint Backlog and the Product Backlog, with work remaining tracked on the vertical axis and the time periods (days of a Sprint, or Sprints) tracked on the horizontal axis.

Daily Scrum
A short meeting held daily by each Team during which the Team members inspect their work, synchronize their work and progress and report and impediments to the ScrumMaster for removal. Follow-on meetings to adapt upcoming work to optimize the Sprint may occur after the Daily Scrum meetings.

Development Team
Another name for the Team role.

Done
Complete as mutually agreed to by all parties and that conforms to an organization's standards, conventions, and guidelines. When something is reported as "done" at the Sprint Review meeting, it must conform to this agreed definition.

Estimated Work Remaining (Sprint Backlog items)
The number of hours that a Team member estimates remain to be worked on any task. This estimate is updated at the end of every day when the Sprint Backlog task is worked on. The estimate is the total estimated effort remaining, regardless of the number of people that perform the work.

Increment
Product functionality that is developed by the Team during each Sprint that is potentially shippable or of use to the Product Owner's stakeholders.

Increment of Potentially Shippable Product Functionality
A complete slice of the overall product or system that could be used by the Product Owner or stakeholders if they chose to implement it.

Sprint
An iteration, or one repeating cycle of similar work, that produces increment of product or system. No longer than one month and usually more than one week. The duration is fixed throughout the overall work and all teams working on the same system or product use the same length cycle.

Product Backlog
A prioritized list of requirements with estimated times to turn them into completed product functionality. Estimates are more precise the higher an item is in the Product Backlog priority.. The list emerges, changing as business conditions or technology changes.

Product Backlog Item
Functional requirements, non-functional requirements, and issues, prioritized in order of importance to the business and dependencies, and estimated. The precision of the estimate depends on the priority and granularity of the Product Backlog item, with the highest priority items that may be selected in the next Sprint being very granular and precise.

Product Owner
The person responsible for managing the Product Backlog so as to maximize the value of the product. The Product Owner is responsible for representing the interests of everyone with a stake in the project and its resulting product.

Scrum
Not an acronym, but mechanisms in the game of rugby for getting an out-of-play ball back into play.

ScrumMaster
The person responsible for the Scrum process, its correct implementation, and the maximization of its benefits.

Sprint Backlog
A list of the Team's work for a Sprint. This is often decomposed into a set of more detailed tasks. The list emerges during Sprint Planning and may be updated by the team during the Sprint with items being removed or new tasks being added as needed. Each Sprint Backlog task will be tracked during the Sprint and will show the estimated effort remaining.

Sprint Backlog Task
One of the tasks that the Team or a Team member defines as required to turn committed Product Backlog items into system functionality.

Sprint Planning meeting
A meeting time boxed to four hours (for a two week Sprint) that initiates every Sprint. The meeting is divided into two two-hour segments, each also time boxed. During the first part the Product Owner presents the highest priority Product Backlog to the team. The Team and Product Owner collaborate to help the Team determine how much Product Backlog it can turn into functionality during the upcoming Sprint. During the second part, the Team plans how it will achieve this by designing and decomposing the work so they understand how they will meet the Sprint Goal.

Sprint Retrospective meeting
A meeting facilitated by the ScrumMaster at which the complete Team discusses the just-concluded Sprint and determines what could be changed that might make the next Sprint more enjoyable or productive.

Sprint Review meeting
A time-boxed two hour meeting (for a two week Sprint) at the end of every Sprint where the Team collaborates with the Product Owner and stakeholders and they inspect the output from the Sprint. This usually starts with a review of completed Product Backlog items, a discussion of opportunities, constraints and risks, and a discussion of what might be the best things to do next (potentially resulting in Product Backlog changes). Only completed product functionality can be demonstrated.

Stakeholder
Someone with an interest in the outcome of a project, either because they have funded it, will use it, or will be affected by it.

Team
A cross-functional group of people that is responsible for managing themselves to develop an increment of product every Sprint.

Time box
A period of time that cannot be exceeded and within which an event or meeting occurs. For example, a Daily Scrum meeting is time boxed at fifteen minutes and terminates at the end of fifteen minutes, regardless. For meetings, it might last shorter. For Sprints, it lasts exactly that length.