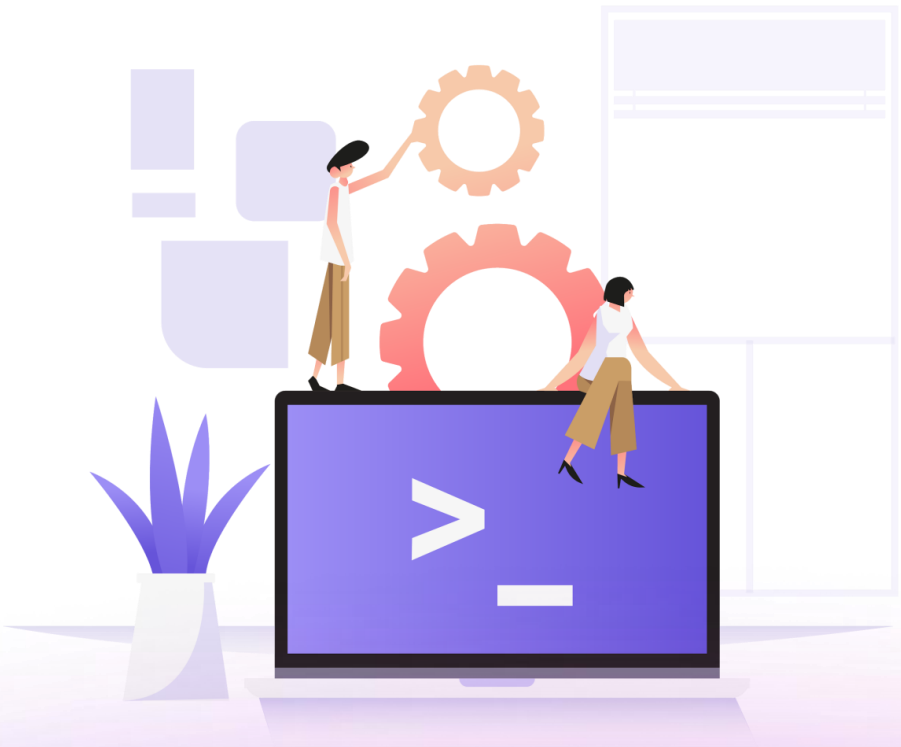
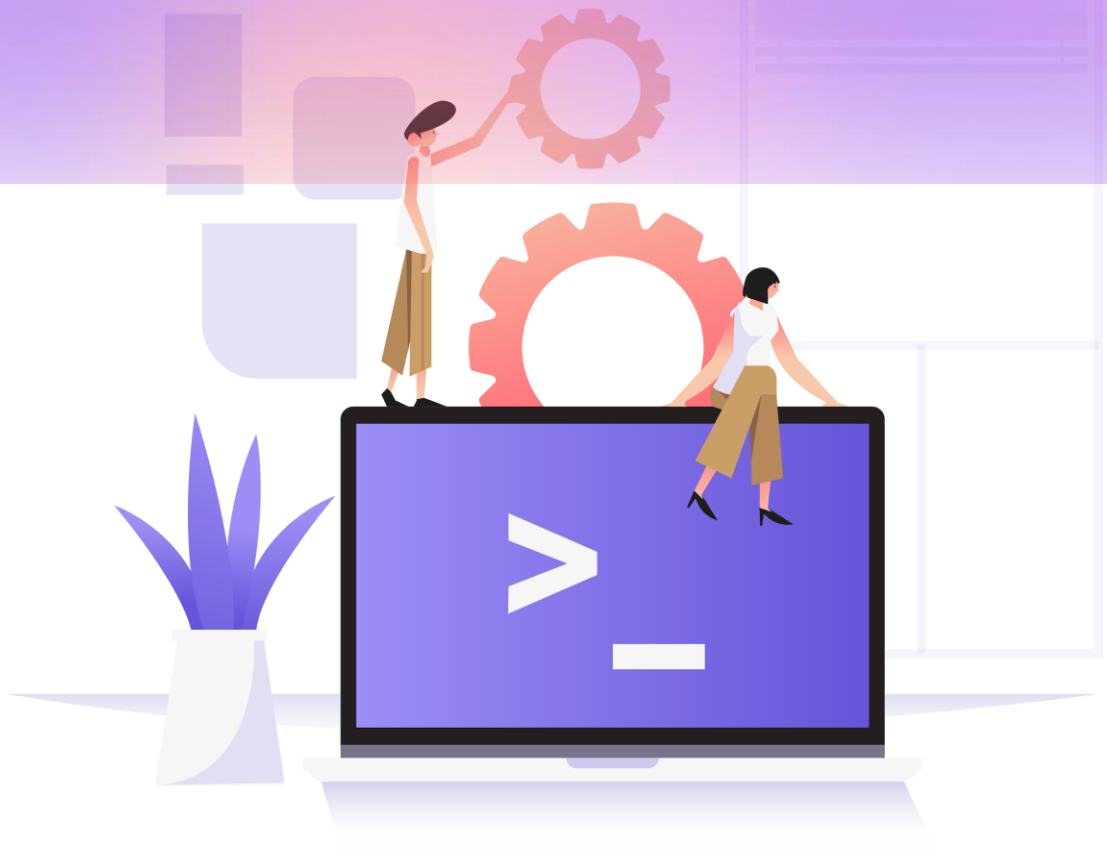
GeekBrains

Командная разработка с использованием Agile подхода и фреймворка Scrum.

Подготовка к старту проекта





На этом уроке

1. Определим цель проекта и объём работ.
2. Оформим бэклог продукта и подготовим его к планированию спринта.
3. Проведём планирование спринта: составим бэклог спринта, подготовим задачи к работе.
4. Оформим доску визуализации в Trello.

Оглавление

Введение

На прошлом уроке мы подробно разобрали роли участников Scrum-команды и рассмотрели основные события, посредством которых организуется весь рабочий процесс в рамках фреймворка.

На завершающем уроке нашей теоретической части мы начнём готовиться к старту проекта. Научимся работать с бэклогом продукта и подробно разберём этап планирования спринта: подготовку задач к работе и оформление доски визуализации в Trello.

Цель проекта и объём работ

Чтобы увеличить вероятность успешной реализации того или иного проекта, необходимо поставить конкретную цель. Под целью проекта обычно понимают общее описание предполагаемых результатов и ожиданий, к которым стремится команда в ходе реализации проекта.

Цель формулируется в результате процесса, который называется целеполаганием.

Самый простой способ сформулировать цель — это описать решение проблемы, которая стала поводом для инициации проекта. В результате проекта и достижения его цели проблема должна быть решена полностью или частично. Однако это не самый надёжный способ, так как он может не учитывать ряд необходимых критериев, например, что у проблемы может быть несколько решений.

Один из самых популярных способов, который используется в целеполагании, — метод SMART.

SMART-метод

SMART — мнемоническая аббревиатура, используемая для определения целей и постановки задач в проектном менеджменте, управлении производством и личном развитии.

Впервые термин SMART употребил в 1965 году П. Мейер, который изучал проблематику эффективного менеджмента, однако известная сегодня расшифровка с отдельным смыслом каждого слова описана Дж. Дораном в статье There’s a S.M.A.R.T. way to write management’s goals and objectives из журнала Management Review за 1981 год. Опираясь на русский перевод названия статьи («Вот умный путь написать управленческие цели и задачи»), можно сделать вывод, что в английском языке слово smart используется ещё и в значении «умный». Отсюда и требование к процессу целеполагания, который должен быть умным, и синонимичные формы употребления: SMART-цели, умные цели.

Постановка целей по SMART приобрела значительную популярность в менеджменте, что способствовало развитию техники в целом и закреплению смысла, вкладываемого в каждый отдельный элемент.

Рассмотрим современные трактовки более подробно, чтобы понять, по какому принципу формируются SMART-цели.

Рисунок 1. SMART — расшифровка аббревиатуры

Specific — конкретная, определённая. Отвечает на вопрос «Что нужно сделать?» В процессе целеполагания конкретизируются результаты, к которым нужно прийти. Формулировка цели должна быть чёткой, исчерпывающей, чтобы любой участник проекта правильно понимал её суть.

Measurable — измеримая. Отвечает на вопрос «Как определить, что цель достигнута?» В процессе целеполагания определяются целевые показатели и выбираются критерии для оценки. Цель должна приводить к измеримому результату — либо количественно, либо качественно. В первом случае фиксируется цифра, которую нужно достичь. Например, для клиента это чаще всего коммерческий успех: прирост выручки или повышение веб-трафика. Во втором случае приводится эталон. Например, для команды это соответствие результата работы критериям приёмки.

Achievable — достижимая. Отвечает на вопрос «Может ли цель быть достигнута с учётом имеющихся в распоряжении проекта ресурсов?» В ходе целеполагания оценивается и прогнозируется степень достижимости результата. Под ресурсами в данном случае подразумеваются бюджет, время, состав команды, оборудование и всё, что может понадобиться для достижения цели.

Relevant — релевантность, целесообразность, актуальность. Отвечает на вопрос «Эта цель актуальна и соответствует нашим потребностям?» В ходе целеполагания обосновывается необходимость и релевантность целей. Цель должна соответствовать потребностям и отвечать актуальным требованиям к технологиям и качеству на момент достижения.

Timed/Time-bounded — привязана к временному интервалу. Отвечает на вопрос «Когда мы сможем достичь этой цели?» В процессе целеполагания определяются сроки достижения полностью сформированной цели. Достижение цели должно быть ограничено во времени, иначе она потеряет свою релевантность и актуальность. Подобное ограничение положительно влияет на мотивационную составляющую, а также облегчает контроль над выполнением задач.

От того, насколько корректно сформулирована цель проекта, напрямую зависит объём работ. Именно поэтому, когда команда разработки начинает обсуждать требования к результату проекта, необходимо понимать истинную цель проекта, которую ставит перед собой клиент.

Например, в случае работы с веб-продуктом для клиента целью проекта будет не создание сайта или мобильного приложения, а коммерческая выгода от дальнейшего использования этого продукта.

Команде при работе над продуктом следует всегда помнить, что ценность продукта в первую очередь должна выражаться в выгодах, которые он даёт клиенту.

Agile-концепция и, в частности, фреймворк Scrum позволяют произвести дифференциацию потребностей клиента, выбрав самые ценные функции к ближайшей поставке, а затем с каждой новой итерацией наращивать инкремент. Степень ценности функции зависит от соотношения стоимости затрат на её реализацию к выгодам, которые она принесёт после запуска.

Чаще всего для первого релиза Agile-команды готовят продукт с минимальным набором основных функций, который, возможно, далёк от идеала, но уже позволяет получать выгоду от его использования.

MVP

MVP (Minimum Viable Product) — минимально жизнеспособный продукт. Это завершённый, готовый к запуску продукт с базовым функционалом, которого достаточно, чтобы удовлетворить основные потребности клиента или конечного пользователя. MVP находится на пересечении областей минимального сырого продукта и жизнеспособного идеального продукта, который хочет получить клиент.

Рисунок 2. Область образования MVP

Термин был придуман и определён Фрэнком Робинсоном около 2001 года и популяризирован Стивом Бланком и Эриком Рисом — известными предпринимателями и признанными специалистами в области стартапов.

Область применения MVP — стартапы и продуктовый менеджмент. В обоих случаях он необходим для отработки гипотез и получения обратной связи от конечного пользователя или клиента. Это позволяет постепенно совершенствовать продукт, повысить качество разработки и сократить риск того, что финальная версия продукта не будет соответствовать ожиданиям клиента и потребностям рынка.

Рисунок 3. Пример MVP с последующим наращиванием функционала

Создание MVP полностью соответствует основным принципам итеративно-инкрементальной разработки, которую использует Agile: быстрая поставка продукта с целью получения обратной связи и дальнейшее его усовершенствование за счёт внесения изменений и наращивания функциональности в следующих итерациях.

Чтобы определить, какой объём работ войдёт в создание MVP, необходимо проработать все требования к продукту, которые поступили от клиента, и выделить из них самые ценные путём приоритизации. Scrum использует для этого бэклог продукта.

Бэклог продукта и работа с ним

Бэклог продукта (Product Backlog) — это упорядоченный список всего, что может понадобиться в продукте. Перечень требований и функций, поступивший от заказчика.

Исходя из определения, очевидно, что бэклог продукта формируется на основе данных, полученных от заказчика, единый представитель которого по фреймворку — это владелец продукта. Требования могут поступить в формализованной форме в виде ТЗ или могут быть выявлены командой в ходе интервью. Независимо от того, каким способом поступили требования, их необходимо занести в бэклог продукта в виде списка пользовательских историй.

Пользовательские истории (англ. User Story) — способ описания требований к разрабатываемой системе или продукту, сформулированных как одно или более предложений на повседневном или деловом языке пользователя. Пользовательские истории — это элементы бэклога, они пишутся простым языком, в общих формулировках, без технической специфики и служат контекстом для команды разработчиков и их деятельности. Истории короткие. Они представляют маленькие кусочки деловой ценности, которые можно реализовать в период от нескольких дней до нескольких недель.

Правила написания пользовательских историй:

1. Составляется от лица пользователя: *как <роль>*.
2. Описывает функциональную возможность ПО: *я хочу/нуждаюсь/могу/<функция>*.
3. Поясняет, какую пользу принесёт функция: *для того, чтобы <цель>*.

Пример из работы с мобильным приложением «Служба доставки продуктов»: *как покупатель, я хочу иметь свой личный кабинет для того, чтобы управлять своими заказами*.

Это клиентоориентированный подход к формулировке требований, который способствует тому, чтобы команда говорила на одном языке с заказчиком и не забывала, что главное — это ценность, которую даёт реализация конкретного функционала для той или иной роли.

По правилам Scrum за формирование бэклога продукта отвечает владелец продукта, однако он может делегировать эти полномочия команде или попросить у неё помощи с правильным составлением списка.

Первый вариант бэклога может быть неструктурированным и включать в себя только изначально известные и наиболее понятные требования. Необходимо помнить, что список никогда не бывает исчерпывающим и по мере продвижения проекта и изменений, которые могут произойти в процессе, видоизменится и сам бэклог продукта. В процессе работы туда могут попадать не только новые требования к функционалу, но и пользовательские истории, связанные с обновлением технологий, интеграцией, исправлением багов, улучшением производительности, оптимизацией пользовательского интерфейса и т. д. Такая динамика позволяет продукту оставаться актуальным, конкурентоспособным и приносящим максимальную пользу.

Уточнение бэклога продукта

Уточнение бэклога продукта — это совместная деятельность владельца продукта и команды разработки по уточнению, оценке и приоритизации элементов бэклога продукта.

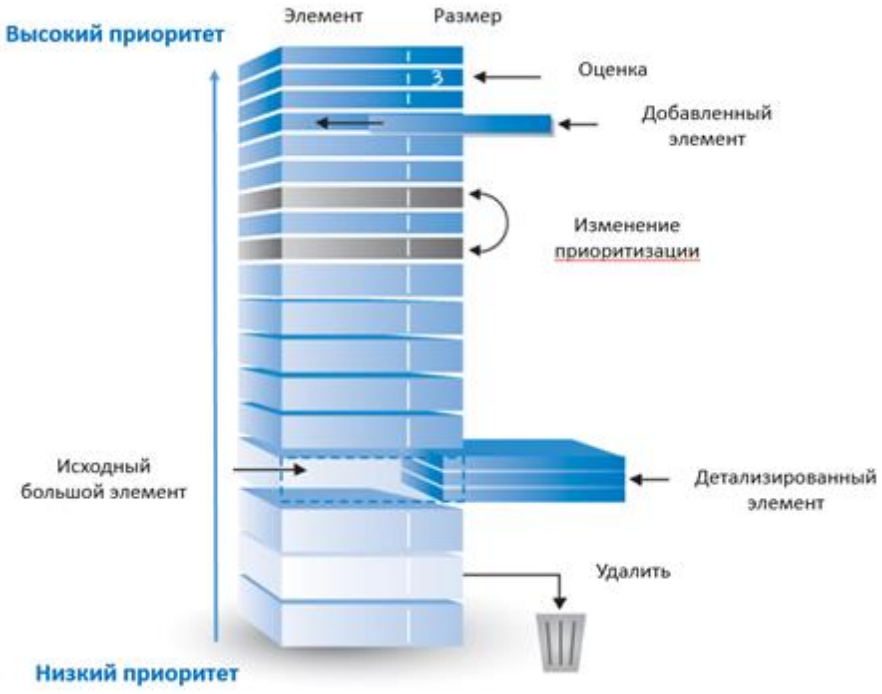


Рисунок 4. Схема работы с бэклогом продукта

Работа с бэклогом строится по определённым правилам.

Правило № 1. За бэклог и его уточнение отвечает владелец продукта, остальные участники команды могут помогать.

Правило № 2. Чем большую ценность для клиента или конечного пользователя несёт в себе функция, тем раньше она должна быть реализована и тем выше будет её позиция в списке.

Правило № 3. Чем выше находится элемент в списке бэклога, тем лучше он должен быть подготовлен к реализации: смысл истории понятен всем участникам команды, известны критерии приёмки, её объём легко оценить.

Правило № 4. Список элементов бэклога, как и их приоритизация, может меняться на протяжении всего проекта и адаптироваться к новым условиям.

Приоритизация элементов

Уточнение бэклога начинается с декомпозиции и приоритизации его элементов.

Декомпозиция — это метод, позволяющий заменить решение одной большой задачи решением серии меньших задач, взаимосвязанных, но более простых.

Команде разработки совместно с владельцем продукта необходимо обсудить весь список пользовательских историй, чтобы понять смысл каждого требования и его состав. Скорее всего, среди элементов будут и объёмные, включающие в себя несколько функций меньшего размера, которые могут иметь совершенно разный приоритет для бизнеса. Общая большая история для подмножества таких историй называется **эпиком**. Многие команды объединяют эпики в общие темы.

Бэклог мобильного приложения «Служба доставки продуктов»		
Тема	**Эпик**	**Пользовательские истории**
	Как покупатель, я хочу иметь свой личный кабинет, чтобы управлять своими заказами	Просматривать свои заказы
		Повторить свой заказ
		Отменить текущий заказ

Личный кабинет		Отследить статус заказа
		Осуществить возврат
	Как покупатель, я хочу иметь возможность управлять своими личными данными	Изменить личные данные
		Добавить выбор адреса
		Добавить выбор карты
		Добавить бонусные карты

Рисунок 5. Пример бэклога: тема, эпик, пользовательские истории

Корректность расстановки приоритетов зависит от многих факторов, но в первую очередь от бизнес-ценности элементов — соотношения выгоды, полученной от использования функции, и затрат, которые понадобятся на её реализацию.

Приоритизация элементов бэклога продукта с точки зрения коммерческой выгоды — это ответственность владельца продукта, который лучше всех в команде понимает, что необходимо получить бизнесу в первую очередь. Может показаться, что ценность — это достаточно понятный критерий для сортировки, но часто клиенту сложно определить, что важнее, или кажется, что всё имеет первостепенную важность. В данном случае команде необходимо проявить соучастие и высказать своё мнение или помочь с упорядочиванием задач.

Существуют специальные методы и техники, призванные помочь с приоритизацией элементов, особенно в том случае, когда их очень много или клиент не может определиться с тем, что важнее.

Например, один из самых простых методов для приоритизации требований — метод MSCW. Это техника, широко используемая в управлении проектами, бизнес-анализе и разработке программного обеспечения.

MoSCoW-метод

В основе метода четыре категории приоритетов, начальные буквы которых создают акроним MSCW.

Must — должно быть обязательно: базовые или срочные функции, имеющие первостепенное значение. Чаще всего это как раз тот объём требований, который необходим для создания MVP.

Should — надо сделать: важные, добавляющие ценность функции. Обычно они не имеют решающего значения, но всё равно обязательны к исполнению.

Could — желательно: не входит в обязательный пул работ, но может быть реализовано, если появится возможность.

Would — хотелось бы: функции с самым низким приоритетом. Мы ничего не потеряем, если их не будет.

В результате упорядочивания появятся четыре последовательные группы элементов.

Если пользовательских историй много и они имеют взаимосвязь, то внутри групп часто применяют систему ранжирования путём присвоения рейтинга каждому элементу в списке.

Ренкинг (ranking, ранкинг) — расположение, расстановка, точное упорядочивание по какому-либо показателю или признаку.

При ранжировании задач необходимо:

- оценить важность с технической стороны и со стороны бизнес-выгод;
- выделить наиболее востребованные задачи со стороны большинства;
- выделить важные задачи со стороны ключевых пользователей, спонсоров;
- учесть технические риски, взаимозависимости, интеграцию.

Для ренкинга команды обычно используют не последовательные числа, а десятки, сотни или тысячи, это позволяет добавлять новые требования, присваивая им промежуточный ренк. Главное правило: не может быть двух историй с одинаковой важностью.

Бэклог мобильного приложения «Служба доставки еды»				
Тема	**Эпик**	**Пользовательские истории**	**MSCW**	**Ренк**
Личный кабинет	Покупатель / управление заказами	Просматривать свои заказы	M	100
	Покупатель / управление личными данными	Изменить личные данные	M	90
	Покупатель / управление заказами	Отменить текущий заказ	M	80
	Покупатель / управление заказами	Осуществить возврат	M	70
	Покупатель / управление личными данными	Изменить пароль	M	60
	Покупатель / управление заказами	Отследить статус заказа	S	100
	Покупатель / управление личными данными	Добавить выбор адреса	S	90
	Покупатель / управление личными данными	Добавить выбор карты	S	80
	Покупатель / управление заказами	Повторить свой заказ	C	100

	Покупатель / управление личными данными	Добавить бонусные карты	W	100

Рисунок 6. Пример оформления бэклога: приоритизация по методу MSCW и присвоение рейтинга историям

На примере бэклога мобильного приложения «Служба доставки продуктов» можно увидеть, что после приоритизации список сформирован по группам, каждая из которых имеет свой порядок рейтинга. В случае добавления новых историй или изменения приоритетности рейтинг присваивается в соответствии с выделенным местом в новой группе.

Если команда работает по этому принципу приоритизации, то чаще всего в MVP входят задачи из группы MUST — «должно быть обязательно». Далее в зависимости от бюджета и сроков продукт совершенствуется за счёт наращивания функциональности из остальных групп в порядке приоритетности.

Основное достоинство такого подхода — простота, однако, если список задач слишком большой, анализ требований может занять слишком много времени. В таком случае Agile-команды предпочитают сначала сделать верхнеуровневую приоритизацию всего списка по MSCW независимо от размера элементов, а декомпозицию совершать по необходимости при подготовке элемента к реализации.

Story mapping

Ещё один популярный способ приоритизации задач — метод Story mapping (карта историй). Она позволяет визуализировать объём работ, опираясь на темы, эпики или отдельные процессы.

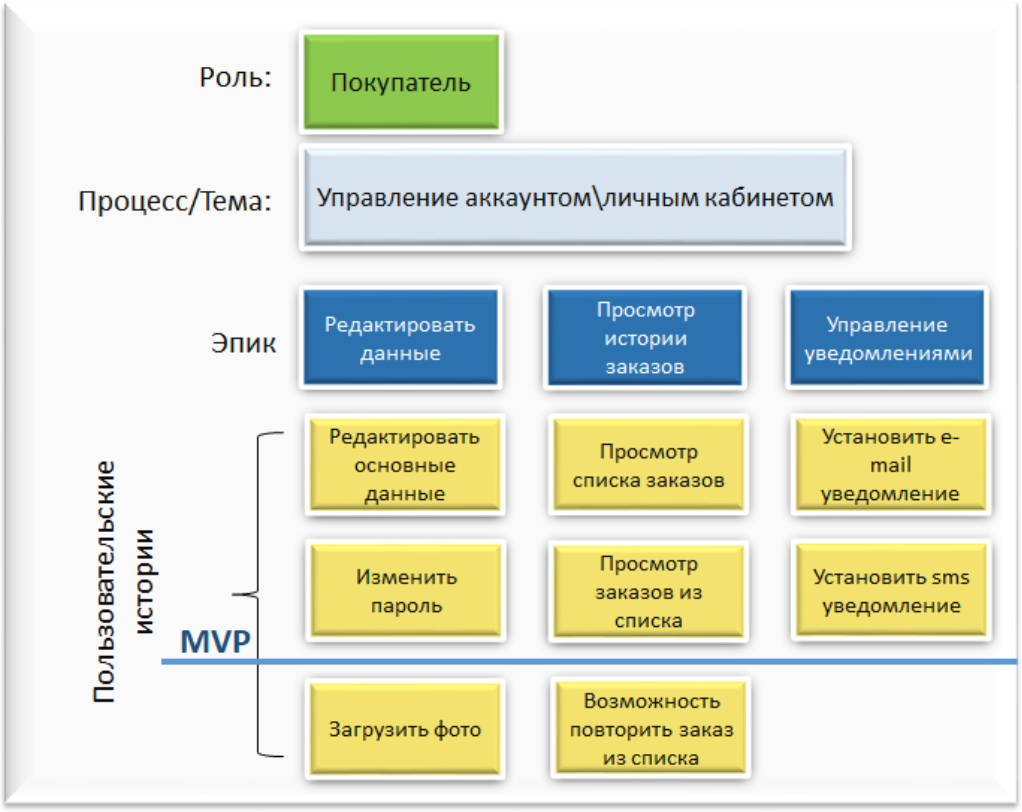


Рисунок 7. Пример оформления карты историй на основе темы и эпиков

Рассмотрим логику работы с картой историй на примере мобильного приложения «Служба доставки еды», опираясь на тему и эпики:

1. Обозначить на карте роль, тему и эпики.
2. Декомпозировать эпики на пользовательские истории по принципу «чем ниже история находится в колонке, тем ниже её приоритет».
3. Определить, что должно войти в MVP.
4. В случае изменения приоритетов или появления/удаления историй, карточки меняются местами.

Второй вариант составления карты — ориентироваться на процесс, то есть порядок действий той или иной роли:

1. Горизонтальная ось представляет последовательность использования. Истории на ней размещаются в последовательности, в которой они выполняются пользователем, — это будет «хребет» карты историй.
2. Вертикальная ось представляет детализацию каждого действия в «хребте» процесса с приоритизацией. Истории располагаются по степени важности сверху вниз. Одинаково важные задачи можно определять на одной высоте.

Рисунок 8. Пример оформления со связанными между собой задачами в хребте карты

Из примера с действиями покупателя книжного онлайн-магазина видно, что каждое действие разбито на ряд функций, которые оно в себя может включать. С помощью такого способа визуализации легко определить, какие истории приоритетны и должны войти в MVP.

Основной недостаток карты историй — визуализировать весь объём проекта представляется сложной, а иногда и невозможной задачей. Этот метод чаще всего используется для простых продуктов или для проработки отдельных элементов, содержание и объём которых не очевидны.

К основным плюсам этого метода можно отнести визуализацию и очевидное построение связей между функциями, на основе которых проще выделить объём, который войдёт в MVP.

После распределения элементов по приоритетам наступает очередь оценки. За оценку задач отвечает команда разработки.

Оценка элементов бэклога

К оценке элементов команда приступает только в том случае, если истории имеют понятное для всех участников содержание: смысл и общий состав работ.

В связи с тем, что на старте проекта состав работ и финальная конфигурация продукта обладают наибольшей степенью неопределённости, командам сложно прогнозировать сроки разработки конкретных функций или свои трудозатраты. Как показывает опыт, план работы, сформированный на основе оценок относительно срока реализации тех или иных функций, оказывается нежизнеспособным,

как только начинают происходить изменения в составе работ, которые неминуемы на всех проектах. Agile-концепция основана на гибкости, принятии и адаптации к изменениям, в том числе и в объёме работ, поэтому перед командами не стоит задача точного прогнозирования сроков.

Выделяют два основных способа оценки элементов бэклога: в идеальных днях и пойнтах (баллах).

Первый способ основан на том, что сотрудник не продуктивен на 100% в течение рабочего дня, потому что он не только работает над задачами, но и отвлекается на совещания, письма, телефонные звонки, беседы с коллегами и личные перерывы. По статистике, среднее продуктивное время составляет 50–60%, поэтому один идеальный восьмичасовой день будет равен 1,5–2 стандартным рабочим дням.

Команде необходимо оценивать элементы бэклога именно в идеальных днях. Основной недостаток такого подхода: оценки в днях всегда привязаны к опыту сотрудника. Два идеальных дня для одного — это один или три идеальных дня для другого. К тому же, участники часто путают идеальные и обычные дни в обсуждении прогнозов по срокам работы. Несмотря на это, опытные команды часто предпочитают такой способ оценки и делают достаточно успешные прогнозы. Однако команды, участники которых не имеют достаточного уровня компетенции, сталкиваются с большими сложностями.

Считается, что когда человек не может точно оценить какую-то величину, самый верный способ — сравнение объекта оценки с величиной, которая наверняка известна. Scrum предлагает оценивать элементы бэклога, исходя не из срока их реализации, а из предполагаемой сложности конкретного элемента относительно остальных элементов в списке. Оценивать необходимо именно усилия, которые требуются для реализации той или иной функции, каждый раз сравнивая её с другими элементами.

В оценку усилий следует заложить все факторы, которые могут повлиять на неё:

- объём требуемой работы;
- техническую сложность задачи и зависимости;
- возможные риски и неопределённость в требованиях.

Пользовательским историям в данном случае присваиваются Story points — очки, баллы или пункты сложности, по которым история занимает своё место в общей шкале от самого маленького (лёгкого) элемента к самому большому (сложному) элементу.

Для оценки задач можно использовать любую шкалу, например степени двойки: 1, 2, 4, 8 и 16. Некоторые команды, чтобы как можно больше абстрагироваться от привязки к срокам, используют размеры маек: XS, S, M, L, XL. Однако самая распространённая шкала — числа Фибоначчи. Это понятные числовые значения, последовательность которых хорошо отражает рост неопределённости, возникающий с ростом сложности оцениваемой задачи.

Как это работает на практике:

1. Команде необходимо выбрать самый простой элемент и присвоить ему наименьший балл, равный единице. Этот элемент будет эталоном — точным образцом.
2. Элемент, равный 2, будет в 2 раза сложнее эталона.
3. Элемент, равный 3, — это сумма усилий, которые потребуется приложить для разработки элементов, равных 1 и 2 и т. д. по аналогии, используя ряд Фибоначчи, где каждое следующее число — это сумма предыдущих.

Обычно команды используют следующий набор чисел для оценки: 0.5, 1, 2, 3, 5, 8, 13 и 20, где 0,5 — это истории, требующие совсем незначительных усилий, а 20 — это задачи с наибольшей степенью неопределённости. Считается, что задачи, имеющие оценку сложности больше 20, необходимо декомпозировать, потому что работать с такими большими элементами — это неоправданный риск.

Специально для вовлечения всех участников команды в процесс была придумана коллективная игра Planning poker — покер планирования.

Покер планирования

Джеймс Греннинг описал этот метод в 2002 году. Впоследствии он стал настолько популярным, что теперь можно купить настоящие колоды карт или поиграть в покер планирования онлайн.

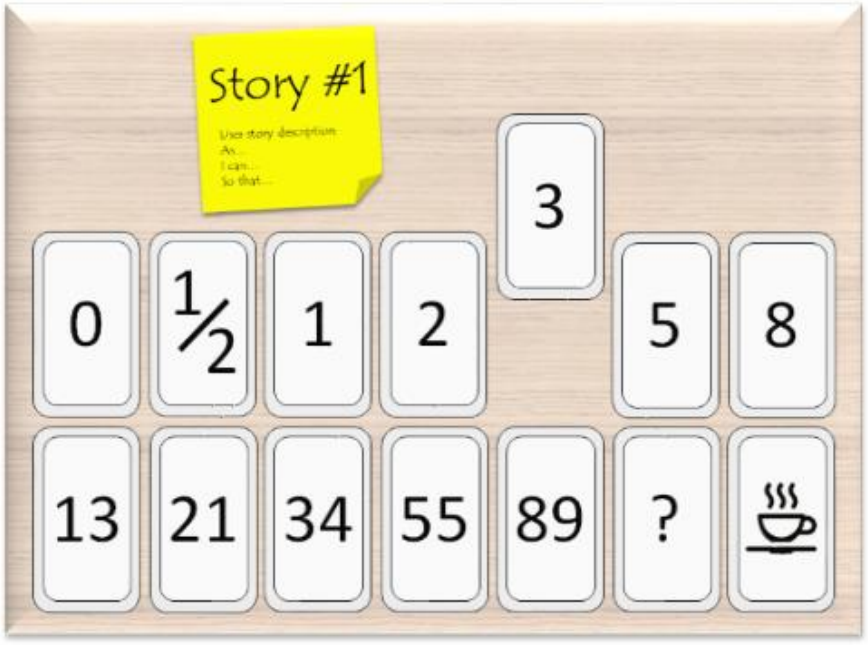


Рисунок 9. Пример набора карточек для игры в покер планирования

Суть метода заключается в следующем:

1. Всем участникам команды раздаются карты с числами из шкалы оценки.
2. Затем выбирается задача, зачитываются и обсуждаются требования к ней.
3. После обсуждения модератор просит всех членов команды выбрать карту с оценкой и положить её «рубашкой» вверх.
4. Затем модератор даёт сигнал показать карты.

Если участники поставили одну и ту же оценку, то она фиксируется как единогласно принятая и к рассмотрению выбирается следующая задача. Стоит отметить, что согласие не должно быть абсолютным. Команда может договориться, что набор соседних оценок также считается согласием. Можно договориться согласовать вариант, который показали большинство участников.

Если участники сильно разошлись во мнениях, то инициируется обсуждение. В первую очередь своё мнение высказывают те участники, которые поставили самую низкую или самую высокую оценку, затем все желающие. После того как все высказались, раунд проводится повторно, и так до тех пор, пока команда не найдёт устраивающий всех вариант.

Покер планирования — достаточно ресурсоёмкое мероприятие и требует времени, но его преимущество заключается в том, что оценки опираются на совокупность опыта и независимых мнений всех участников.

Вот как может выглядеть бэклог мобильного приложения «Служба доставки продуктов» после оценки.

Бэклог мобильного приложения «Служба доставки еды»					
Тема	**Эпик**	**Пользовательские истории**	**MSCW**	**Ранк**	**Оценка**
Личный кабинет	Покупатель / управление заказами	Просматривать свои заказы	M	100	3
	Покупатель / управление личными данными	Изменить личные данные	M	90	1
	Покупатель / управление заказами	Отменить текущий заказ	M	80	2
	Покупатель / управление заказами	Осуществить возврат	M	70	5
	Покупатель / управление личными данными	Изменить пароль	M	60	1
	Покупатель / управление заказами	Отследить статус заказа	S	100	3
	Покупатель / управление личными данными	Добавить выбор адреса	S	90	2
	Покупатель / управление личными данными	Добавить выбор карты	S	80	2
	Покупатель / управление заказами	Повторить свой заказ	C	100	2

	Покупатель / управление личными данными	Добавить бонусные карты	W	100	2

Рисунок 10. Пример бэклога продукта с оценкой историй

Важно помнить, что после того, как команда произвела оценку элементов бэклога, необходимо ещё раз вернуться к приоритизации историй и пересмотреть её с учётом новых знаний. В ходе оценки могут появиться функции, связанные с технической базой продукта и не очевидные для владельца продукта. Задачи по созданию технической базы для реализации функционала всегда будут в группе must-have с высоким рейтингом.

Перед началом оценки команде необходимо определиться, с каким способом оценки она будет работать: идеальные дни или пойнты. Подход к оценке не должен меняться на протяжении всего проекта. В оценке задач участвует вся команда, и каждый участник должен высказать своё мнение относительно оценки. Финальное решение должно быть коллективным.

Все оценки носят относительный характер, это лишь предположения, особенно на начальном этапе планирования. В процессе работы неизбежны изменения. Может оказаться, что какие-то истории были недооценены или переоценены. В данном случае команда не пересматривает оценки и не вносит изменения в первоначальные значения. Важнее сравнить результат с прогнозом на ретроспективе спринта и учесть полученные знания в будущем. Чем больше опыта будет получать команда, тем точнее будут становиться прогнозные оценки.

Следующим шагом будет определение объёма работ, который войдёт в первый релиз продукта — MVP, и распределение его по спринтам.

Производительность команды

Перед стартом проекта Scrum-команде необходимо определиться, какой величины спринт она выбирает для работы. Величина спринта должна быть достаточной, чтобы команда смогла показать готовый результат. Это может зависеть от сложности продукта и степени неопределённости требований. Чаще всего команды выбирают промежуток в две недели, но если степень неопределённости крайне высока, а функциональность продукта достаточно проста, то команда может использовать самый короткий вариант — в одну неделю. Это позволит чаще получать обратную связь и оперативно вносить изменения. Если продукт сложный и в оценке историй много тяжеловесных функций, команда может увеличить промежуток до четырёх недель, чтобы иметь достаточно времени для тестирования и отладки.

После того как команда выбрала длину спринта, необходимо определить производительность, с которой команда может работать. Это позволит спрогнозировать, сколько спринтов потребуется команде для разработки MVP и запуска первого релиза.

Производительность в терминах Scrum — это скорость движения команды (velocity).

Скорость работы команды (velocity) — это сумма оценок в идеальных днях или баллов, реализованных за спринт.

Для прогноза сроков реализации используют среднюю скорость работы команды.

Рассчитывается следующим образом:

1. Команда складывает все баллы, присвоенные при оценке историям, которые она успела выполнить за спринт. Например, за 2 недели команда успела сделать истории со следующими баллами сложности: 3, 5, 1 и 8. Значит, сумма баллов за спринт — 17. Это производительность команды за конкретный спринт.
2. Далее необходимо сложить общее количество баллов по нескольким прошедшим спринтам. Например, за несколько последних спринтов команда показала следующие результаты производительности: 12, 14, 19, 15 и 17. Сумма баллов за 5 прошедших спринтов — 77.
3. Считаем среднее арифметическое: 77 : 5 = 15,4 — это средняя скорость, с которой предположительно будет двигаться команда.

Исходя из полученных данных о средней скорости команды, можно спрогнозировать, сколько спринтов понадобится команде, чтобы выпустить первый релиз MVP. Например, если Scrum-команда определила для себя, что будет работать спринтами по 2 недели, а объём работ, необходимый для разработки MVP, равен 120, то команде понадобится около 8 спринтов, чтобы выполнить свои обязательства. Это 4 месяца. При этом не надо относиться к полученной цифре как к точному дедлайну. Может получиться и так, что команда будет двигаться медленнее среднего значения или, наоборот, быстрее, поэтому ориентир — это лишь прогноз.

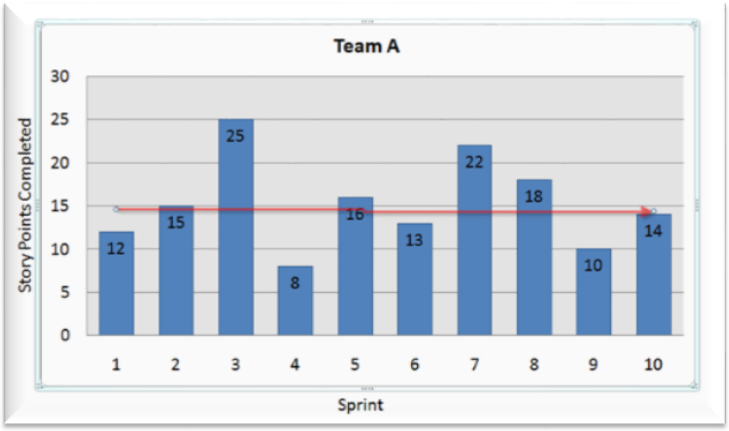


Рисунок 11. Производительность команды по спринтам и уровень её средней скорости

Важный вопрос: каким образом узнать производительность команды, если она новая и проект только начинается?

В этом случае Agile предлагает опираться на эмпирический подход к делу. Необходимо сделать предположение, а затем после каждого спринта с учётом полученного опыта адаптироваться и корректировать прогноз. Считается, что после нескольких спринтов команда выйдет на средний показатель своей скорости.

В случае определения скорости на основе пойнтов не учитывается степень доступности членов команды, которая может меняться от спринта к спринту.

В случае работы с идеальными днями прогноз может быть точнее, и считается он следующим образом:

1. Определить, сколько календарных дней на спринт есть у команды: 4 человека × 5 дней в неделю = команда имеет 80 календарных дней на спринт в 4 недели.
2. Перевести календарные дни в идеальные: 50% от 80 = 40 идеальных дней.
3. Заложить риск на фокус-фактор (раскачку команды) для первого спринта: 40% от 40 идеальных дней = 24 идеальных дня.
4. Для второго спринта: 20% от 40 идеальных дней = 32 идеальных дня.
5. Для последующих спринтов: средняя скорость предыдущих спринтов.

Расчёт начальной скорости:
Общее количество человекодней на спринт
(4 чел. х 5 дней) х 4 нед. = 80
Идеальные дни
50%
Средняя скорость
40
Фокус фактор для 1ого спринта (-40%)
24
Фокус фактор для 2ого спринта (-20%)
32

Рисунок 12. Расчёт начальной скорости команды в идеальных днях

Получается, что команда по прогнозу возьмёт в работу такое количество элементов, которое будет соответствовать 24 идеальным дням в первый спринт и 32 идеальным дням во второй спринт. Далее можно спрогнозировать, сколько команде потребуется спринтов, чтобы реализовать объём функционала, выбранного для реализации MVP. При общем объёме в 120 идеальных дней и средней производительности команды в 40 идеальных дней на четырёхнедельный спринт команде понадобится 3 спринта или 3 месяца.

При расчёте по этой схеме можно учесть, что не все члены команды работают полный рабочий день. Например, два разработчика из четырёх работают на двух проектах параллельно, выделяя на каждый по полдня, тогда общее количество календарных дней на команду будет уже не 80, а 60. Остальные показатели также изменятся, соответственно, прогнозный срок релиза будет уже не 3 спринта, а 4.

Оценка в идеальных днях, как и оценка в пойнтах, не пересматривается в ходе проекта. Изменения, происходящие в процессе реализации, анализируются на ретроспективе после каждого спринта.

Первая большая часть подготовки к старту проекта на этом завершается, и команда переходит к планированию спринта. К уточнению бэклога продукта Scrum-команда будет возвращаться после каждого спринта на его обзоре. Новые задачи, в том числе задачи, связанные с выявленными ошибками, будут проходить тот же цикл работ: декомпозиция, приоритизация, оценка.

Планирование спринта (Sprint Planning Meeting)

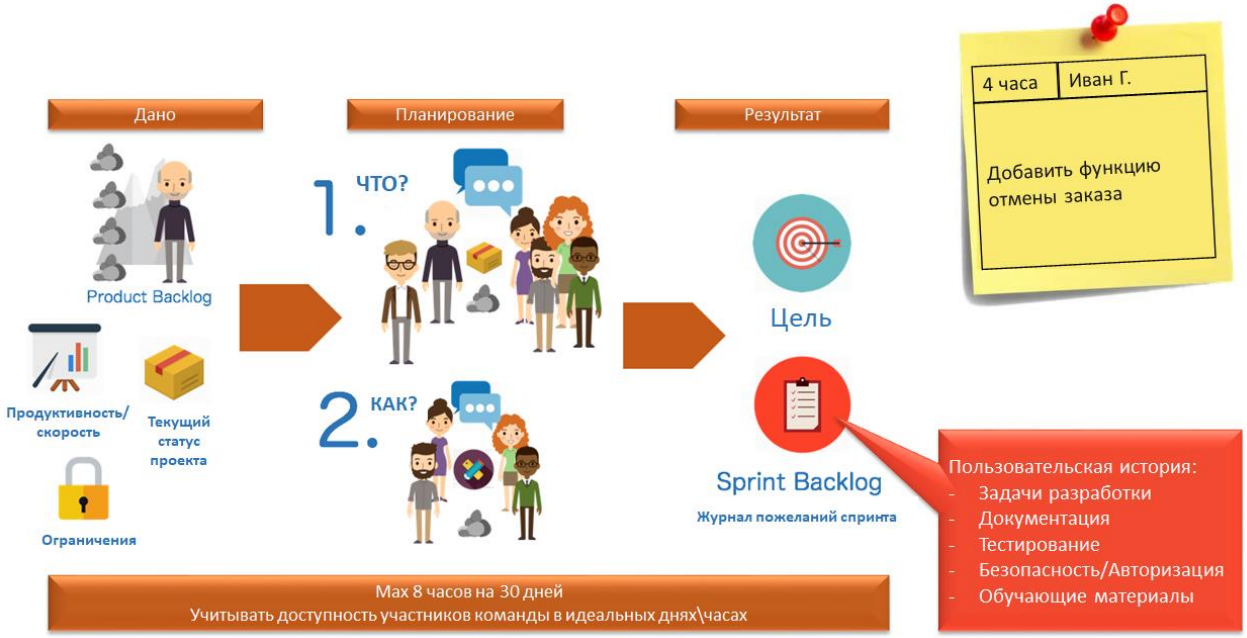


Рисунок 12. Планирование спринта

Продолжительность совещания ограничивается 4–8 часами в зависимости от продолжительности итерации.

Встреча разбивается на две части.

Первая часть совещания: участвует вся Scrum-команда

1. Формируется цель спринта.
2. В соответствии с целью спринта и прогнозируемой производительностью команды из бэклога продукта выбираются пользовательские истории, обязательства по выполнению которых принимает на себя команда.
3. Обсуждаются и фиксируются критерии готовности и приёмки для каждой истории.

Вторая часть совещания: участвует только команда разработки

1. Декомпозиция историй на конкретные задачи: технические детали реализации, взаимосвязи, тестирование и документация.
2. Оценка задач: решение задачи не должно занимать более 12 идеальных часов или 2 рабочих дней. При необходимости задача разбивается на подзадачи.
3. Формирование бэклога спринта.
4. Оформление доски визуализации.

Результат планирования спринта — сформулированная цель спринта и бэклог спринта, который содержит истории и конкретные задачи к ним. Каждая задача должна иметь краткое название, понятное всем участникам, подробное описание, оценку в часах, критерий готовности и критерий приёмки.

Для визуализации процесса работы над спринтом команды используют Scrum-доску (часто её называют просто Kanban-доской) и диаграммы сгорания.

На доске оформляется несколько колонок в зависимости от правил, которых придерживается команда. Чаще всего это:

1. Бэклог спринта: список историй.
2. Задачи, готовые к работе (понятна, имеет подходящий размер, оценку, критерий готовности и приёмки).
3. Задачи, которые находятся в работе.
4. Готовые задачи.

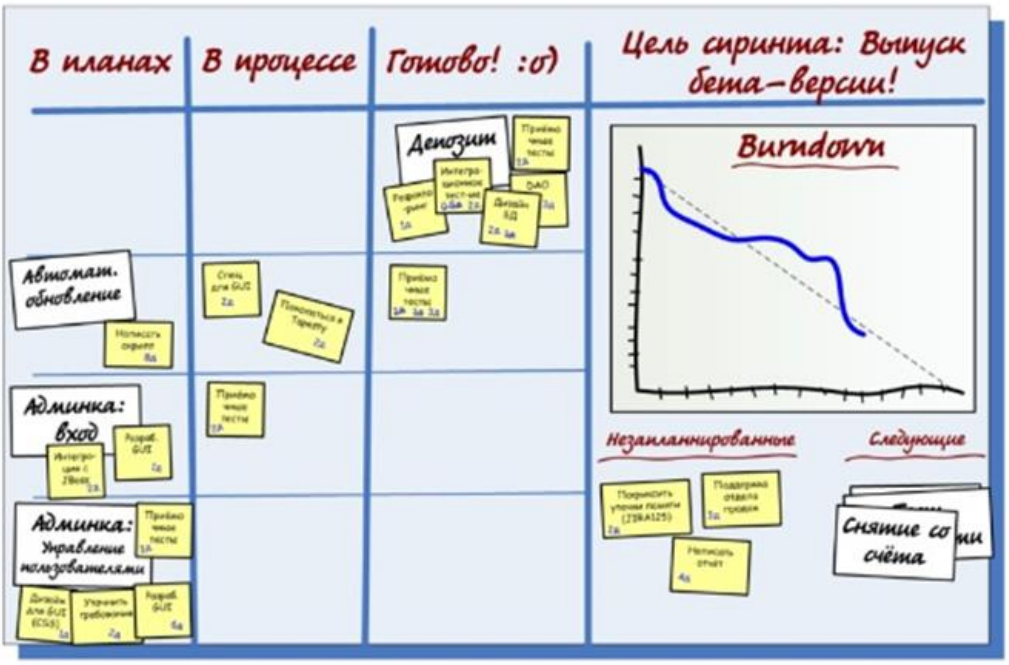


Рисунок 13. Scrum-доска и диаграмма сгорания спринта

Диаграмма сгорания показывает график, который формируется на основе скорости работы команды. По оси X обозначают время в днях, по оси Y — общее количество баллов / идеальных дней из бэклога спринта, каждый день делают отметку с количеством оставшейся работы. Получившийся график позволяет отслеживать ежедневную динамику выполнения работ и целесообразен только для спринтов длиной от двух недель.

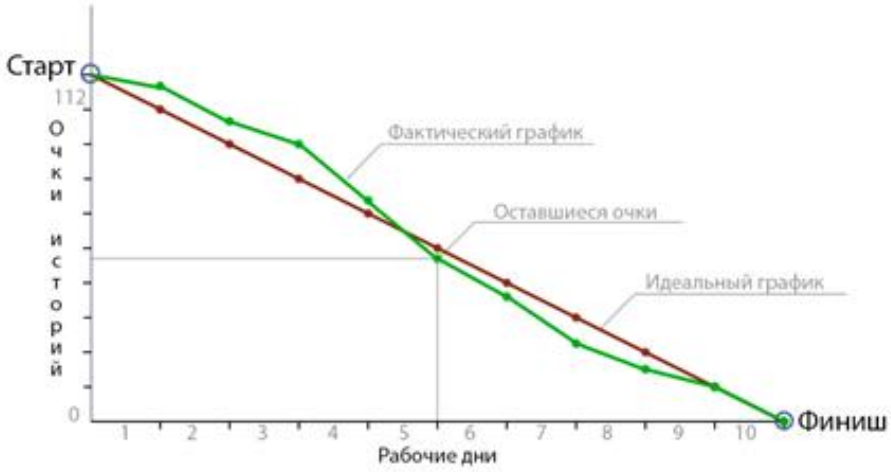


Рисунок 14. Диаграмма сгорания спринта

Для крупных проектов с обозначенным дедлайном или зафиксированным набором функционала команды используют диаграмму сгорания и в рамках релизов или всего проекта. В таком случае график по оси X отражает количество спринтов, а по оси Y — общее количество баллов / идеальных дней из бэклога продукта.

Для распределённых команд, работающих онлайн, актуально использование виртуальных Scrum-досок.

Использование виртуальных досок

Scrum-команды, которые работают онлайн, используют виртуальные доски для визуализации и оптимизации рабочего процесса. Виртуальные аналоги давно стали ключевой функцией любого инструмента Agile-разработки, ПО в том числе. Многие команды перешли на работу с виртуальными досками и в офисах: с ними проще отследить процессы, а также организовать совместную работу и доступ из разных мест.

Виртуальная доска не только позволяет визуализировать и оптимизировать рабочий процесс, но и может стать единственным достоверным источником информации о работе команды и статусе прогресса. Это позволяет обеспечить полную прозрачность работы и обмен информацией в режиме реального времени.

Помимо визуализации процесса виртуальная доска обладает ещё одним важным преимуществом: её функциональность позволяет фиксировать все действия с карточками, информировать участников об изменениях в статусах задач. Также она может быть удобной базой данных после завершения проекта, поскольку на доске сохраняются все важные артефакты проекта.

Три наиболее популярных сервиса виртуальных досок: Trello, Jira, Asana. Сервисы сильно отличаются друг от друга пользовательским интерфейсом, у каждого есть свои особенности и довольно много бесплатных функций. Trello и Jira создала IT-компания Atlassian, её продукты переведены на русский язык. Asana по состоянию на август 2020 года русский язык не поддерживает. Как выглядит их пользовательский интерфейс, вы можете увидеть на примере классической Kanban-доски. В примерах

— доски из разных сфер деятельности, чтобы можно было оценить их адаптивность под задачи любого пользователя.

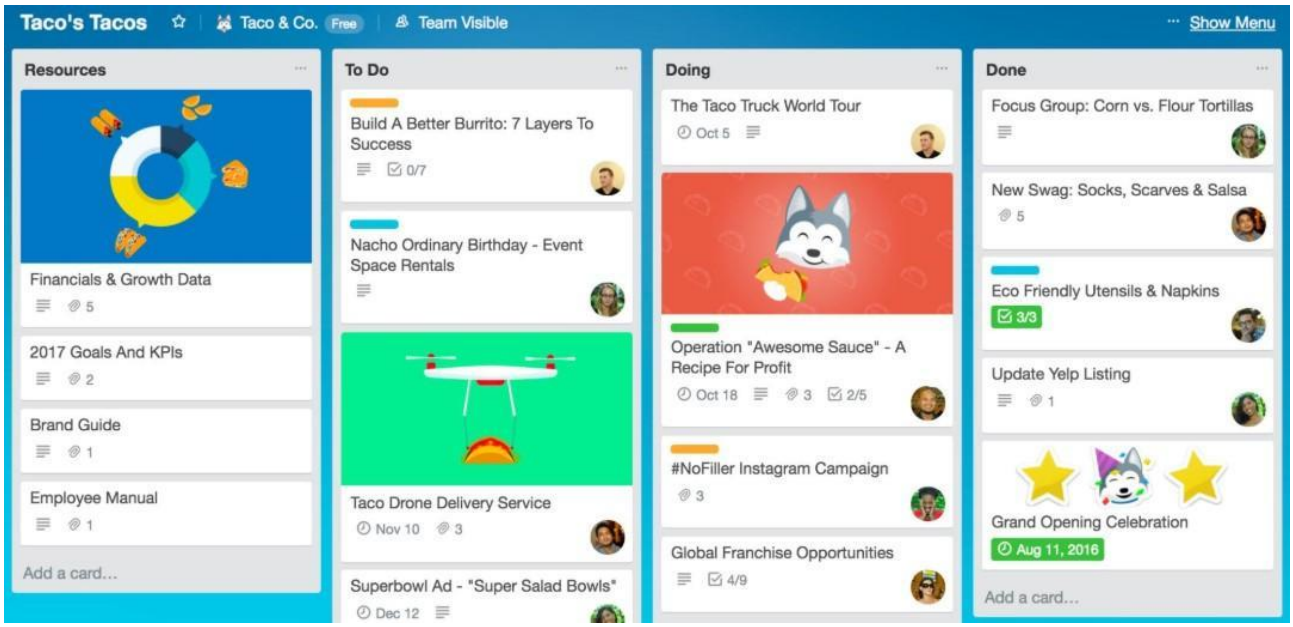


Рисунок 15. Пример Kanban-доски в Trello

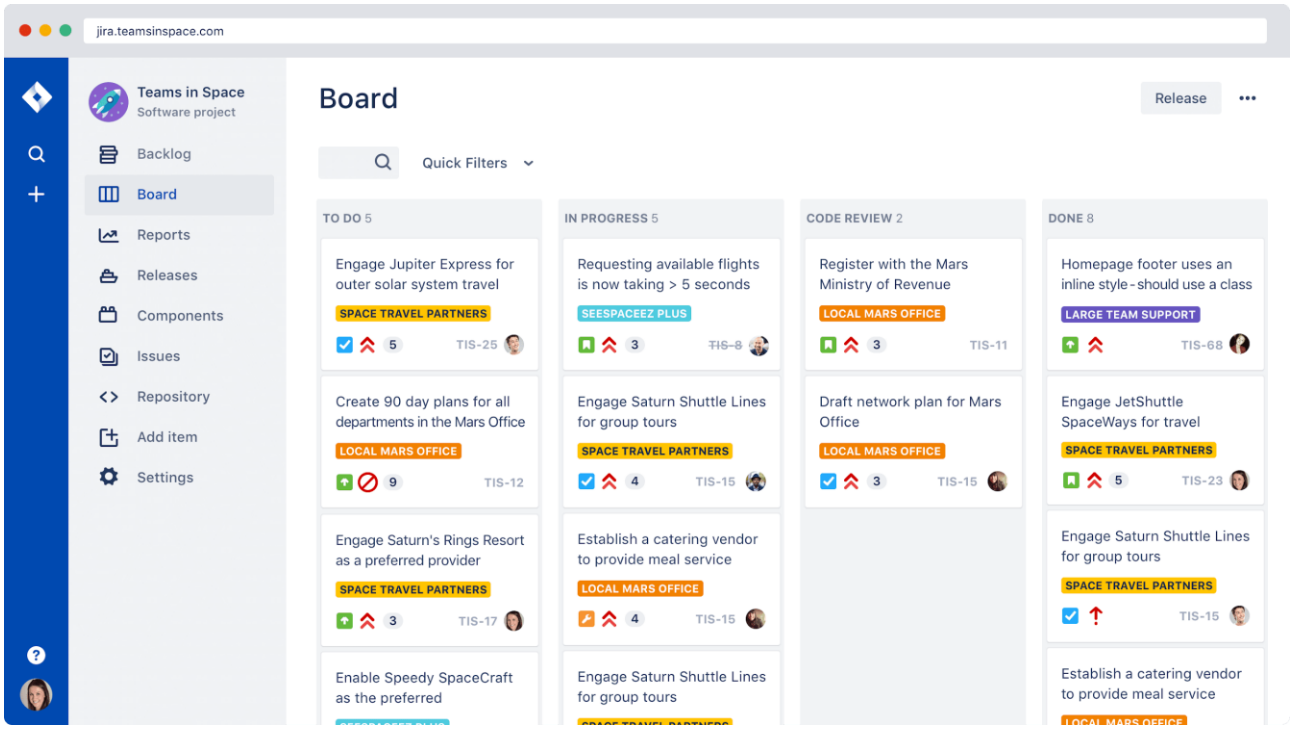


Рисунок 16. Классический и самый простой вариант Kanban-доски в Jira

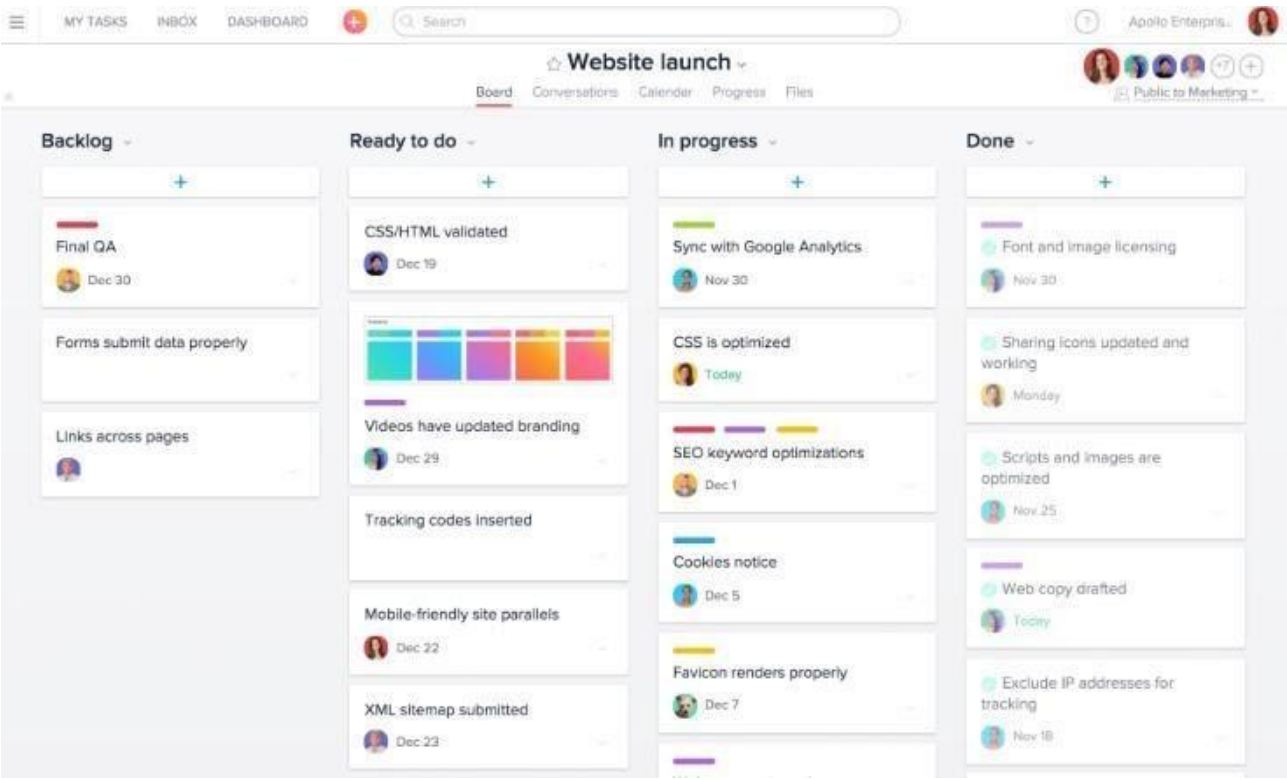


Рисунок 17: Kanban-доска Asana, адаптированная под выпуск контента

Преимущества использования виртуальных досок в командной работе

Самое главное в работе с онлайн-доской — строго придерживаться правила совместной работы: доска должна стать единственным достоверным источником информации о проекте, чтобы обеспечить полную прозрачность работы и обмен информацией в режиме реального времени. Таким образом, тщательное заполнение карточек и ведение онлайн-доски даёт команде несколько преимуществ:

1. Визуализация и организация процесса разработки.
2. Простота в оформлении и использовании.
3. Возможность персонализировать и настроить доску в соответствии с особенностями проекта и с принятыми в команде правилами работы.
4. Возможность совместного доступа и прозрачность рабочего процесса: откуда и когда бы ни обращались участники команды к виртуальной доске, они всегда увидят самый актуальный статус проекта.
5. Возможность отследить зависимости и оперативно обнаружить упущения, которые могут помешать прогрессу.
6. Хранение всех артефактов проекта в структурированном виде и в хронологическом порядке, что позволяет в любой момент получить доступ к детализированной информации о любом артефакте проекта.
7. Возможность отслеживания неограниченного количества обсуждений и комментариев в любое время по мере реализации проекта с удобными настройками уведомлений.

8. После завершения проекта доску можно очистить, удалив из неё всю конфиденциальную информацию, но сохранив полезные вещи в свободном доступе. Такая доска может стать отличной визитной карточкой реализованных проектов.

Процесс работы с виртуальной доской на примере Trello

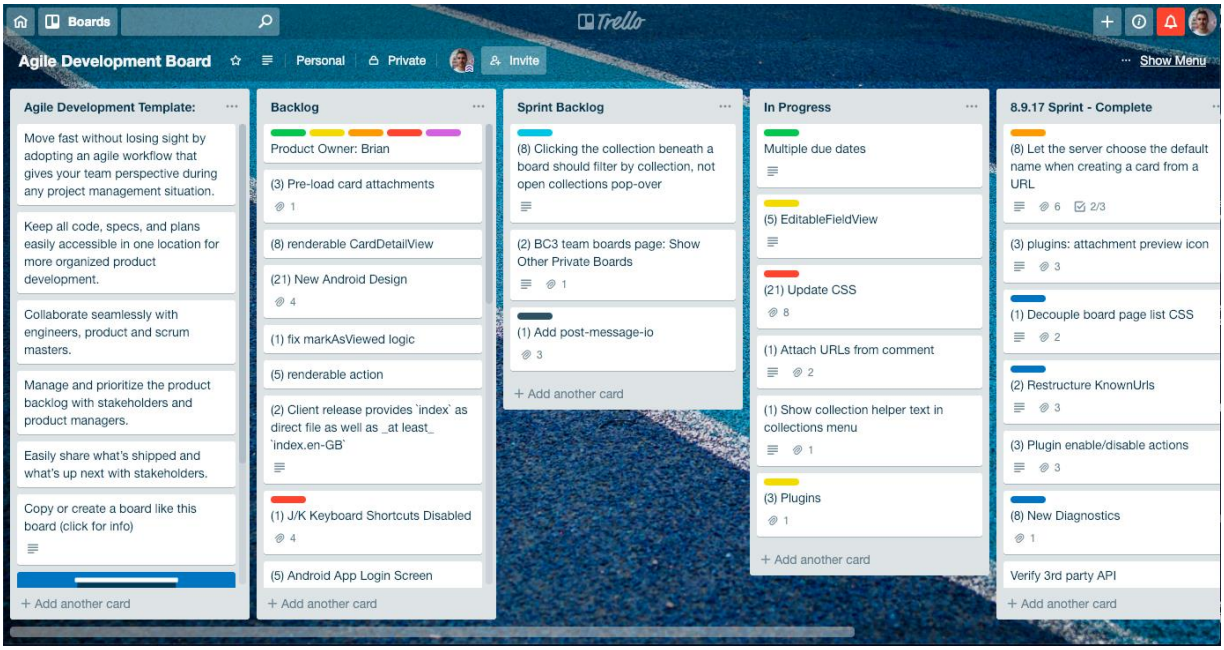


Рисунок 18. Пример классической виртуальной Scrum-доски Trello

Рассмотрим подробнее работу с виртуальной доской на примере Trello. Это бесплатный интуитивно понятный сервис, который оптимально подходит для работы с небольшими несложными проектами.

Как вы видите на рисунке, в Trello можно создать виртуальную доску, которая будет полностью соответствовать требованиям руководства по Scrum.

Стандартная Scrum-доска будет содержать четыре обязательные колонки:

1. Колонка с бэклогом продукта — за неё отвечает владелец продукта.
2. Колонка с бэклогом спринта — перечень задач, над которыми будет трудиться команда разработки во время спринта. Бэклог спринта формируется в результате планирования спринта всей Scrum-командой.
3. Колонка «В работе» — сюда разработчики перемещают те задачи, которые берут в работу. В Kanban и Scrumban число таких задач ограничено WiP limits.
4. Колонка завершённых в рамках текущего спринта задач — сюда переносятся полностью завершённые и протестированные задачи.

Колонку «В работе» (In Progress) чаще всего формируют после ежедневных коротких Scrum-совещаний (стендапов), в ходе которых команда разработки планирует свою работу на ближайшие 24 часа.

В работу берутся только готовые к реализации задачи, у которых есть:

- краткое название;
- подробное описание и технические детали, могут быть добавлены схемы или макет дизайна;
- оценка в часах;
- критерий «Готово» для команды;
- критерий приёмки для владельца продукта;
- после попадания задачи в колонку «В работе» необходимо указать исполнителя.

Поскольку Scrum-доска предусматривает создание любых других дополнительных колонок для удобства команды, то многие пользователи выделяют ещё одну колонку — «Тестирование», чтобы не забыть уделить этому особое внимание или если тестированием занимаются QA-специалисты. Ещё может быть добавлена колонка «Отложено», куда переносятся задачи, которые в силу определённых причин невозможно завершить в рамках текущей итерации/спринта (заказчик отложил реализацию функции на неопределённый срок, субподрядчик/партнёр не успел выпустить свою часть продукта и т. д.).

В Scrum незавершённые задачи после окончания спринта возвращаются в бэклог продукта, сортируются в соответствии с новыми приоритетами и берутся в работу (если это, конечно, потребуется) в рамках нового спринта.

По завершении спринта проводится его обзор с целью демонстрации инкремента продукта стейкхолдерам и адаптации бэклога продукта, а также ретроспектива спринта для проведения внутреннего аудита и создания плана внедрения улучшений в рамках будущего спринта.

Заключение

В четвёртом теоретическом уроке мы разобрали инструменты, которые необходимы для подготовки к старту проекта с использованием фреймворка Scrum. Этих знаний и знаний, которые вы получили на предыдущих уроках, будет достаточно, чтобы организовать процесс разработки с использованием Agile-концепции.

Первая неделя практического блока будет посвящена подготовке команд к старту проекта: изучению ТЗ, подготовке бэклога продукта и подготовке бэклога спринта № 1.

Глоссарий

Бэклог продукта (Product Backlog) — это упорядоченный список всего, что может понадобиться в продукте. Перечень требований и функций, поступивший от заказчика.

Декомпозиция — это метод, позволяющий заменить решение одной большой задачи решением серии меньших задач, взаимосвязанных, но более простых.

Пользовательские истории (англ. User Story) — способ описания требований к разрабатываемой системе/продукту, сформулированных как одно или более предложений на повседневном или деловом языке пользователя.

Ренкинг (ranking, ранкинг) — расположение, расстановка, точное упорядочивание по какому-либо показателю или признаку.

Скорость работы команды (velocity) — это сумма оценок в идеальных днях / баллов, реализованных за спринт.

Уточнение бэклога продукта — это совместная деятельность владельца продукта и команды разработки по уточнению, оценке и приоритизации элементов бэклога продукта.

MVP (Minimum Viable Product) — минимально жизнеспособный продукт. Это завершённый, готовый к запуску продукт с базовым функционалом, которого достаточно, чтобы удовлетворить основные потребности клиента или конечного пользователя.

SMART — мнемоническая аббревиатура, используемая для определения целей и постановки задач в проектном менеджменте, управлении производством и личном развитии.

Дополнительные материалы

1. Как писать User story (статья).
2. Бэклог груминг (статья).
3. Приоритизация и оценка задач (статья).
4. Story mapping (статья).
5. What story point is? Mike Cohn (видео).
6. Диаграмма сгорания спринта (статья).
7. Бесплатный покер планирования для распределённых команд.
8. Бесплатный сервис для командного ведения виртуальных досок.

Используемые источники

1. Руководство по Scrum (Кен Швабер, Джефф Сазерленд)

2. Agile: оценка и планирование проектов. Майк Кон
3. Пользовательские истории. Искусство гибкой разработки ПО. Джефф Паттон
4. Пользовательские истории. Гибкая разработка программного обеспечения. Кон Майк

Практическое задание

1. Изучить методический материал.
2. Изучить ТЗ.
3. Создать бэклог продукта: приоритизировать, оценить.
4. Создать бэклог спринта № 1: подготовить задачи, приоритизировать.
5. Оформить виртуальную доску в любой инструмент для организации задач таск-трекер, например trello, notion, youtrack и тп или просто excel таблица: создать командную доску под проект, внести бэклог продукта и бэклог спринта № 1.
6. Подготовиться к демонстрации и обсуждению.