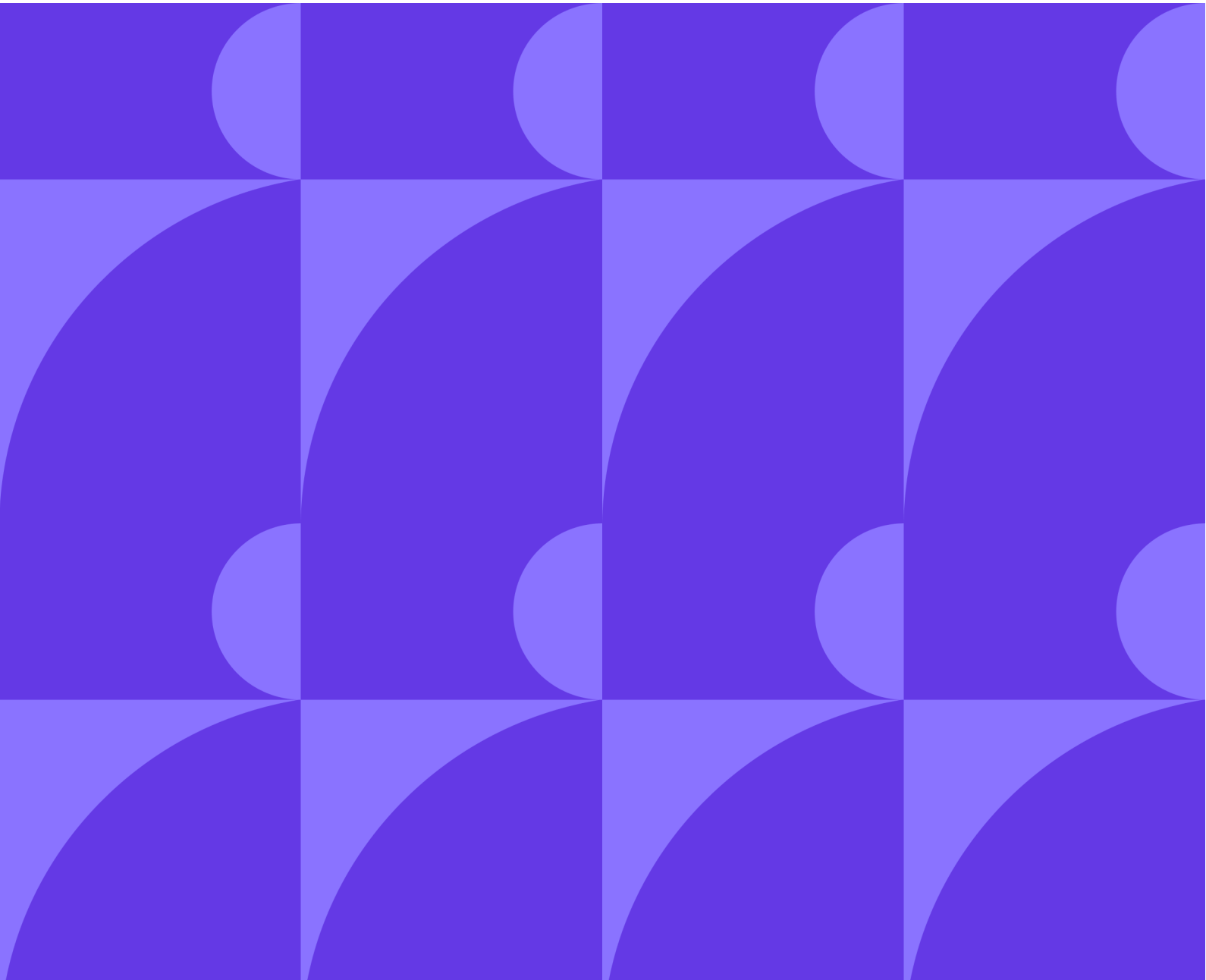
Гибкие методологии

Scrum



На этом уроке

1. Узнаем, что такое Scrum, и рассмотрим историю его появления.
2. Поговорим про роли, практики и ценности Scrum.
3. Обсудим, как строится работа команды по Scrum.
4. Узнаем, где можно применять Scrum, а где лучше не надо.

Оглавление

Глоссарий

Артефакт — продукт творческой деятельности. Документ, который появляется в результате умственной работы команды.

Бэклог — перечень рабочих задач, которые команда должна выполнить.

Инкремент — потенциально готовая к поставке часть продукта. Цель спринта.

IT (ИТ) — информационные технологии.

ПО — программное обеспечение.

Спринт — ограниченные по длительности итерации, в рамках которых выполняются задачи (обычно 1-3 недели).

Стендап — ежедневная встреча scrum-команды. Длится не больше 15 минут и обычно проходит у доски с задачами.

ТЗ — техническое задание

Фреймворк (от англ. framework — каркас, структура) — алгоритм, набор действий, правил, событий и инструментов для успешного выполнения какой-либо деятельности.

Теория урока

1. История возникновения Scrum
2. Связь Scrum и Agile
3. Что такое Scrum
4. Ценности Scrum
5. Принципы Scrum
6. Scrum-команда
7. Роли в Scrum-команде
8. Принципы работы Scrum-команды
9. Как устроена работа Scrum-команды или Практики Scrum
10. Артефакты Scrum
11. Частые ошибки
12. Преимущества и недостатки Scrum

История возникновения Scrum

Термин «scrum» заимствован из игры в регби. С английского переводится как «схватка» и описывает поведение команд перед вбросом мяча: после остановки игры или нарушения соперники обхватывают друг друга руками, создавая три линии игроков, пытающихся вывести мяч из схватки.

Применительно к разработке понятие Scrum впервые появилось в середине 80-х годов XX века в работах японских учёных Икуджиро Нонаки и Хиротаки Такеучи. Они говорили об успехе проектов, в разработке которых участвовали небольшие команды без жёсткой специализации.

Позже, в 1993 году американский программист Джеф Сазерленд применил этот подход, когда разрабатывал методологию для компании Easel. Об этом он детально рассказывает в своей книге «Scrum — революционный метод управления проектами». Там же он дал этому подходу официальное название.

Два года спустя разработчик и консультант по разработке ПО Кен Швабер формализовал методологию применительно ко всей IT-сфере.

В 1995 году на конференции «Объектно-ориентированные системы, языки и приложения для программирования» Кен Швабер указал, что основа Scrum — это итеративная разработка.

Несмотря на то что первоначально Scrum был рассчитан на разработку IT-продуктов, сегодня он применяется и в других областях.

Новички часто путаются, как Scrum связан с Agile. А некоторые и вовсе считают их одной и той же методологией. Давайте внесём ясность и поговорим про связь Scrum с Agile.

Связь Scrum и Agile

Scrum и Agile — понятия, которые идут рука об руку. Если не разобраться, что такое Agile, понять Scrum будет сложно.

Agile — это целая философия, семейство гибких подходов к разработке IT-продуктов, а Scrum — один из таких подходов, точнее — фреймворк.

Суть Agile заключается во взаимодействии команды и заказчика для создания работающего продукта, а также в готовности к изменениям.

Например:

- Вы работаете в большой команде и у вас одна цель — создать мобильную игру.
- Вы собрались, обсудили детали, изучили, какие есть коды, программы, специалисты, прочитали профессиональную литературу, подумали над дизайном и даже узнали много моделей организации разработки ПО.
- Вы решили, что для разработки подойдёт Agile. Но после более глубокого изучения поняли, что это лишь философия, которая вбирает в себя идеи, принципы и ценности.
- Вам непонятно, как это работает. Ясно, что по Agile вы должны быть гибкими и толерантными, но как этого достичь — неизвестно.
- На помощь приходит практический инструмент — Scrum — методология, которая описывает, какие действия нужно выполнить, чтобы получить качественный продукт, и как организовать работу команды по Agile-философии.

Согласно Agile-философии, при реализации проекта не стоит руководствоваться исключительно утверждёнными планами. Нужно ориентироваться на изменяющиеся

условия, учитывать обратную связь от заинтересованных лиц. Такие принципы мотивируют разработчиков к поиску уникальных решений, не ограниченных жёсткими стандартами.

Так вся команда меняет своё отношение к созданию итоговой ценности — продукта. При этом быстро достичь таких изменений без специальных инструментов не получится. Для этого используют Scrum.

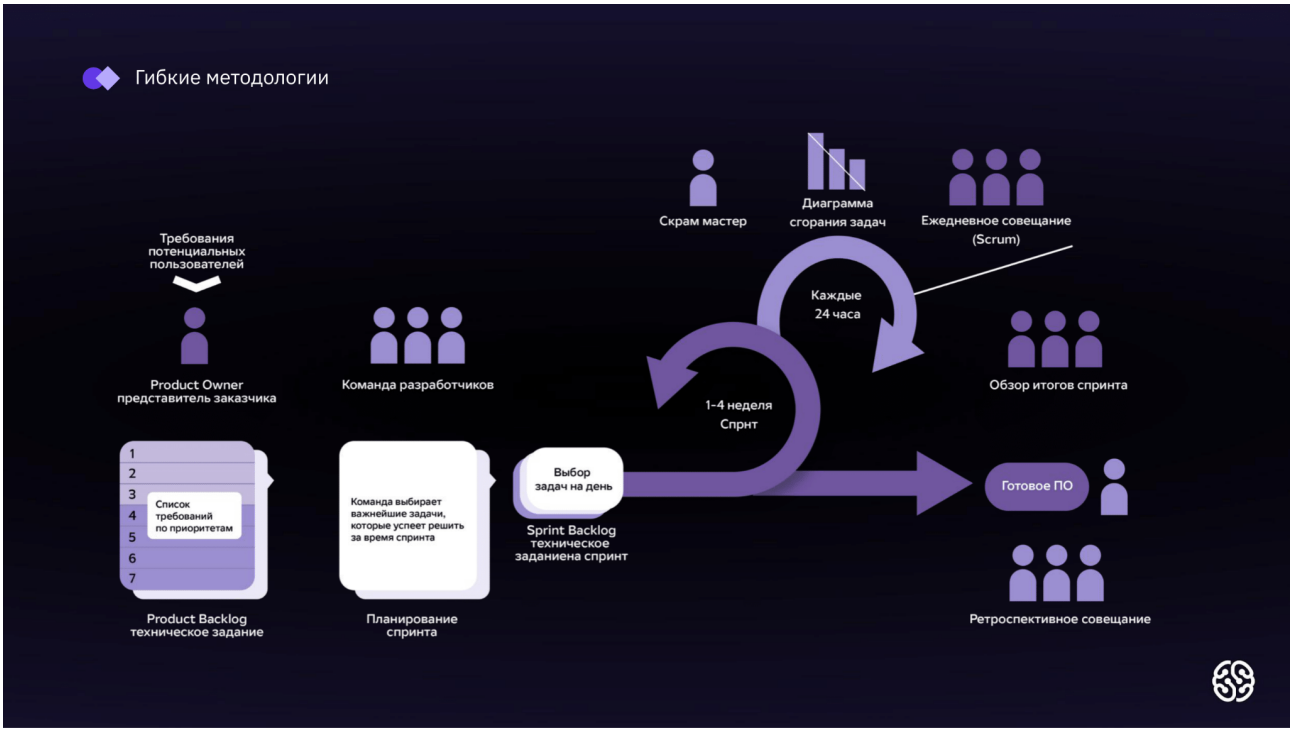
Что такое Scrum

Scrum — это набор принципов и инструментов, которые чаще всего применяют в IT-разработке. Scrum — это фреймворк.

Фреймворк (от англ. framework — каркас, структура) — алгоритм, набор действий, правил, событий и инструментов для успешной реализации какой-либо деятельности.

Фреймворк подробно описан в **Руководстве по Scrum (Scrum Guide 2020)**.

Спринт — короткий временной интервал, в течение которого команда выполняет определённый объём задач.



При планировании спринта определяют функционал и возможности IT-продукта. Так как спринты краткосрочны, можно предсказать ход разработки и обеспечить её гибкость.

Например:

Задача — создать сайт для крупного маркетплейса.

Для этого нужно:

- придумать архитектуру всего сервиса;
- написать серверную часть;
- сделать вёрстку и дизайн для мобильных приложений;
- выпустить сайт, который может делать то же самое, что и приложение;
- предусмотреть интеграцию с другими сервисами;
- сделать авторизацию через популярные соцсети.

Срок — 10 месяцев. В голове у команды начинается хаос. Мало кто понимает, за что нужно браться в первую очередь. Каждый специалист забирает на себя определённую задачу и начинает её выполнять. В итоге за пару месяцев до окончания срока выясняется, что эти части плохо интегрируются друг с другом или вовсе не работают вместе:

- сервер не отдаёт данные в нужном формате для веб-страницы,
- приложения хранят данные только у себя и не умеют отправлять их на сервер,
- авторизация в соцсетях работает только на Android.

⚠️ **Внимание!**

Чтобы такого не случилось, работы нужно разделить на спринты (небольшие отрезки времени, обычно 1-3 недели) и поставить цель для каждого.

Пример цели — сделать единую авторизацию и настроить синхронизацию между всеми устройствами пользователя. В течение спринта вся команда будет работать только над ней. А затем, спринт за спринтом, достигнет конечной цели.

В итоге команда работает слажено, весь объём задач распределяется на короткие промежутки времени, промежуточные цели чётко формулируются. Всё это помогает создать успешный работоспособный продукт.

Теперь погрузимся в суть использования фреймворка

Суть Scrum — в использовании инструментов для ускоренной разработки. Проще говоря, Scrum — это каркас, состоящий из множества типовых шаблонов (библиотек), которые можно дорабатывать. При работе над продуктом исполнитель не тратит время на создание элементарных вещей и может сосредоточиться на уникальных задачах.

Пример: бригаде рабочих нужно построить дом. Для строительства потребуются разные материалы: кирпич, арматура, балки, кровля и прочее. Если вести все работы с нуля, рабочим придётся самостоятельно делать кирпичи, отливать арматуру и создавать другие стройматериалы. Это долго и сложно.

Вместо этого строители используют готовые решения, то есть уже существующие материалы, и строят из них каркас: фундамент, стены, крышу. Получается почти готовый

дом. На этапе чистовой отделки рабочие уже могут экспериментировать, чтобы создать уникальный продукт.

То же происходит и при разработке ПО: разработчики не пишут код с нуля, а используют готовые библиотеки. А в конце работают с дизайном интерфейса.

В основе скрам-структуры непрекращающееся обучение и готовность приспосабливаться к изменяющимся факторам: изначально команда не знает, что будет делать, но знает, как это сделать. В процессе создания продукта она развивается и сразу применяет полученный опыт.

⚠️ **Внимание!**

Scrum — одна из реализаций принципов Agile. Если Agile — это абстрактные принципы и ценности, то Scrum — это конкретные инструкции, как воплотить в жизнь эти принципы и ценности.

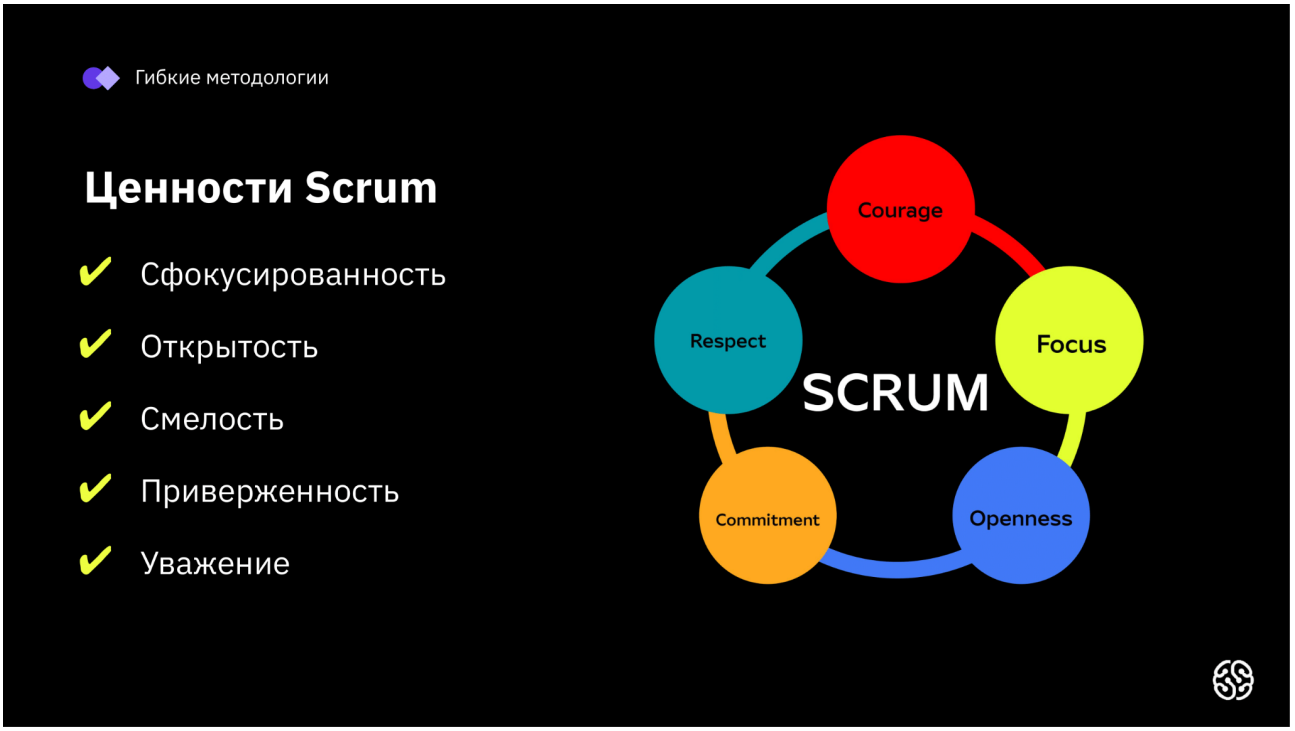
Важно понимать, Scrum — не пошаговая инструкция к действию. Фреймворк не описывает, **как именно нужно работать и какие решения принимать**. Scrum даёт комплекс базовых рекомендаций по организации процесса разработки и предполагает уникальное содержание любого проекта.

Scrum ориентируется не столько на процесс управления, сколько на процесс разработки продукта. Как показала практика, Scrum может как дополнить собой любую другую методологию управления, так и выступать в качестве самостоятельной методологии применительно ко всему процессу разработки продукта.

Чтобы успешно применить Scrum, важно разбираться в его составных элементах:

- ценности,
- роли,
- артефакты (элементы),
- практики (ритуалы, процессы).

Ценности Scrum



В основе Scrum пять основных ценностей:

- сфокусированность,
- открытость,
- смелость,
- приверженность,
- уважение.

Соблюдение и поддержание этих ценностей — основа для успешной слаженной работы команды.

Сфокусированность (Focus)

Команда фокусируется на выполнении ограниченного количества задач и на достижении цели спринта. Распыление на другие задачи снижает продуктивность и размывает точку внимания команды. Это снижает качество работы и уменьшает эффективность контроля.

⚠️ **Внимание!**
Scrum-команда должна фокусироваться на решении максимально актуальных и ценных для пользователя задач.

Открытость (Openness)

Открытость нужна, чтобы достичь гибкости. Члены команды должны вовремя обозначить и исправить проблемы в продукте или в процессе разработки, а не следовать первоначальному плану, если понятно, что он не приведёт к успешному результату.

⚠️ **Внимание!**
Члены команды могут открыто говорить о том, что считают важным.

Члены команды готовы использовать новые креативные решения, способы работы и взаимодействия.

Смелость (Courage)

Смелость позволяет сделать максимально прозрачным то, что происходит с командой и продуктом. Команда должна смело говорить о проблемах и препятствиях, чтобы устранить их в кратчайшие сроки. Важно, чтобы члены команды смело признавали ошибки и открыто о них говорили, сообщали об отсутствии навыков и просили о помощи или дополнительных ресурсах.

⚠️ **Внимание!**

Команда должна смело признавать:

- ошибки;
- отсутствие навыков;
- необходимость в дополнительных ресурсах. </aside>

Всё это помогает команде расти и развиваться.

Приверженность (Commitment)

Команда отвечает за свои слова и принимает ответственность за то, что обещано сделать. Контраст с классическим проектным управлением в том, что команда, вовлекаясь и осознанно давая обещание выполнить, несёт персональную ответственность за решение задачи и достижение цели спринта. В таком случае, команда, имея контроль над своими действиями и обещаниями, будет склонна к достижению успеха.

⚠️ **Внимание!**
Команда должна отвечать за свои слова и выполнять данные обещания в рамках каждого спринта.

Уважение (Respect)

Каждый член команды, его мнение, работа, ценности и личные границы уважаемы. Уважаются и все заинтересованные лица (заказчики, пользователи). Без уважения команда не сможет принимать обратную связь от заинтересованных сторон по проделанной работе и не сможет сформировать максимальную ценность для клиента.

⚠️ **Внимание!**
Важно уважать каждого члена команды и все заинтересованные стороны проекта, чтобы обратная связь была качественной, а ценность для заказчика — максимальной.

Принципы Scrum

- Работа короткими циклами (спринтами) — ****проект делится на части (спринты) и выполняется поэтапно. Спринт длится до того момента, когда работа над определённой частью продукта будет окончена.

- Гибкость — после окончания каждого спринта проводится тестирование. Если есть ошибки, меняется стратегия реализации или пересматривается список текущих задач (бэклог) по продукту.

- Пользователи и заказчик участвуют в разработке — ****в любой скрам-команде есть «владелец продукта» — заказчик или его представитель. Через него команда взаимодействует с пользователями, которые участвуют в тестировании проекта по окончанию каждого спринта. Владелец продукта собирает фидбэк, передаёт команде и на основе этих данных принимаются решения по дальнейшему развитию. Немного позже мы рассмотрим все эти роли и действия более подробно.

- Взаимодействие команды — ****скрам-команда — это несколько человек, которые взаимодействуют друг с другом и стремятся к достижению общей цели. Ориентация члена команды на свой результат неприемлема для гибких методологий.

Пример: вы работаете над разработкой мобильной игры. Чтобы получить качественный продукт, не делаете всё сразу, а разбиваете работу на короткие спринты, в рамках которых:

- создаёте дизайн,

- рисуете графику,

- интегрируете единую авторизацию всех устройств пользователя,

- пишите код,

- и много чего ещё.

По мере работы могут обнаруживаться ошибки в дизайне или коде, могут меняться требования заказчика. Вы гибко подходите к изменениям, вносите их и корректируете остальные составные части продукта под изменение.

Заказчик не просто ставит задачу и удаляется на период разработки, а участвует вместе с вами, подкидывает идеи, смотрит за процессом и сроком реализации, а потом даёт обратную связь по результатам.

При этом внутри команды нет авторитарного руководителя, есть скрам-мастер, так же участвующий в разработке. Он следит, чтобы вы всё успевали, создаёт приятный эмоциональный фон, анализирует список задач в спринте и помогает продвигаться в работе. В команде вы не просто наёмный сотрудник, который выполняет поставленные задачи, — вы можете влиять на процесс, придумывать и предлагать новые решения, а после обсуждения с командой и заказчиком внедрять их в продукт.

Чтобы реализовать все принципы Scrum и следовать ценностям, нужна команда — scrum-team. Давайте рассмотрим, из кого она состоит, и какие роли выполняют её участники.

Scrum-команда

Scrum-команда обычно состоит из 5-9 человек, редко из 3-4.

В рамках Scrum нет больших команд, потому что усложняется взаимодействие между звеньями, и работа становится менее эффективной.

В Scrum-команде есть 3 основные роли:

- Владелец продукта (Product Owner);
- Скрам-мастер (Scrum Master);
- Разработчики (Delivery Team).

Универсальная команда «солдат», которая работает над продуктом.

Роли в Scrum-команде

1. Владелец продукта (Product Owner)

Ответственный за разработку — заказчик продукта или его официальный представитель.

Владелец продукта:

- формирует видение продукта;
- отвечает за составление бизнес-плана, в котором отражается ожидаемый экономический эффект;
- определяет план развития, в котором для каждого пункта рассчитывается коэффициент окупаемости вложений;

- формирует список требований к продукту и сортирует его по важности (в порядке убывания);
- управляет ожиданиями заказчика и других заинтересованных лиц (стейкхолдеров);
- координирует и приоритизирует бэклог;
- предоставляет команде понятные и тестируемые требования;
- взаимодействует с командой проекта и заказчиком;
- принимает и оценивает результаты работы в конце каждого спринта.

Владелец продукта — центр принятия решений для проектной команды. Должен быть одним в рамках проекта, иначе принцип быстрого принятия важных решений нарушается.

2. Scrum-мастер

Отвечает за соблюдение фреймворка Scrum:

- контролирует инициативность и самостоятельность всех членов команды;
- следит за удовлетворенностью команды получаемыми результатами;
- налаживает атмосферу в коллективе;
- контролирует итоги работы команды в целом;
- обозначает проблемы и открытые вопросы;
- отвечает за взаимодействие всех членов команды между собой;
- участвует во всех встречах;
- поддерживает работоспособность;
- устраняет проблемы;
- следит за следованием графику работ.

Перед спринтом скрам-мастер вместе с командой определяет задачи, которые нужно решить, а в конце спринта — подводит итоги.

С помощью спринтов скрам-мастеру легко контролировать эффективность команды, находить недочёты в работе и определять способы мотивации её участников.

⚠️ **Внимание!**
Скрам-мастер — не обособленное лицо, наблюдающее за разработкой со стороны. Это координатор действий разработчиков. Он не только контролирует, но и принимает участие в технической реализации IT-продукта.

3. Разработчики

Отвечают за разработку IT-продукта. В одну команду входит 5-9 разработчиков.

Основные задачи:

1. Постановка реально достижимых, прогнозируемых, интересных и значимых целей для каждого спринта.
2. Достижение поставленных целей в установленные сроки (дедлайн).

⚠️ **Внимание!**
Достижение цели — понятие растяжимое, его нужно конкретизировать в зависимости от специфики проекта. Например, где-то цель считают достигнутой после написания всех кодов, а где-то нужно ещё и успешно пройти тестирование.

При оценке достижения цели разработчики руководствуются собственным видением и опытом.

Ключевые навыки для разработчиков в скрам-команде:

- умение планировать,
- объективная оценка выполненной работы,
- умение взаимодействовать с другими членами коллектива.

Разработчики занимаются:

- оценкой задач в бэклоге,
- разработкой продукта,
- отслеживанием прогресса разработки вместе со скрам-мастером,
- предоставлением результата владельцу продукта.

⚠️ **Внимание!**

Разработчики — это не только программисты, которые пишут код, но и маркетологи, верстальщики и другие специалисты, нужные для создания конкретного продукта.

Принципы работы скрам-команды

- Постоянное самосовершенствование — совершенствование продукта невозможно без самосовершенствования каждого члена команды.

- Автономность — все члены команды должны уметь работать не только в команде, но и автономно, индивидуально.

- Кросс-функциональность — любая команда самодостаточна, так как в неё входят специалисты с разными навыками и опытом. Члены команды могут развиваться и перенимать навыки своих коллег.

Как устроена работа скрам-команды или практики Scrum

Работа команды делится на этапы:

1. Планирование списка задач спринта.

2. Проведение регулярных совещаний.

3. Организация скрам-доски.

4. Изменение планов в ходе итерации.

5. Аварийная остановка спринта.

6. Подведение итогов спринта.

7. Проведение ретроспективы спринта.

Рассмотрим подробно, что происходит на каждом этапе.

1. Планирование списка задач спринта

Каждый спринт начинается с планирования. Все члены команды собираются, оценивают бэклог и выбирают приоритетные задачи, которые нужно выполнить в рамках текущей итерации. Так формируется список задач (бэклог) текущего спринта.

Затем на основании бэклога обсуждается объём работ и сроки спринта.

Важно уже на этом этапе определить ожидаемый результат — что должно получиться по итогам спринта.

2. Проведение регулярных совещаний

Ежедневно или еженедельно команда проводит короткие совещания (15-30 минут) — стендапы. Другое название — ежедневные скрам-встречи (Daily Scrum Meeting).

В таких совещания участвуют все члены команды: владелец продукта, скрам-мастер и все разработчики. Каждый член команды должен ответить на три вопроса:

1. Что сделано с прошлой встречи?
2. Какие планы на сегодня?
3. Есть ли препятствия к достижению цели?

Скрам-мастер в ходе совещания выявляет текущие проблемы и помогает команде решить их.

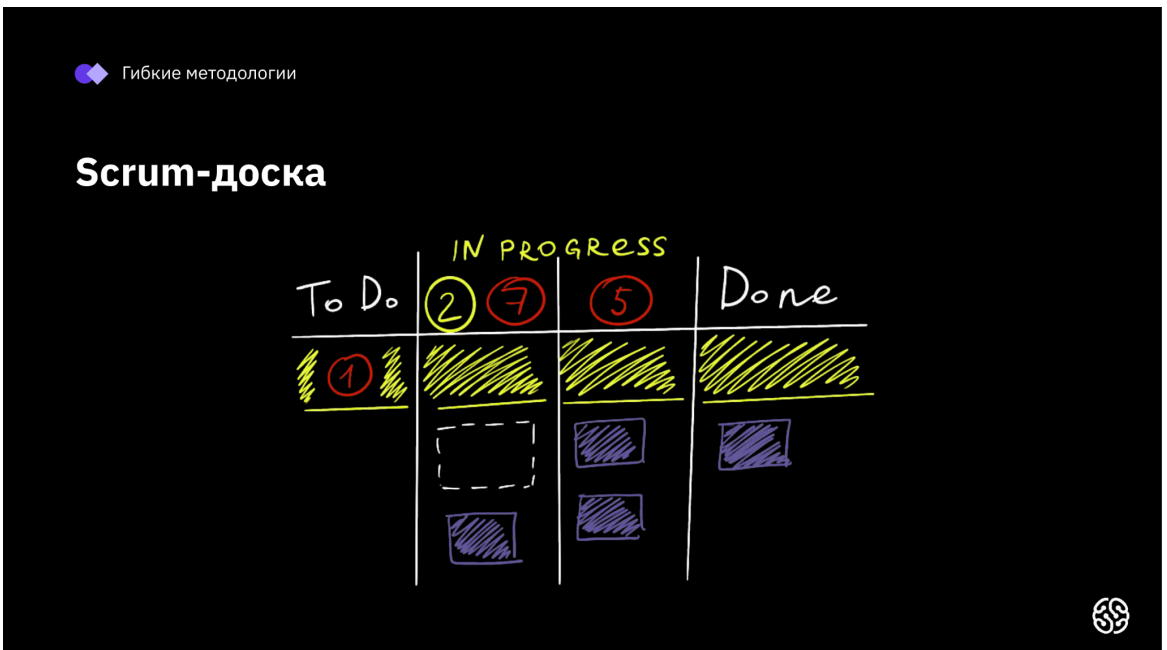
Также проводятся встречи по обзору спринта (Sprint Review Meeting). Они похожи на стендапы, но отличаются большим погружением в весь процесс выполнения работ.

3. Организация скрам-доски

В переговорной комнате, где проводятся регулярные стендапы, вешается доска, разделённая на три части:

- что нужно сделать;
- в работе;
- сделано.

В каждой части размещаются стикеры разных цветов с основными задачами. По мере выполнения они перемещаются от одной части к другой. Это помогает отслеживать прогресс текущего спринта каждому члену команды.



4. Изменение планов в ходе итерации

Изменения планов — одна из особенностей работы по Scrum. Команда должна быть открытой, и если один специалист не успевает уложиться в срок, он сообщает об этом владельцу продукта. Владелец продукта меняет расстановку задач, оптимизирует рабочее время и помогает специалисту уложиться в сроки.

То же происходит, когда задачи выполняются до запланированного срока. Значит, была допущена ошибка в планировании. Владелец продукта дополняет бэклог новыми целями по собственному усмотрению, чтобы реализация всего продукта протекала быстрее. Речь не о том, чтобы нагрузить членов команды какими-то работами и занять свободное время. Это делается, чтобы команда за один спринт смогла продвинуться дальше в разработке и быстрее получить конечный продукт.

5. Аварийная остановка спринта (Sprint Abnormal Termination)

Используется только в особых случаях. Команда может остановить спринт до наступления дедлайна, если осознает, что добиться поставленных в этом спринте результатов не получается. Также спринт может остановить владелец продукта, если необходимости в достижении цели спринта больше нет.

Если спринт остановлен, все участники проекта собираются на общей встрече, обсуждают причины остановки и дальнейшие действия. После этого принимают решение начать и планировать новый спринт.

Пример: крупный банк заказал вам разработку мобильного приложения. Вы разбили проект на месячные спринты. На одном из них нужно разработать интерфейс мобильного приложения.

Составлен бэклог спринта, команда приступает к разработке интерфейса.

Через неделю становится известно, что у заказчика сменился руководитель, и он видит концепцию интерфейса иначе. Большая часть требований меняется, поставленные задачи теряют актуальность.

В этом случае нет смысла ждать окончания спринта. Нужно сделать аварийную остановку и кардинально поменять бэклог.

6. Подведение итогов

После окончания спринта команда тестирует выполненную часть продукта. В таком тестировании принимают участие потенциальные потребители (фокус-группа) — либо конечные пользователи, либо специалисты со стороны заказчика. Владелец продукта собирает обратную связь, принимает решения для дальнейшей работы и для формирования задач на следующий спринт.

И так вплоть до завершения всей разработки и получения конечного продукта.

7. Ретроспектива спринта

Мероприятие для обзора завершённых этапов. Команда записывает результаты, обсуждает нюансы спринта и сопутствующих процессов.

Основная задача — привлечение внимания команды к тому, что получилось и что можно попытаться улучшить в следующий раз. При этом нет цели акцентировать ошибки.

Такой подход помогает команде использовать ценный опыт из прошлого спринта и учитывать сделанные выводы в будущем.

Артефакты Scrum

На протяжении всей работы команды появляются определённые документы. В Scrum они называются артефактами. Основные среди них:

- журнал продукта (Product Backlog),
- журнал спринта (Sprint Backlog),
- график спринта (Burndown Chart).

Журнал продукта (Product Backlog)

Журнал продукта (бэклог) в начале разработки готовит владелец продукта. Этот артефакт содержит требования, отсортированные по значимости. Первичную версию артефакта дополняют разработчики — они оценивают стоимость реализации каждого требования.

Бэклог продукта должен включать технические и функциональные аспекты разработки.

Для каждого требования определяется приоритет (например, от 1 до 5). Самые приоритетные описываются детально, чтобы команда смогла оценить их и протестировать.

Владелец продукта обязан не только подготовить бэклог, но и предоставить его команде в срок, иначе не получится своевременно реализовать продукт.

Журнал спринта (Sprint Backlog)

Задачи для реализации в текущем спринте. Составлением занимается скрам-мастер с командой.

⚠️ **Внимание!**
Грамотная разбивка спринта на задачи позволяет к концу итерации выполнить всё необходимое, чтобы часть ПО работала корректно и представляла ценность для конечного потребителя.

После составления бэклога спринта его оценивает команда разработчиков и сопоставляет с журналом продукта. Если есть существенные отклонения, коллектив определяет приоритетные задачи в рамках текущего спринта и менее важные, которые можно перенести в следующий спринт.

При этом перед владельцем продукта возникает задача исключить из бэклога спринта мелкие и незначительные задачи, выполнение которых никак не повлияет на результат работы.

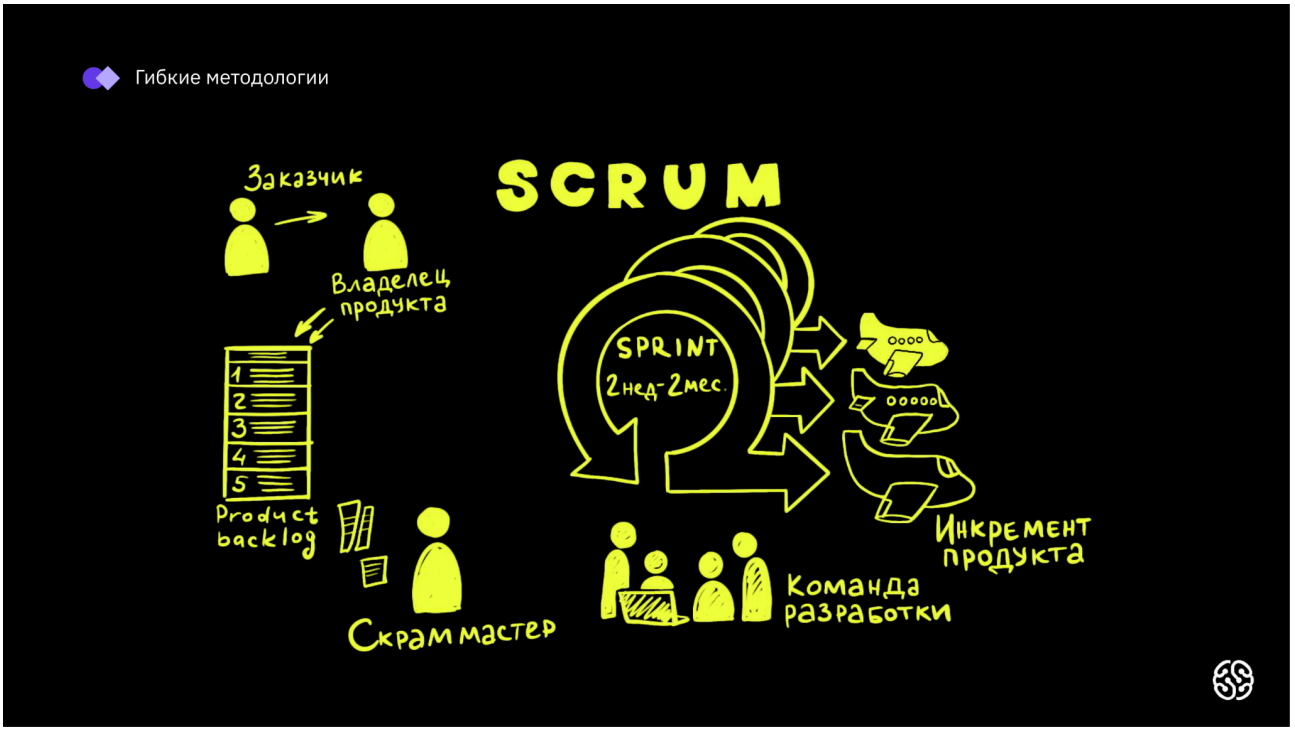
График спринта

Расписанный задачами календарь работы в рамках одного спринта. Команда регулярно оценивает график и при необходимости реагирует на изменения.

⚠️ **Внимание!**
С помощью графика спринта владелец продукта может оценить скорость работы и соблюдение дедлайнов. Если со временем объём работы не уменьшается, значит в процессе есть отклонения или ошибки в планировании. Нужно срочно скорректировать действия команды.

Инкремент

Инкремент, а точнее «потенциально готовый к поставке инкремент» — часть готового продукта. Объединяет реализации элементов бэклога продукта, сделанные во время текущего спринта. Каждый спринт должен включать минимум один инкремент, чтобы считаться завершённым успешно.



Содержание и характеристики инкремента:

- Инкремент — конкретная ступень к достижению цели продукта.
- Чтобы у инкремента была ценность, он должен быть пригодным для использования.
- Каждый инкремент — дополнение ко всем предыдущим. Они тщательно проверяются для совместной работы всех инкрементов.
- В рамках одного спринта можно создать несколько инкрементов.

Возьмём в качестве примера создание сайта для маркетплейса. Вы разбили проект на спринты. В конце каждого спринта стоит цель, чтобы можно было оценить, насколько успешно вы продвинулись в разработке.

Допустим, первый спринт длится месяц. По завершении команда должна достичь цели — получить инкремент в виде выстроенной архитектуры сервиса.

⚠️ **Внимание!**
Инкремент — это цель спринта. Часто под инкрементом подразумевают критерии готовности. Причём это может относиться к определённой контрольной точке, цели отдельного спринта либо полноценной версии продукта, готовой к использованию.

Частые ошибки

1. Ошибки артефактов

Одна из самых крупных ошибок команд, совмещающих Scrum с каскадной моделью, — они превращают выполнение спринтов в бесконечный марафон. Погоня за максимальной продуктивностью без перерывов между спринтами быстро изматывает. Настоящие Scrum-команды пользуются каждым удобным случаем отпраздновать свои достижения, успехи клиента, поощрить тех, кто превзошёл ожидаемые результаты, оглянуться и отметить, насколько выросла команда за прошедший период. Это может быть как пышное торжество в честь знакового события, так и совместный обед в конце спринтов. Важно выделять время на небольшие перерывы для отдыха и рефлексии, потому что они заряжают команду энергией и энтузиазмом для следующего спринта.

2. Скрам-мастер = менеджер проекта

В Agile-методологии команды используют стендапы для синхронизации. Перед началом рабочего дня (или в другой момент) команды собираются у доски, чтобы обсудить задачи текущего и предыдущих дней. В плохом сценарии скрам-мастер проверяет статус и назначает на исполнителя новую задачу. Это не даёт команде самоорганизоваться. Скрам-мастер действует как менеджер. По Scrum он должен вернуть вопрос в команду, чтобы они сами спросили: «Что должно быть дальше?»

3. Скрам-мастер принимает решения за команду

Есть случаи, когда команда полагается на скрам-мастера в принятии каждого решения, что абсолютно неправильно и противоречит самоорганизации. Коллеги доверяют практическому опыту скрам-мастера, и вместо генерации своих идей и решений ждут ответа от этой роли.

Скрам-мастер пытается показать свою значимость и вместо коучинга, консультаций и советов берёт на себя роль принимающего решения. Это неправильно.

Избегайте диктатуры, так как она влияет на командный дух, душит прогресс и возможности для дискуссии, общения, а в итоге для самоорганизации. Скрам-мастер должен построить команду так, чтобы она функционировала без него.

4. Скрам-мастер работает на DONE, а не на качество

Есть распространённая проблема в применениях Scrum, когда от команд требуют больше выполненных задач в ущерб качеству.

Скрам-мастеру дают метрику задач в производство для измерения эффективности его работы. Вместо исправления процессов и мотивирования команд, он концентрируется на доставке продукта.

Чтобы избежать этого, прекратите планировать задачи индивидуально. Планирование осуществляется всеми членами команды коллективно. Нужно, чтобы каждый член команды придерживался обязательств, понимал цель и способы её достижения.

5. Скрам-мастер — прокси (передатчик) между командой и владельцем продукта

Допустим, команда сталкивается со сложностями во время разработки, и нужно получить новую информацию от владельца продукта. Тут скрам-мастер связывается с владельцем продукта, передаёт информацию команде.

Зачем в этом процессе нужен скрам-мастер? Чтобы ощутить собственную значимость, снять негатив или зачем-то ещё?

Нужно решить корневую проблему и перестать работать передатчиком.

6. Скрам-мастер позволяет нарушать правила

Есть множество аргументов у незрелых команд и организаций, почему не нужно ревью, стендап или ретроспектива. Скрам-мастер не должен нарушать фреймворк и позволят это делать своей команде. Работайте над качеством событий, фасилитацией и другими препятствиями, но не отменяйте скрам-церемонии.

То же касается всех обязательных в Scrum артефактов. Без статистики, которую ведёт команда каждый спринт, путь к совершенствованию будет затруднён.

7. Скрам-мастер не работает с владельцем продукта

В силу разных причин работа скрам-мастера и владельца продукта может быть осложнена. Это воспринимается как данность. В итоге бэклог не подготовлен, задачи собраны по случайным признакам, а приоритеты меняются изо дня в день.

Помочь владельцу подготовить бэклог — это ответственность скрам-мастера. Установите цели для владельца продукта и работайте с ним несмотря на то что это будет занимать много времени.

Компаниям важно правильно применять Agile для создания продуктов и быстрого получения ценности. Если вы скрам-мастер, не избегайте решения вышеупомянутых проблем со Scrum, чтобы не упустить возможность построить действительно великолепную команду.

8. Отсутствие ежедневных собраний

Ежедневные собрания — важная часть Scrum. Благодаря таким встречам, участники команды понимают, кто над чем работает. Кроме того, собрания помогают сплотить коллектив и наладить сотрудничество между его участниками.

9. Общение через скрам-мастера

Некоторые команды общаются только через скрам-мастера вместо того, чтобы обратиться напрямую к другому участнику команды. Необходимо демонстрировать культуру взаимной поддержки и плодотворного общения, а участники команды должны научиться общаться тет-а-тет и обращаться друг к другу напрямую. Такой подход формирует правильную атмосферу в коллективе и делает его более продуктивным.

10. Отсутствие ретроспективы после спринта

Зачастую команды пропускают проведение ретроспективы, либо проводят мероприятие только если на это есть время. Но Agile строится на постоянных корректировках и улучшениях, выявить которые позволяют именно такие встречи. Важно вовремя остановиться, оглянуться вокруг, найти что улучшить и начать работать над следующим спринтом.

Преимущества и недостатки Scrum

Преимущества Scrum

Scrum имеет ряд преимуществ как для команды, так и для компании в целом:

- команда работает короткими этапами, на каждом из которых определяет цели и пути их достижения, что ускоряет процесс работы;
- команда работает над разными задачами проекта одновременно, что позволяет быстрее достичь желаемой цели;
- большие задачи разделяют на мелкие, поэтому внести корректировки прямо в процессе работы намного проще, чем в каскадном подходе;
- сокращается время на поиск ошибок и объяснение проблем;
- минимизация финансовых рисков благодаря оперативной реакции на изменения и устранение ошибок;
- каждый член команды чётко знает свою задачу, повышается уровень ответственности;
- обмен информацией открытый, что делает рабочий процесс максимально прозрачным;
- поддержание высокого уровня мотивации в команде благодаря ежедневной видимости достижений.

⚠️ **Внимание!**
Scrum — это win-win подход. Обеспечивает взаимодействие команды и заказчика, при котором каждая сторона остаётся в выигрыше и получает желаемое.

Недостатки Scrum

Несмотря на явные преимущества, у Scrum есть и недостатки:

- успех проекта во многом зависит от скрам-мастера, квалификации команды и их приверженности своему делу;

- далеко не всегда можно адаптировать Scrum под сферу деятельности, поскольку есть проекты, требующие исключительно планового подхода в работе;
- требует регулярной коммуникации с заказчиком, что порой тормозит процесс из-за невозможности получения обратной связи;
- сложность внедрения в масштабных и сложных проектах;
- Scrum больше подходит для малых и средних проектов.

Используемые источники

1. Джефф Сазерленд «Scrum. Революционный метод управления проектами»
2. Scrum Guide 2020
3. Основы Scrum – Scrum Guide на русском
4. SCRUM: Понимание и применение фреймворка
5. Руководство по скраму (Scrum) — К.Швабер и Дж.Сазерленд