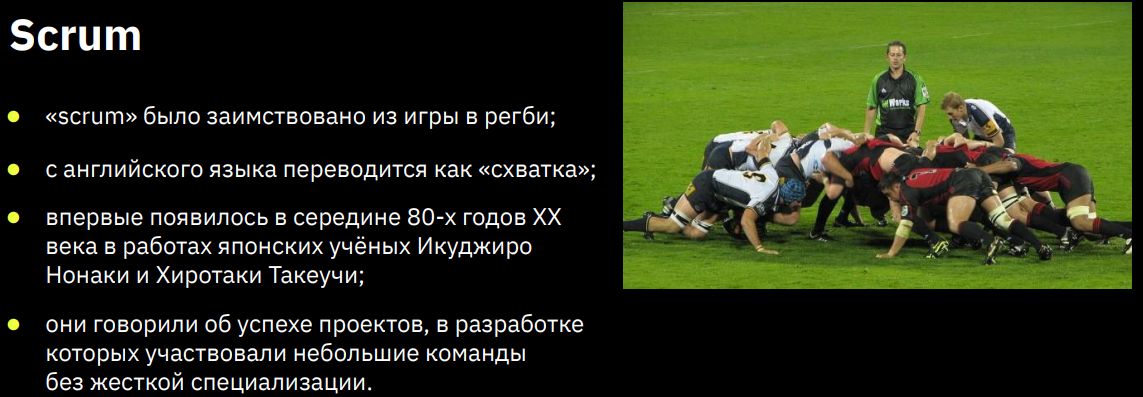
**Лекция № 3. Scrum**

Про что мы будем сегодня говорить, во-первых, поговорим про историю возникновения scrum, поговорим, насколько она взаимосвязана с agile. Что такое agile? Что такое scrum? Как происходит разработка it-продукта с помощью данной методологии? Какие включает в себя данная методология ценности, подходы? Какие есть роли внутри команды, которая ведёт разработку с помощью scrum. Какие есть преимущества, недостатки? И в целом обсудим различные ошибки, которые очень часто встречаются, когда команда только-только переходит к данной методологии и начинает её применять.



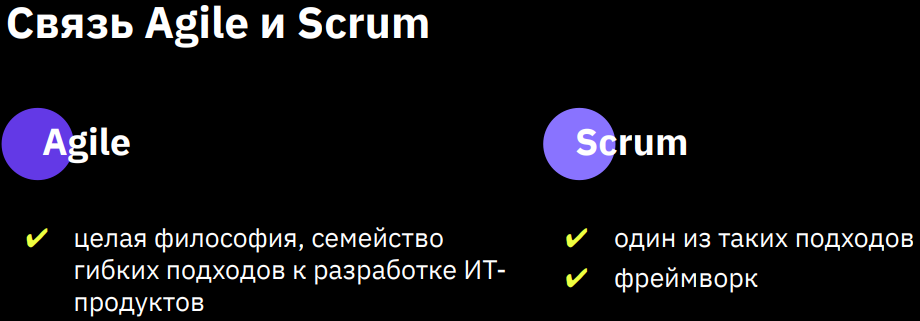
Для начала история возникновения откуда появилось слово scrum. Слово scrum переводится с английского языка как «схватка», появилось это понятие из игры регби. Первый раз его начали использовать в середине семидесятых годов XX века двое японских программистов, два японских учёных, которые активно занимались разработкой различного программного обеспечения это у нас Икуджиро Нонахи и Херотаки Тaкeучи, затем плавно начали применять этот подход и другие программисты,



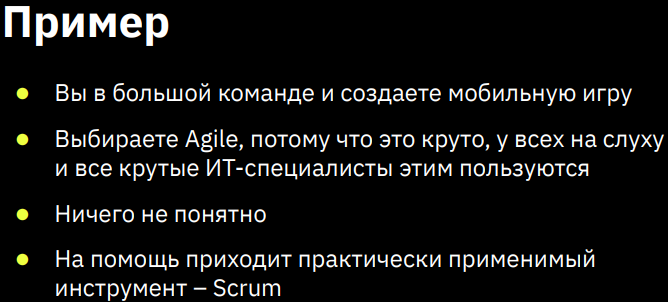
одним из первых был Джев Сазерленд в 1993 году он разрабатывал методологию по управлению it-проектами для одной из компаний Izel и заметил, что чем больше проект, тем сложнее его сделать. Это, казалось бы, очевидно, поэтому большого слона нужно поделить на небольшие маленькие части и выполнять все работы на проекте, итерации. Соответственно, он применил такой подход, немножко его модернизировал, дал ему определённое название scrum. И написал целую книгу, которая называется «Scrum. Революционный подход к управлению it-проектами»,



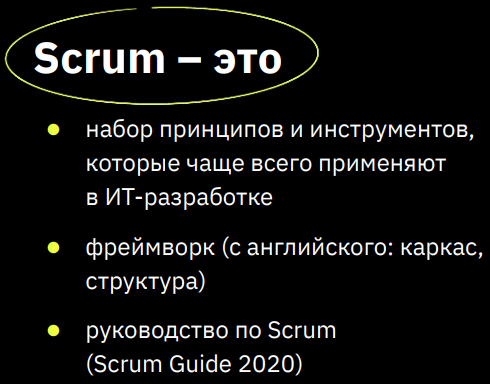
второй американский учёный программист, разработчик программного обеспечения Кен Швайбер, также начал применять этот подход, тоже внёс кое-какие корректировки и формализовал данную методологию, под названием scrum, под управление всеми этими проектами в целом для всей it-сферы. Можно сказать, что эти два человека и явились про-родителями уже того scrum, который мы видим сейчас, который используется при разработке продуктов.



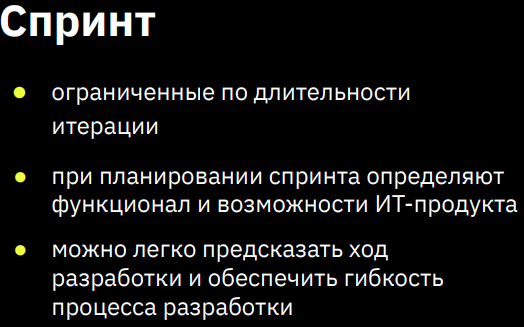
Давайте посмотрим, как взаимосвязаны между собой scrum и agile? И что такое scrum? Если мы говорим про agile, то это целая философия, которая включает в себя определённые принципы, ценности и в целом является более таким верхне-уровневым пониманием, как правильно разрабатывать it-продукты с помощью гибкого подхода, если мы говорим просто scrum, то это уже целый фреймворк, который практически применим при разработке. Конечно, это всего лишь слова.



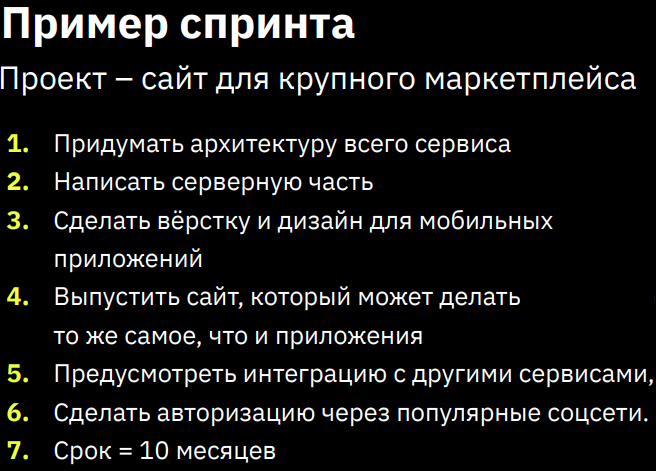
Давайте посмотрим на примере, как это выглядит. Вы работаете в большой команде, и у вас есть один большой проект — это создание мобильной игры, вы определили верхне-уровневые технические требования к своей игре, понимаете как она должна выглядеть. Какой должен быть дизайн? Что это такое? Затем вы вырабатываете определенную методологию с помощью, которой будете разрабатывать её. К примеру, вы решили взять agile, применить данный гибкий подход, сначала ничего непонятно, вроде есть какие-то ценности, принципы, понятия, общие философские наработки. Ну как это всё применить при разработке? Вот здесь как раз-таки приходят на помощь практически применимый инструмент это scrum.



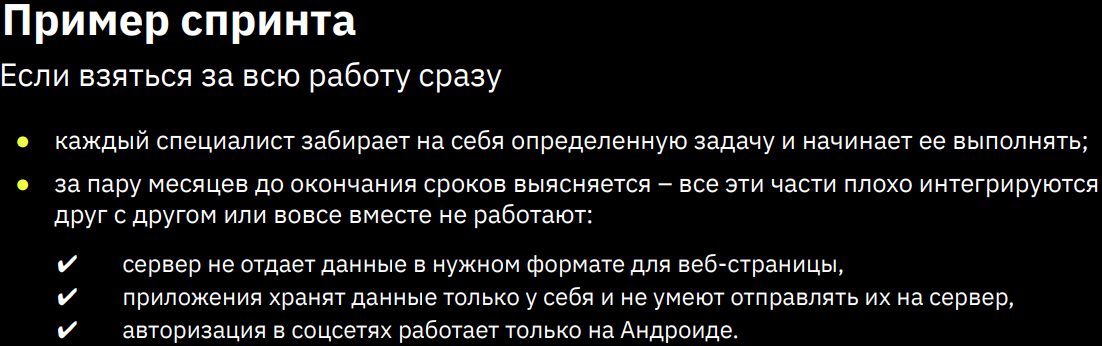
Что такое scrum? Давайте посмотрим детально — это набор принципов, подходов, при разработке, по сути дела это целый фреймворк, с английского языка переводится как «структура». Даже есть целый документ, который называется «Руководство scrum гайд 2020», в которой достаточно детально подробно расписывает. Что такое scrum? Из чего он состоит? И как можно построить scrum, а также как можно построить работу команды с помощью данной гибкой методологии? Scrum предполагает работу спринтами.



Что такое спринты? Спринты — это ограниченные во времени итерации, обычно длятся от 1 до 3 недель — это комфортное время для того, чтобы выполнить определённый набор задач. При планировании спринта, обычно определяются технические требования к нашему it-продукту и основные задачи, которые нам позволят добиться выполнить эти самые технические требования, реализовать их на практике. Когда мы делим наш большой проект на небольшие участки, грубо говоря делим большого слона на маленькие части, нам достаточно легко контролировать разработку и в случае, если возникают какие-то ошибки, быстро их устранять, при таком подходе стоимость оценки нашей ошибки резко снижается.



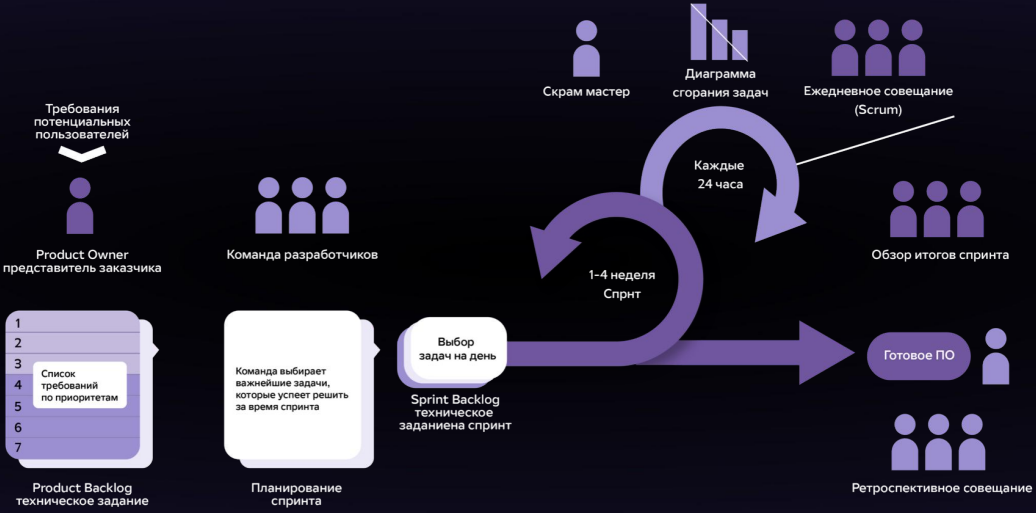
Пример спринта, давайте посмотрим, как это выглядит в жизни? Допустим у вас есть проект, это создание веб-сайт для маркетплейса. Вам необходимо придумать архитектуру для вашего сайта, написать код, отрисовать дизайн, сверстать его, выпустить сайт, провести рекламные акции, чтобы он стал популярен среди ваших потребителей, предусмотреть интеграцию вашего сайта с другими сервисами, выпустить какое-то мобильное приложение для Android, для iOS и на всё на это у вас 10 месяцев.



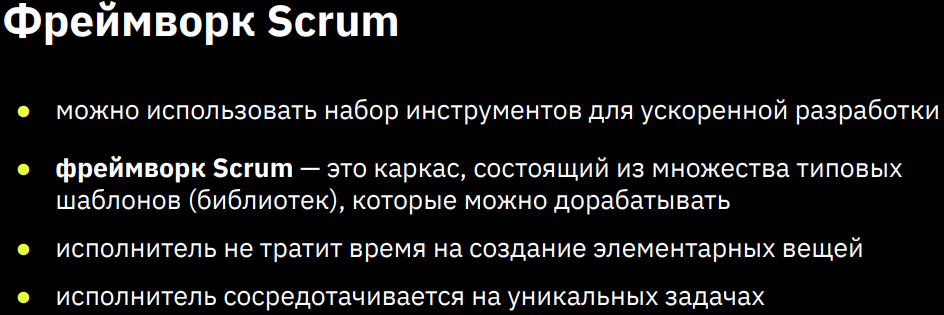
Если взяться за всю работу сразу, то есть, если взять большой масштабный проект, как это будет выглядеть? Конечно, каждый специалист, возьмёт себе в проработку определённые задачи, определённой работы, начнёт их выполнять. Затем, когда эти специалисты закончат выполнять свою работу, может выйти следующее, что их результат работы не интегрируется друг с другом, не взаимосвязан. Допустим, возможно сервер не умеет хранить данные, либо ваша программа не может отправлять эти данные на сервер, либо не интегрированный какие-то социальные сети, в целом что-то не работает, что-то не стыкуется? И это всё будет известно, только в самом конце, когда все ваши специалисты закончат свою работу и получат какой-то небольшой промежуточный результат. Обычно по практике это происходит прямо перед завершением проекта, когда начинают состыковать самый последний день, и всё летит к чертям. Как это всё сделать с помощью scrum, с помощью гибкого подхода?



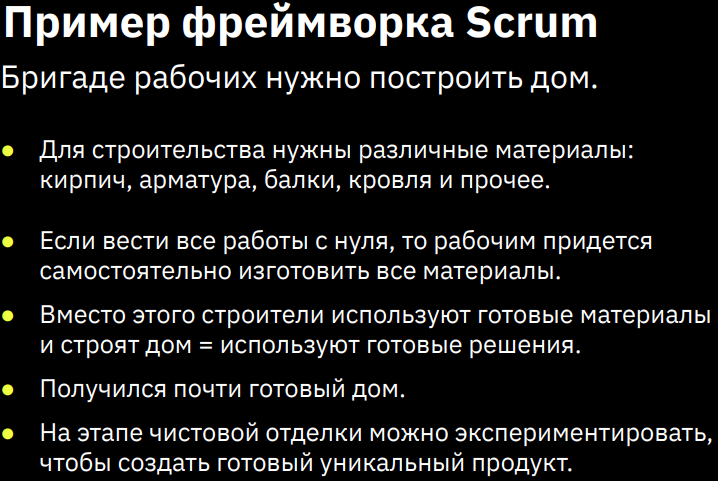
Мы, делим весь наш проект, на небольшие работы, на так называемые спринты, которые длятся от 1 до 3 недель. Затем в рамках этих трёх недель, мы должны выполнить определённый набор задач, это называется backlog нашего спринта, то есть список задач, которые мы выполняем за эти три недели. На первый спринт обязательно нужно поставить цель, впрочем, как и на последующие, это ещё называется инкремент спринта, поэтому поговорим чуть позже. В течение спринта команда ваша работает, только над данной целью, в конце вы получаете результат. И точно также, спринт за принтом двигаетесь к своей итоговой цели, к созданию сайта для маркетплейса.



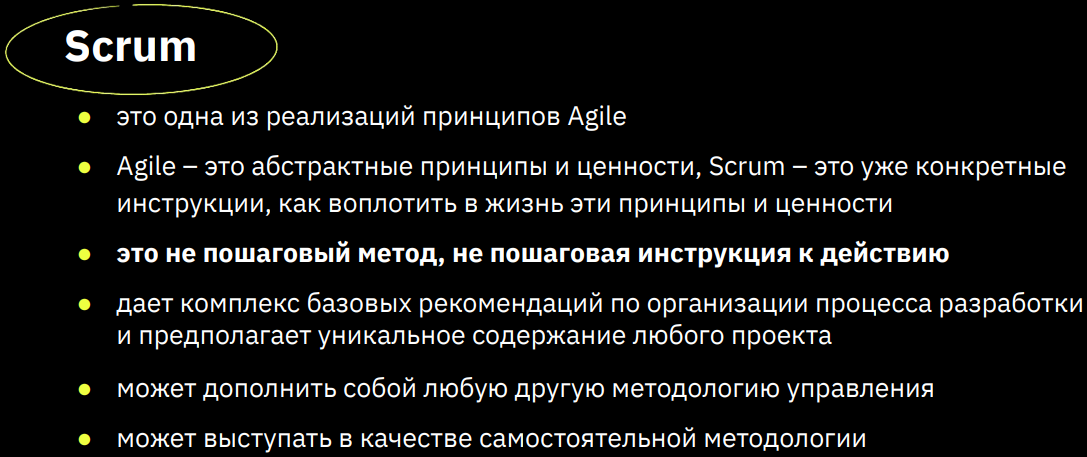
Сейчас на данном слайде вы увидите, как в целом построена работа нашей команды, по нашей гибкой методологии scrum. В самом начале определяются верхне-уровневые требования к it-продукту, составляется перечень необходимых задач. Затем из этого перечня выбираются первостепенные, первоочередные задачи, включаются первый спринт и так далее, во второй, в третьей, четвёртый, пока мы не закончим всю разработку, в рамках данного спринта, которое обычно длится от 1 до 3 недель, команда работает над выполнением этих задач ежедневно синхронизируется и после завершения спринта проводят определённое review, смотрят насколько они успешно достигли цели, что было сделано правильно, а что нет. Давайте будем рассматривать подробно, с примерами, как это всё происходит?



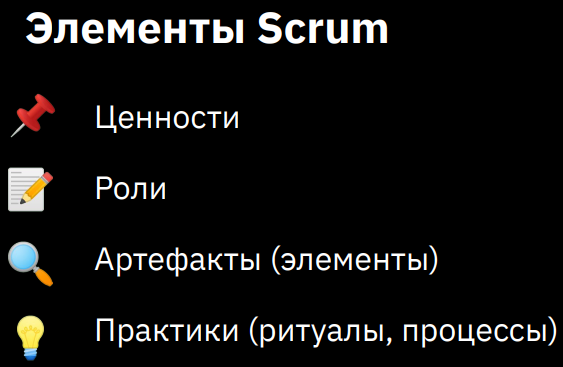
Фреймворк scrum. Что это такое? Фреймворк — это целый набор, практически применимых инструментов, то есть такая своеобразная библиотека, которая содержит в себе шаблоны, зачем это нужно? Во-первых, это позволяет нашим исполнителям не тратить лишнее время на создание чего-то заново, на создание того, что уже было создано, помещено в библиотеку, его можно использовать для своей работы, также это позволяет сконцентрироваться на основных задачах, первостепенных, который позволит достигнуть цели и разработать конечный продукт.



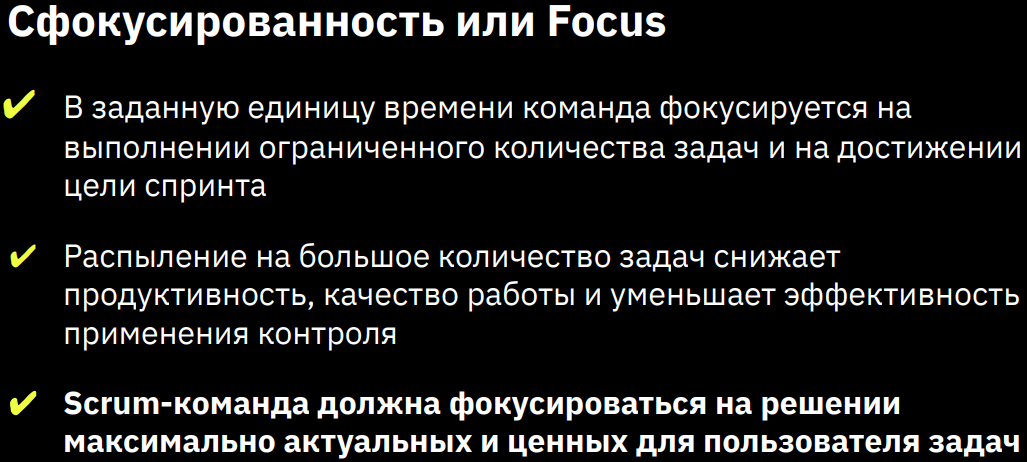
Давайте посмотрим на примере, как это выглядит, допустим у вас есть проект, построить дом, бригаде для того, чтобы построить дом, нужно будет различный материал, например, балки, кирпичи, шпаклёвка и так далее, если бы все работы выполнялись с нуля, то рабочим пришлось бы все эти материалы самостоятельно производить, делать кирпичи, создавать бетон, делать арматуру, и прочие различные материалы и инструменты, которые пригодятся при стройке. Но у нас уже есть уже готовые строительные материалы, которые мы можем использовать в нашей работе, то есть, те самые библиотеки, те самые шаблоны. Мы их берём, применяем, строим дом, и в конце, когда у нас уже полностью выстроена коробка, мы завершаем работы с помощью отделки. Вот там, мы уже можем экспериментировать, работать с нашим дизайном, что-то улучшать, креативить и так далее. Точно также строится разработка и любого ПО с помощью данной методологии.



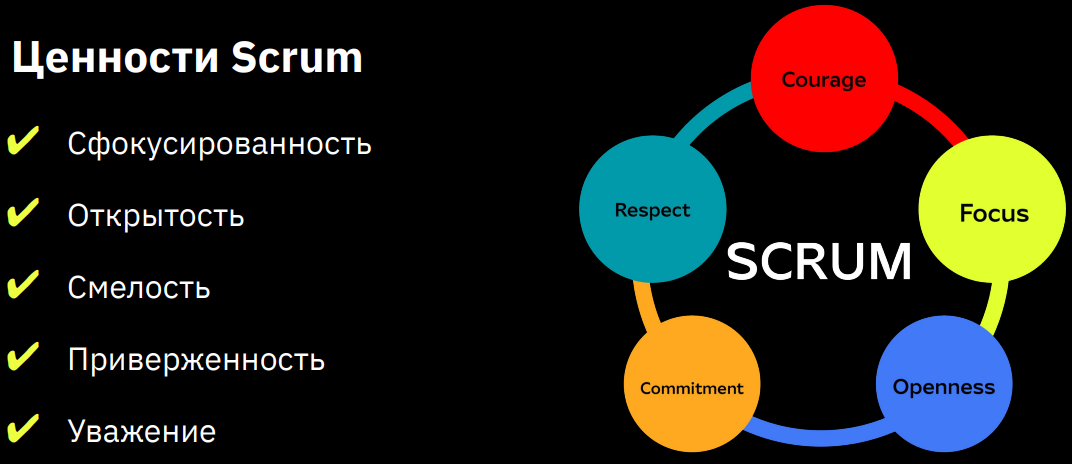
Важно отметить, что scrum у нас — это одна из реализаций принципов agile гибкого подхода. Agile – это целая философия, а scrum это уже практически применимый набор инструментов. Важно понимать, что scrum — это не пошаговый какой-то метод, как сделать наш it-продукт, это всего лишь набор определенных правил, которые помогут нам при разработке, и это обязательно важно помнить. Scrum даёт определённый базовый комплекс мероприятий, которые позволят нам настроить работу нашей команды, определить какие люди нам нужны в этой команде. И при этом учесть уникальность абсолютно любого it-проекта. Scrum может дополнять с собой какую-то уже используемую текущую методологию разработки, либо полностью работать самостоятельно, без каких-то других взаимодействий с иными методологиями, есть определённые элементы и ценности scrum.



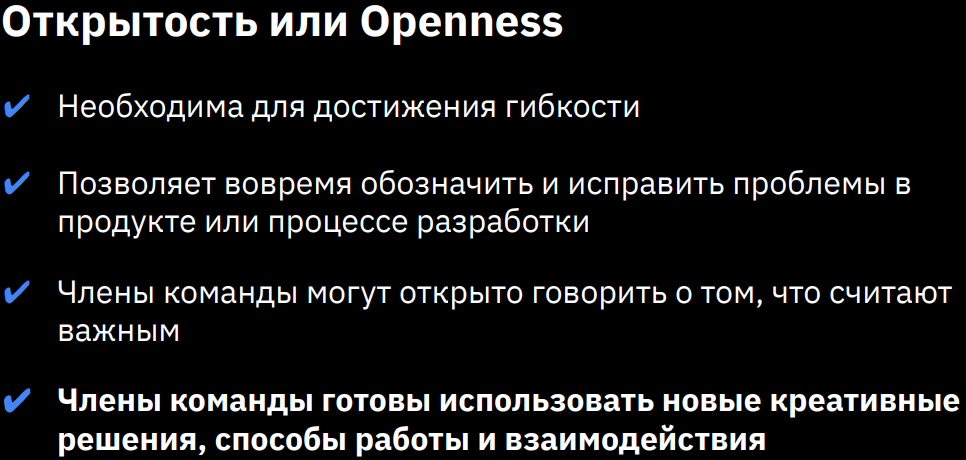
Что это такое? Сначала элементы, есть четыре основных элемента: это ценности, роли, артефакты и практики. Так называемые ритуалы с помощью, которых работает команда именно так это и называется, ритуалы scrum. Для начала поговорим про ценности. Есть пять основных ценностей scrum — это сфокусированность, открытость, смелость, приверженность и уважение.



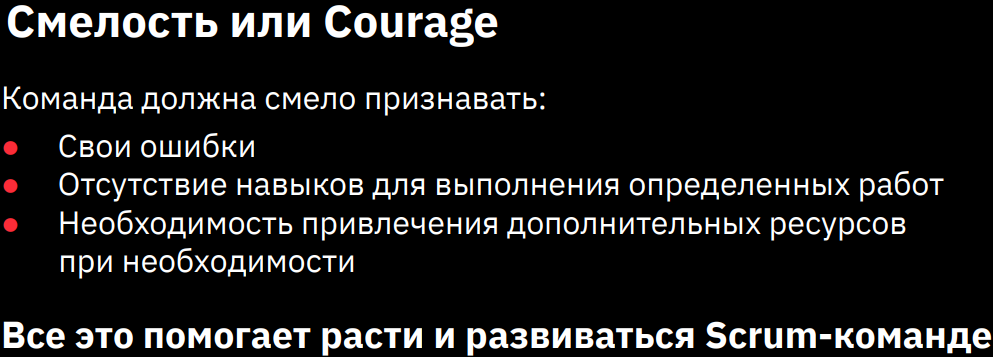
Сфокусированность, что это такое, как это практически применимо? Когда мы делим нашего большого слона на маленькие части, нам необходимо в определенную единицу времени, так называемые спринт, выполнить определённый пул задач, определённый набор работ, которые позволят достигнуть небольшой маленькой цели, в рамках нашего большого it-проекта. В таком случае наша команда концентрируется только на выполнение этих задач, внимание не распыляется, на какие-то иные цели и работы и, мы потихоньку шаг за шагом достигаем своей цели. Почему это важно? Потому что распыление на большое количество задач, оно чревато тем, что снижается продуктивность нашей команды.



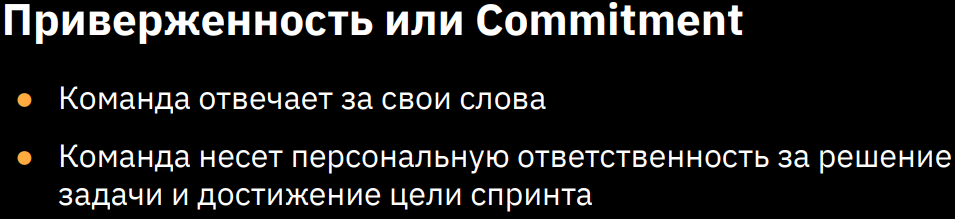
Scrum команда, должна фокусироваться на решении именно максимально актуальных и ценных для нашего заказчика задачах, это важный принцип любой гибкой методологии.



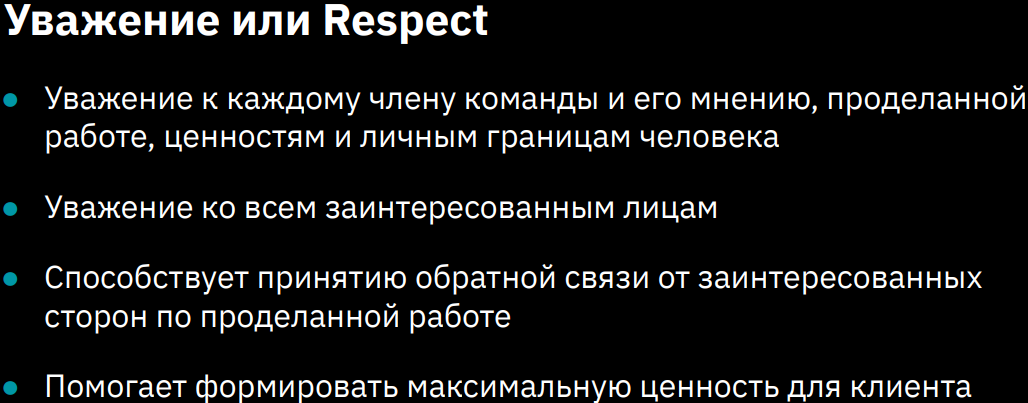
Открытость, что это такое? Как можно это применить? Открытость предполагает то, что наша команда, всегда открыта к чему-то новому. Она готова обсуждать любые изменения с нашим заказчиком, она готова слушать мнение членов команды, прислушиваться к ним и брать самые лучшие практики, самые лучшие предложения при разработке. Важно отметить, что при открытости, члены команды готовы делиться, готовы получать новые знания, готовы обсуждать это всё со своими коллегами, для того чтобы улучшить конечный продукт.



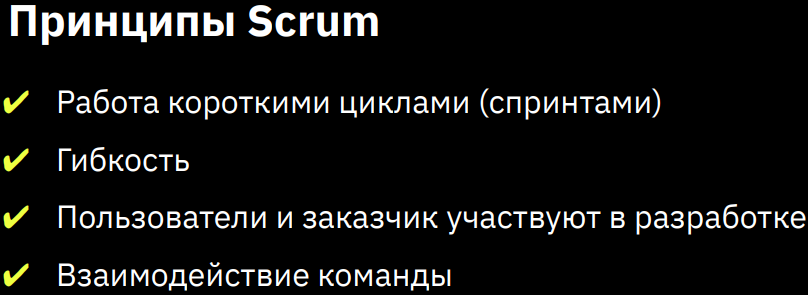
Смелость или Кураж. Что это? Это означает то, что наша команда готова признавать свои ошибки там, где ей необходима помощь она готова это признать и получить эту помощь, либо со стороны заказчика, либо со стороны каких-то привлеченных подрядчиков или членов внутри команды. Всё это необходимо для того, чтобы расти и развиваться scrum команде.



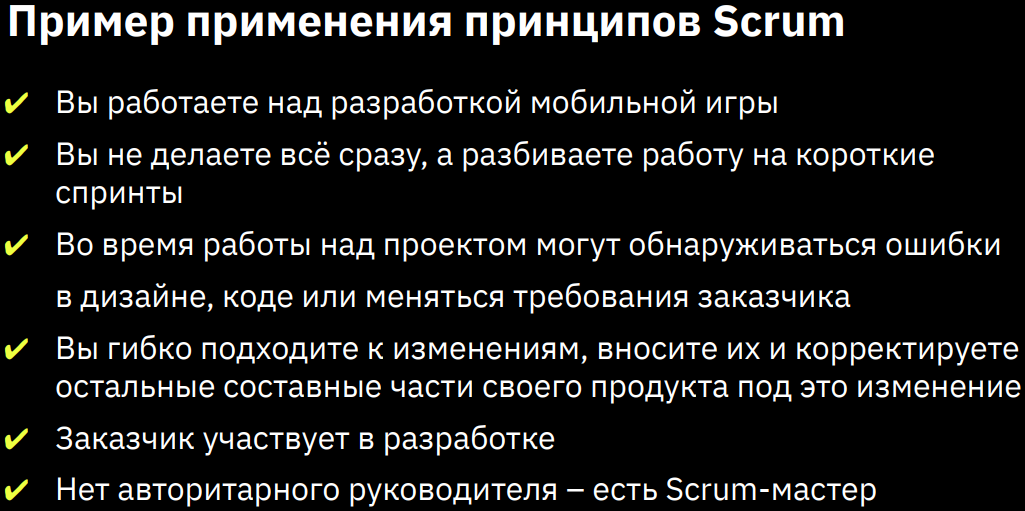
Следующая ценность — это приверженность или commitment, команда должна отвечать за свои слова, должна придерживаться тем, изначально установленным требованиям и соблюдать их. При этом при необходимости, эти требования можно менять, обсуждая что-то с заказчиком, при разработке нашего it-продукта.



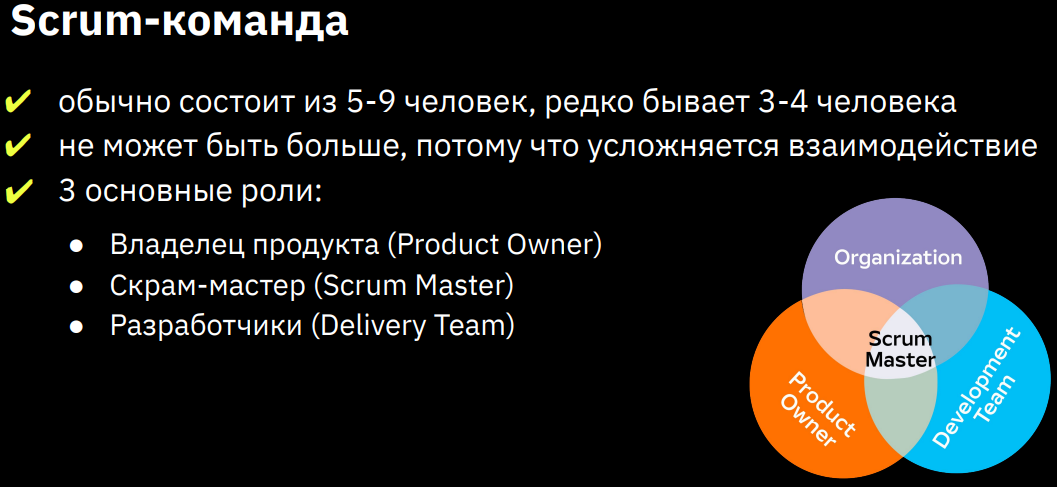
Следующий важный момент — это уважение, которое применяется абсолютно во всех гибких методологиях, в том числе scrum. Это уважение в целом к людям внутри команды, уважения к требованиям нашего заказчика, уважение к мнению внутри нашей команды. Это очень важный момент, именно это, тоже позволяет совершенствоваться и развиваться нашей scrum команде, также уважение способствует тому, что мы можем принимать какую-то обратную связь от заинтересованных сторон, от стейкхолдеров нашего проекта. Помогает сформировать максимальную ценность нашего продукта, для нашего заказчика и конечных пользователей.



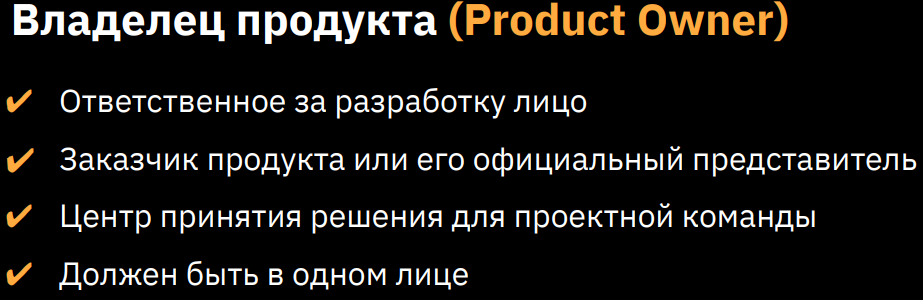
Какие включает в себя принципы scrum? Есть 4 основных принципа — это работа над нашим проектом короткими циклами, спринтами так называемыми, гибкость, пользователь и заказчик участвует в разработке помимо команды разработчиков и конечно же взаимодействие внутри команды.



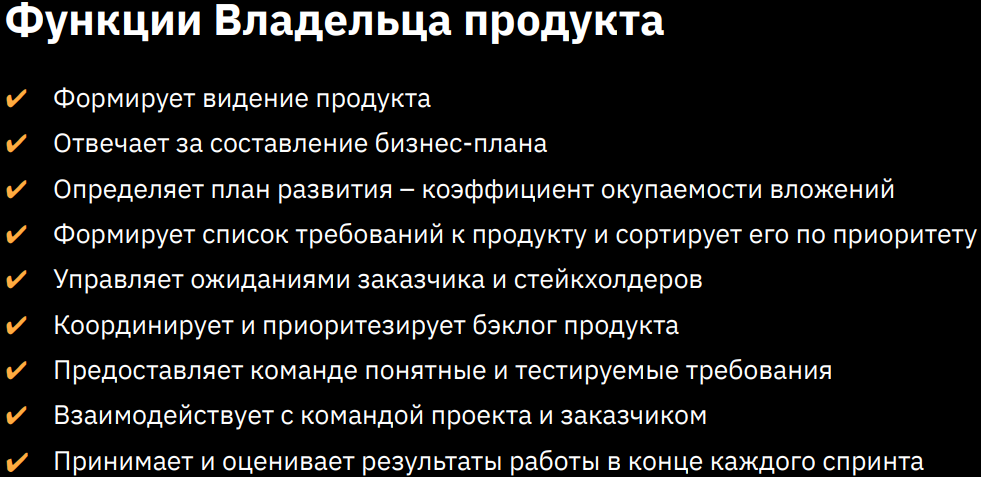
Давайте посмотрим на примере, как эти принципы применяются что это такое? Допустим вы работаете на той же самой мобильной игрой разрабатывается её вы не делаете всё сразу, разбиваете свой большой проект на небольшие спринты, в рамках которых достигаете определённых целей, шаг за шагом идёте к своей конечной цели. Во время работы над проектом могут конечно же обнаруживаться ошибки в коде, в дизайне, в подходе и так далее, вы гибко подходите к этому моменту, устраняете эти ошибки, разговариваете об этом с вашим заказчиком, вносите какие-то корректировки и улучшения в целом в разработку, в сам процесс. Заказчик также должен участвовать в разработке, потому что он контролирует данный процесс и для него даже выделена определённая роль, внутри scrum команды. У вас нет авторитарного руководителя, у вас нет определённого менеджера, есть скрам-мастер. Что он делает? За что он отвечает? И как в целом подходит к работе? Мы поговорим чуть позже.



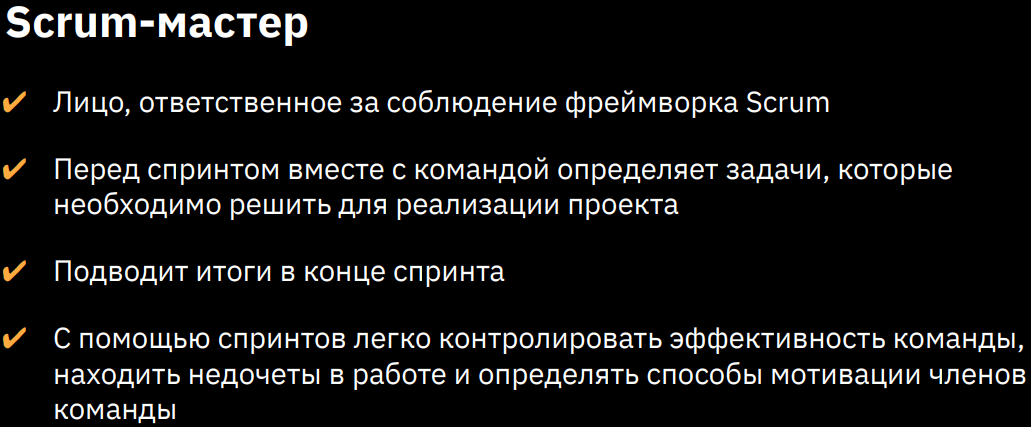
Теперь давайте поговорим про scrum команду потому, что для выполнения любого проекта нам конечно же нужны люди. Из кого состоит наша scrum команда? Обычно это бывает 5-9 человек, редко 3-4, больше, чем 5-9 человек привлекать в команду не имеет смысла. Иначе не получится у нас гибкий подход и тот самый фреймворк scrum, какие есть основные три роли внутри scrum команды? Это владелец продукта (product owner), скрам-мастер (Scrum Master) и команда разработчиков (Delivery Team), которые уже трудятся непосредственно над самой разработкой. Владелец продукта, что он делает и за что у нас отвечает в рамках работы?



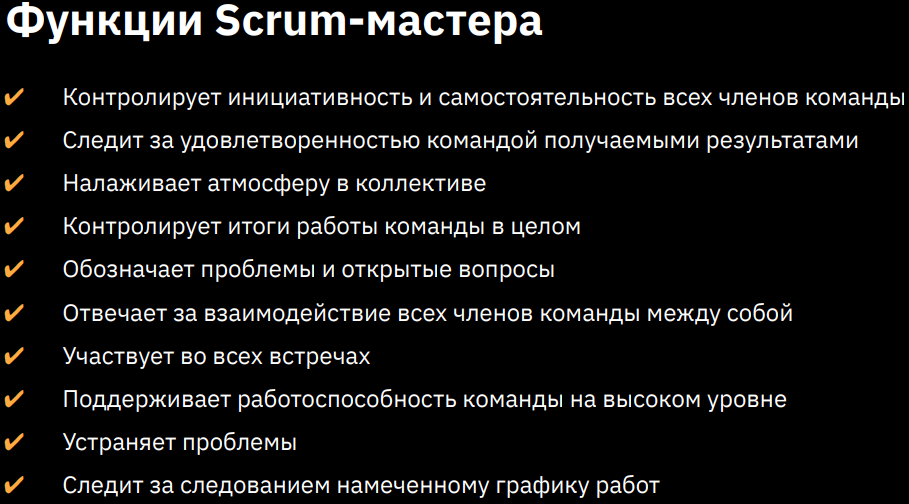
Во-первых, это лицо, которое ответственно в целом за всю разработку, это человек, со стороны заказчика, это может быть его доверенный представитель, либо сам заказчик, может также выступать владелец нашего продукта. Это тот самый центр принятия решений абсолютно всех на нашем проекте. Очень важно отметить, что любой владелец продукта, должен быть в одном лице, два человека недопустимо, произойдёт разногласие в каких-то мнениях и принятием решений, функции которыми он обладает, которыми он наделён и соответственно, что он делает в рамках работы всей команды?



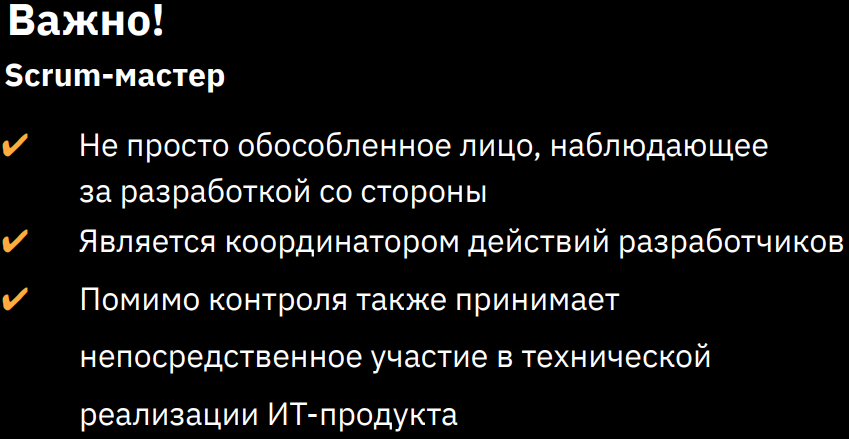
Во-первых, формирует видение всего проекта, рассчитывает бизнес-план, выставляет определенные метрики. Каким образом можно будет протестировать, оценить, наш конечный, готовый продукт? Также он формирует список всех необходимых требований, к нашему it-продукту, управляет ожиданиями заказчика стейкхолдера, предоставляет команде понятные, чёткие требования, в соответствии с которыми они уже работают. Важно понимать, что владелец продукта в конце принимает и оценивает результаты не только всего проекта, но и каждого спринта в рамках, которых выполняются определённые работы.



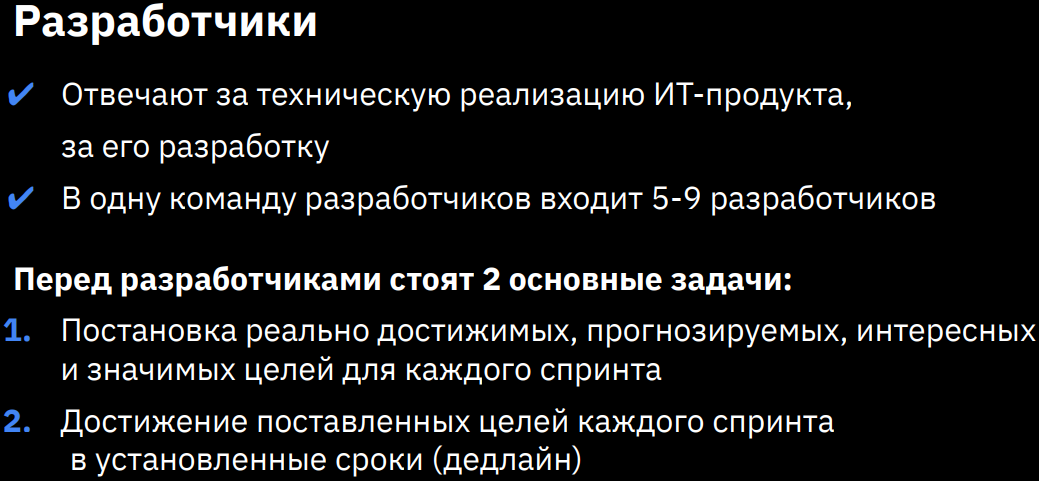
Следующее важное лицо внутри с командой это скрам-мастер важно помнить, что это не руководитель проекта — это именно скрам-мастер, который отвечает за соблюдение самого фреймворка, за соблюдение его ценности, подходов, ритуалов и практик. Что он делает?



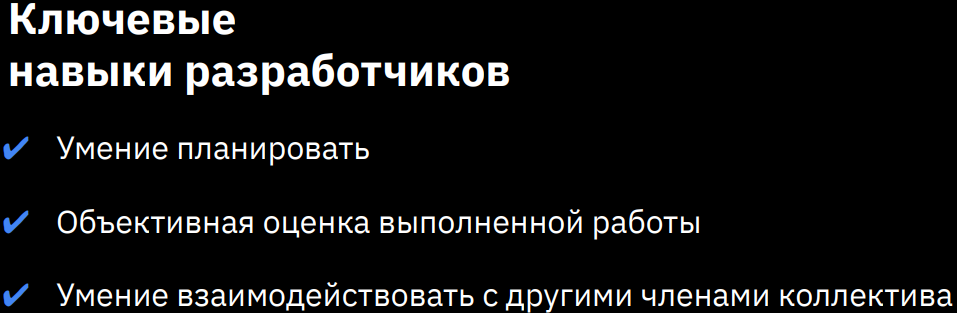
Перед каждым спринтом, он с командой обговаривает, определённые задачи обсуждает это всё, и в целом смотрит за соблюдением фреймворка — это важный момент. В конце любого спринта он подводит итоги, взаимодействует со всеми членами команды, в том числе и с владельцем продукта, и с помощью спринтов легко, контролирует весь цикл разработки, какими функциями у нас наделён скрам-мастер? Важно помнить, что это не руководитель проекта, а именно скрам-мастер. Соответственно, что он делает? Он контролирует инициативность и самостоятельность всех членов, он следит за удовлетворенностью команды, смотрит насколько хорошо происходит коммуникация, контролирует итоги работы команды, налаживает атмосферу в коллективе, естественно поддерживает работоспособность команды, участвует во всех встречах, следит за следованием нашему намеченному графику производства всех работ. То есть такой человек, который организует работу всей команды и соответственно следит за тем, чтобы в конце получался результат, в рамках каждого спринта.



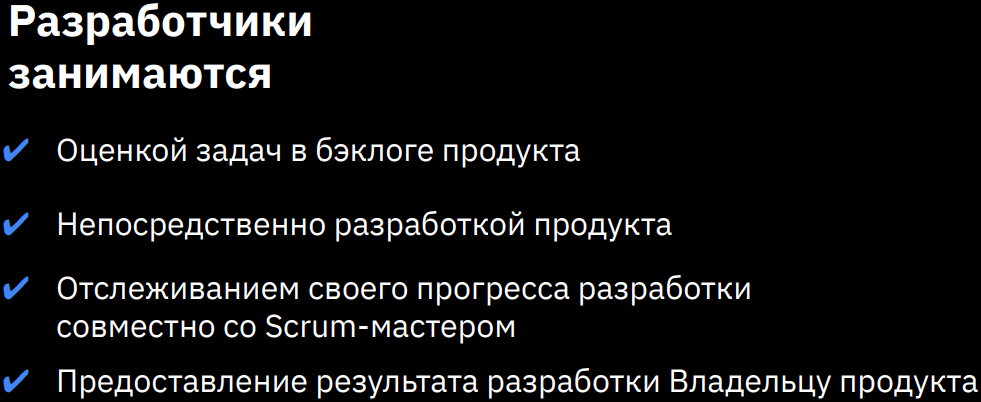
Важно помнить, что скрам-мастер — это не просто обособленное лицо, которое стоит в сторонке, и следит, налаживает процессы. Нет, это такой же участник нашей команды, он также принимает разработки, можно даже какие-то работы выполняет самостоятельно руками, пишет код, рисует дизайн и так далее. Но, помимо этого, является также и скрам-мастером. Важно отметить, что именно хорошая разработка строиться в таком порядке, что скрам-мастер, это действительно хороший технический специалист, но плюс ещё обладатель определённых софт-скилов, которые помогают ему организовать всю работу, и в целом контролировать команду.



Разработчики — это те люди, которые отвечают за техническую реализацию нашего it-продукта, они обычно составляет команду из 5-9 человек, это обычно бывает максимум. Конечно же у разработчиков есть тимлид разработки, то есть их определённый руководитель, который распределяет тоже задачи и следит в целом, за производством работ. Перед разработчиками стоят две основные задачи: во-первых, установка реально достижимых целей в рамках каждого спринта и достижения определённых целей, заданной ded-лайн, в рамках того же самого спринта.



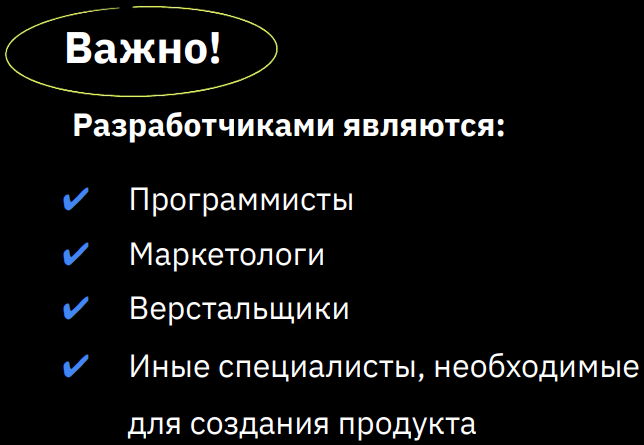
Ключевые навыки, которыми должны обладать абсолютно все разработчики, мало того, что они умеют сделать что-то руками, написать код, отрисовать дизайн и так далее. Важно ещё уметь планировать, объективно оценивать, сроки выполнения работ и технические решения выполнения таких работ. И уметь взаимодействовать с другими членами команды, потому что как вы помните, я говорил, что скрам-мастер — это не просто руководитель — это именно человек, который в целом объединяет команду и следит за выполнением фреймворка. Именно поэтому разработчики также должны самостоятельно взаимодействовать и с владельцем продукта, и с людьми из других команд, и в целом организовывать самостоятельно свою работу.



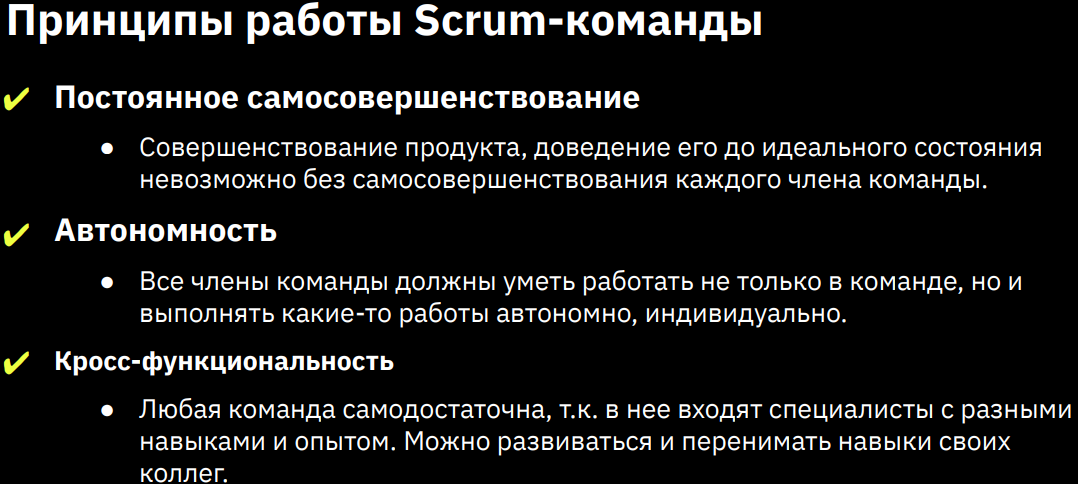
Чем занимаются разработчики? Во-первых, оценивают задачи в рамках бэклога всего it-продукта, непосредственно занимаются разработкой самого продукта, отслеживают сам процесс, вместе со скрам-мастером выявляют, какие-то моменты, тонкости, передают это скрам-мастеру, а тот уже ставит это в край угла, в тот самый вопрос каких-то проблем, которые могут возникать при работе перед владельцем продукта. При этом важно понимать, что её разработчики сами напрямую тоже могут обратиться к владельцу продукта и что-то соответственно рассказать ему. Также разработчики предоставляют в конце, результат своих работ, тому самому владельцу продуктов.



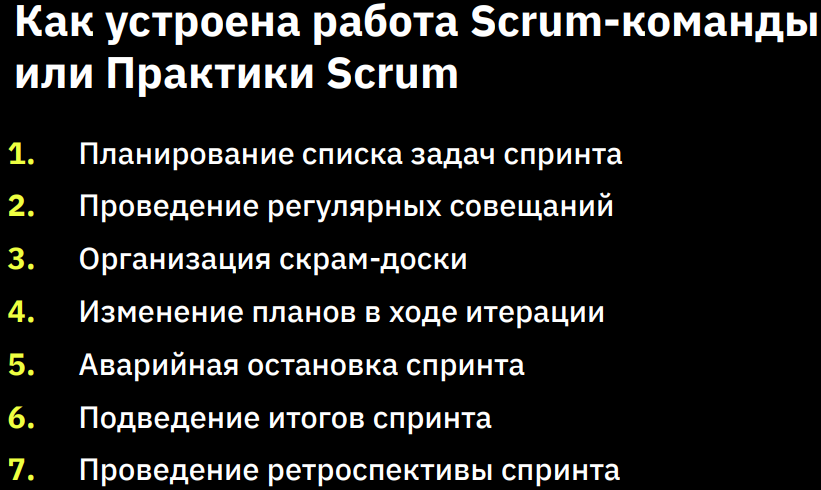
Важно помнить, что достижение цели в рамках каждого спринта — это достаточно растяжимое понятие. Именно поэтому нужно обязательно поставить какие-то метрики, которые будут свидетельствовать о том, что работа закончена, мы получили какую-то определённую часть нашего продукта и её можно уже тестировать. Соответственно, например, где-то целью, считается, что вы достигли эту самую цель, это написание только кодов, а где ты необходимо ещё и протестировать, посмотреть, насколько он работает, способен и в целом выполняет свои функции.



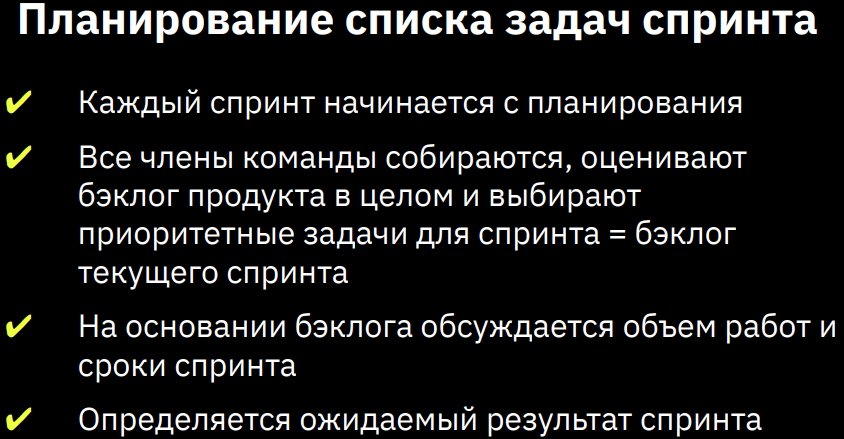
Кто такие разработчики? Здесь очень часто встречается ошибка, многие думают, что разработчики — это именно программисты, нет на самом деле это все те люди, которые задействованы в непосредственном техническом исполнении нашей части продукта, и в целом всего it-продукта, это программисты, маркетологи, верстальщики, дизайнеры и все другие специалисты, которые нам необходимы для того, чтобы что-то создать.



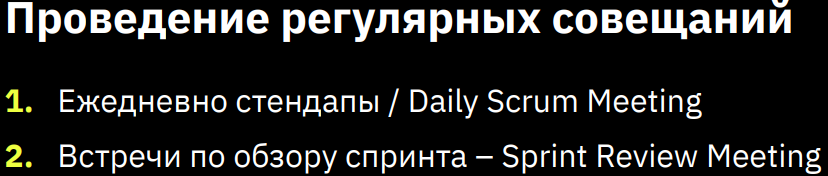
Какие основные принципы работы скрам команды? Есть три основных принципа: это постоянное самосовершенствование, потому что создать что-то уникальное и интересное, при этом не развивая себя невозможно, можно обязательно совершенствоваться самому человеку, внутри команды и совершенствовать свой продукт. Также важна автономность, конечно, хорошо работать всем в коллективе, но и важно некоторые задачи настоятельно автономно, от других людей, и кросс функциональность так как в нашей команде задействованы различные специалисты, то происходит взаимообмен опытом, который позволяет также совершенствоваться и работать автономно. То есть, эти принципы работы, они взаимосвязаны и не могут существовать друг без друга.



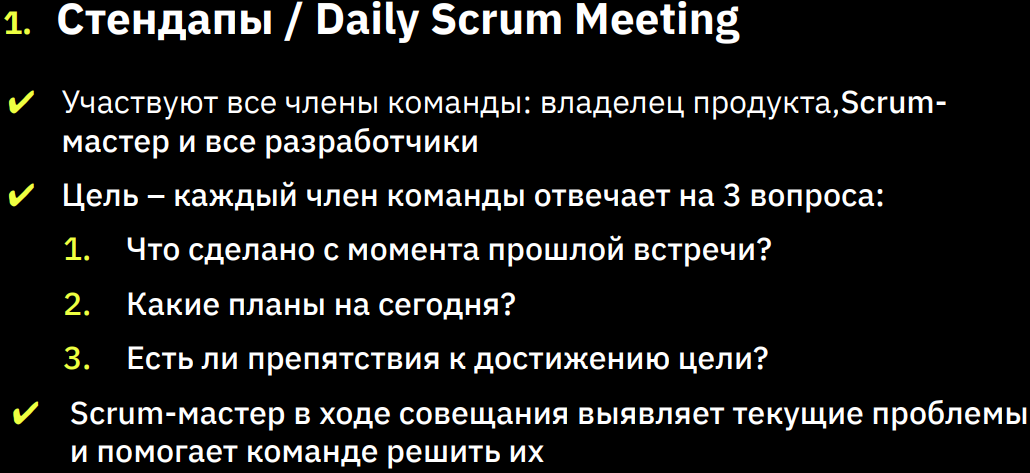
Как устроена работа нашей скрам команды или так называемые практики, как это называется, скрам гайдинг? Что это такое? Во-первых, происходит планирование списка задач всего проекта, проведение регулярных совещаний, ежедневных встреч и конечно с определенной периодичностью, чуть позже расскажу какие-то совещания. Организация скрам доски, где расписаны все необходимые задачи в рамках всего проекта, и соответственно определённого спринта. Изменение планов, то есть та самая гибкость, которая нам необходима при разработке, аварийная остановка спринта, очень интересный момент, который конечно редко встречается в проектах, но тем не менее имеет место быть, подведение итогов спринта и проведения ретроспективы спринта.



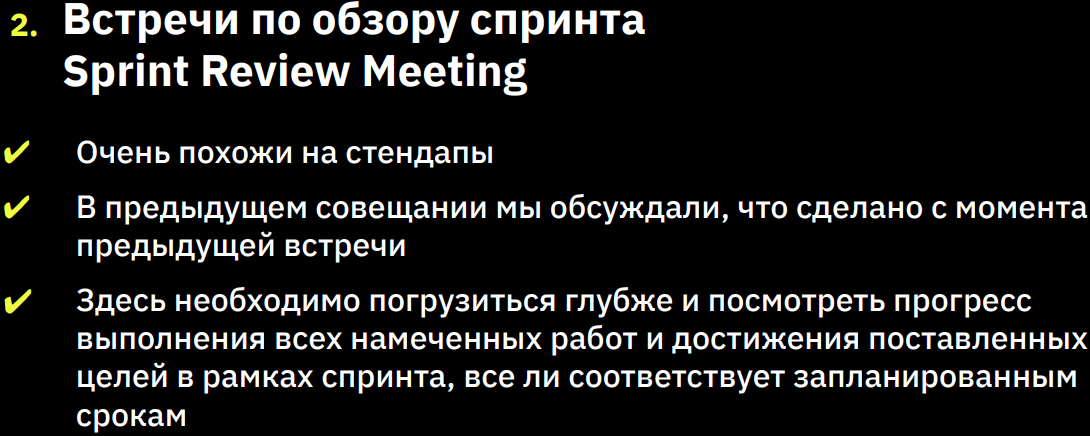
Что такое планирование списка задач спринта? Во-первых, каждый спринт должен начинаться именно с планирования, когда мы берём задачи из бэклога нашего продукта, и перемещаем их уже в блок нашего спринта, над ними мы начинаем свою работу. Как это происходит? Все члены команды собираются, оценивают бэклог продукта, выбирают оттуда те самые задачи, которые будут выполнять в рамках спринта и только после этого, приступают к разработке. Как видите, здесь важный момент, что в распределении задач участвуют не только скрам-мастер, либо владелец продукта, здесь участвует вся команда, то есть распределение задач происходит на общем, небольшом собрании, где каждый может высказать своё мнение, своё видение реализации продукта, и принести что-то новое, более совершенное, для разработки. На основании бэклога, обсуждается уже объём работ и конечно же реальные сроки, эти реальные сроки, обычно выдаёт команда разработчиков, которые оценивают действительно свои способности, насколько они уложатся в определенный промежуток времени и смогут выдать результат. Также в рамках данного совещания определяется ожидаемый результат нашего спринта.



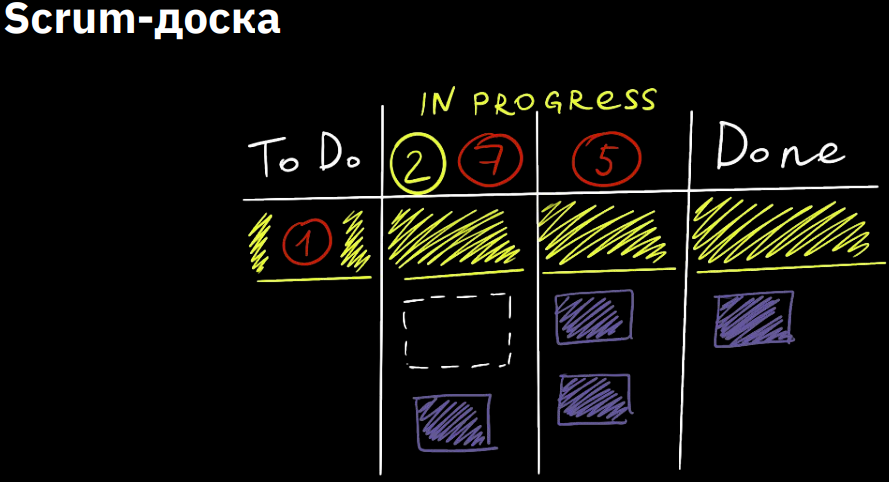
Что такое проведение регулярных совещаний? В скраме, это называется stand-up или ещё бывает, такое понятие как делискрам-митинг, есть ещё второй вид совещания спринт-ревью-митинг, то есть определённые встречи по обзору всего спринта.



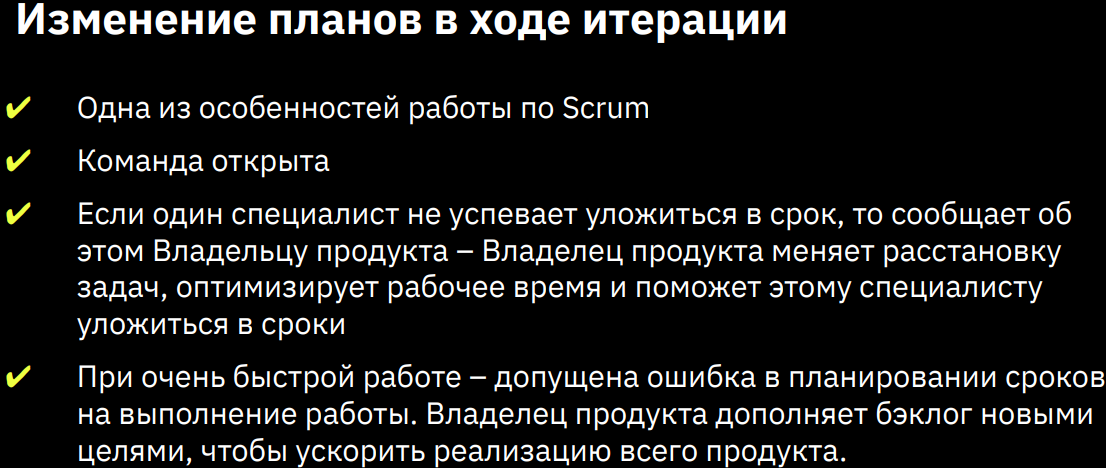
Чем они у нас отличаются? stand-up, те самые ежедневные собрания всех участников рабочий нашей группы, нашей команды. Как они проходят? Здесь участвуют, абсолютно все члены команды, скрам-мастер, и владелец продукта, и все разработчики. Какая основная цель такого совещания? Оно обычно проходит около скрам-доски, либо в небольшой переговорной комнате, где отвечают на три важных вопросов: во-первых, что было сделано с момента прошлой встречи? Что вы планируете сделать сегодня и, если какие-то препятствия для достижения поставленной цели? Такое собрание обычно 15 минут не нужно его растягивать, на несколько часов. Вам нужно чётко сформулировать определённые мысли, что вы сделали? Что вы хотите сделать и какие есть проблемы? Чтобы достичь этой цели? Скрам-мастер как раз-таки выявляет такие текущие проблемы и помогает команде решить их, всё это происходит на вот таком вот самом ежедневном собрание стендапе. Почему это стендап? Потому что, каждый, может высказывать своё мнение, нет какого-то определённого организатора, происходит полное обсуждение, внутри всей команды.



Следующее собрание — это встречи по обзору спринта, Sprint review meeting, что здесь у вас происходит в целом, такие собрания, очень похоже на стендапы, но важно отметить, что в предыдущих стендапах мы обсуждали. Что сделано с момента нашей предыдущей встречи? То есть за один день грубо говоря, мы смотрим задачи, которые мы сделали вчера, будем делать сегодня, и какие есть проблемы. Если мы говорим про встречу, по обзору спринта, здесь нужно посмотреть гораздо глубже, посмотреть в целом, прогресс выполнения всех задач, в рамках всего спринта. Насколько мы двигаемся правильно и плавно, по наши нашему графику и всё ли идёт хорошо? Если есть какие-то проблемы, команда также сообща их решает, находит пути и мероприятия, соответственно, которые позволят решить возникшие проблемы.



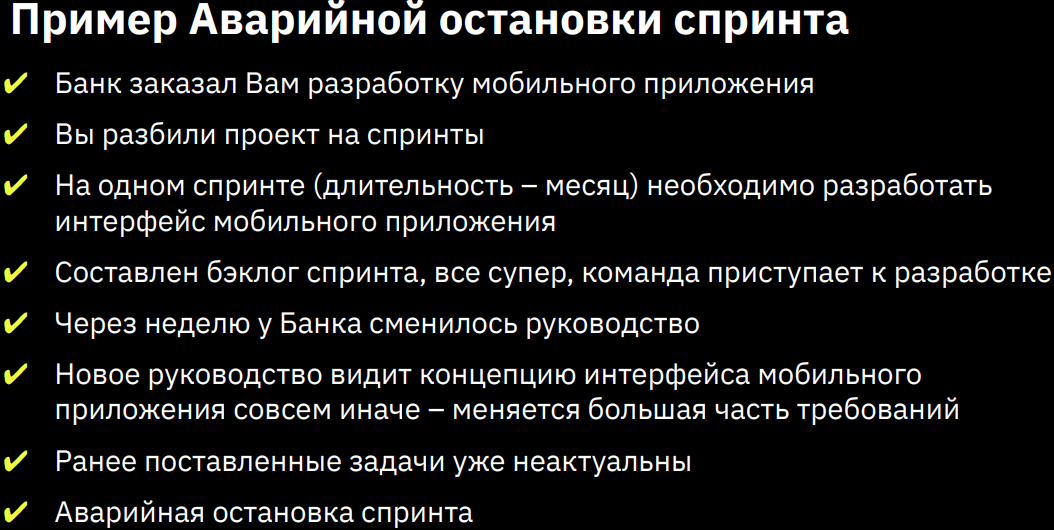
Для того, чтобы наглядно представить все задачи, которые есть в рамках спринта используется скрам-доска, обычно такая доска делится на три колонки. Первая колонка — это to do, то есть то, что нам нужно сделать, вторая колонка — это in progress, то есть то, что у нас сейчас в работе, и третья колонка - то, что у нас уже сделано. Важно отметить, что в рамках работы нашей команды итерациями, происходят какие-то изменения, это тот самый один из основных принципов работы с помощью гибкого подхода — это изменения.



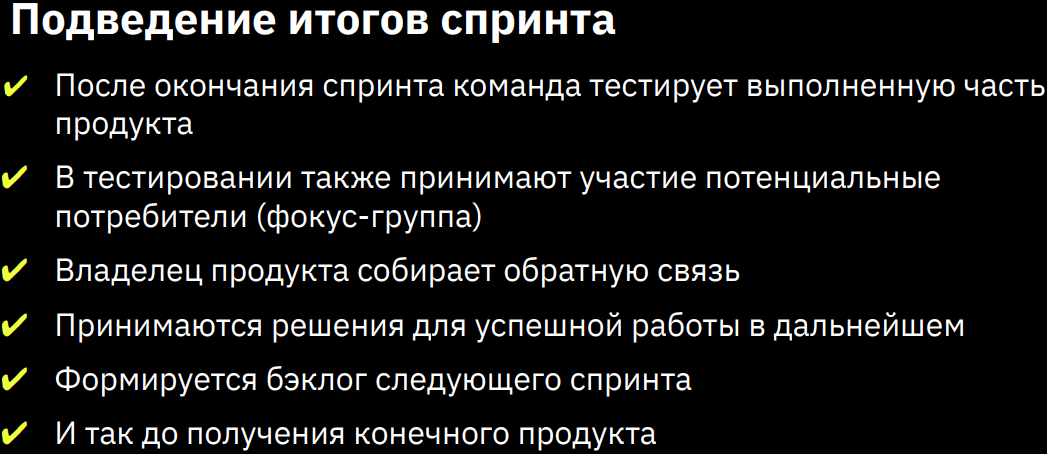
Как это происходит? Команда наша открыта, допустим какие-то члены команды не успевают выполнить определённую работу, потому что возможно был где-то неправильно, некорректно, спланирован срок выполнения такой работы, либо у членов команды не хватает хардскилов, чтобы это сделать. Таким образом они, на общем собрании, обсуждают это и грубо говоря просят помощи, скрам-мастер, владелец продукта, должен либо перераспределить эти задачи, сроки их выполнения, то есть понять истинную причину, найти её и решить эту самую проблему. Также бывает, что разработка двигается достаточно быстро — это тоже определённая ошибка, которая была допущена в самом начале, то есть где-то это были заложены слишком большие сроки на выполнение работы, в таком случае владелец продукта, должен скорректировать наш бэклог спринта и возможно, даже бросить какие-то дополнительные работы, чтобы сделать нашу разработку максимально, удобной, комфортной и возможно даже завершить её раньше установленного срока. Есть такой интересный момент, как аварийная остановка спринта, своеобразно сигнал стоп.



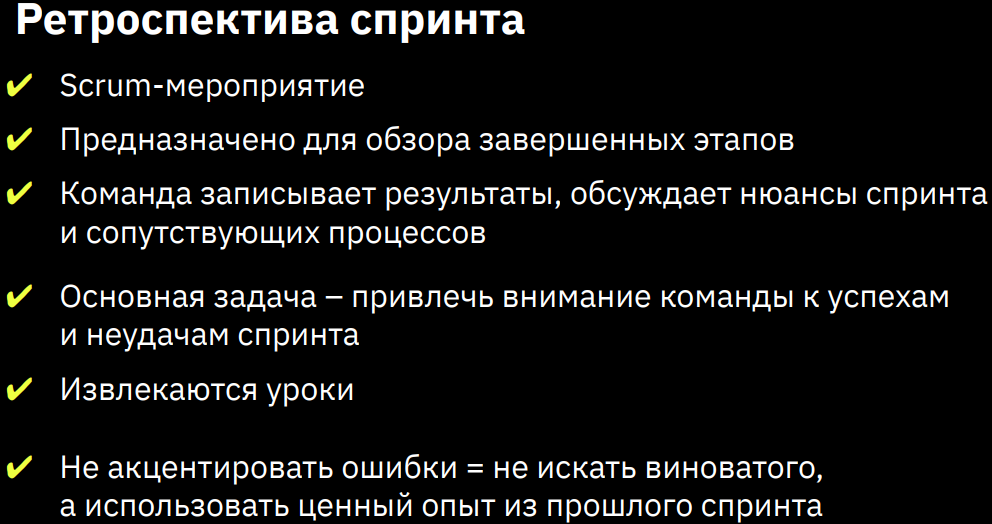
Что это такое? Когда это бывает? Такая аварийная остановка спринта, необходима только в редких случаях, команда разработчиков, скрам команда, может остановить спринт, если понимает, что намеченных целей, уже не достичь с помощью текущих поставленных задач, либо владелец продукта видит, что достигать цели данного спринта уже нет никакой необходимости. Что-то поменялось, возможно требование, возможно технические решения и тогда он тоже может остановить наш спринт. После того как спринт останавливается, происходит совещание, обсуждаются проблемы, вырабатываются какие-то мероприятия, по улучшению текущей ситуации, либо изменяются в целом технические требования, к разработке. После остановки, конечно же планируется новый спринт, с новыми задачами.



Как это происходит на практике? Давайте посмотрим, пример аварийной остановки спринта, допустим банк заказал вам разработку, какого-то мобильного приложения, для своей системы, возможно это будет мобильное приложение банка, которым будут пользоваться много клиентов, каждый день. Приложения для Android, для iOS. Вы разбили ваш большой проект на определенные спринты и начали разработку, пока вы разрабатываете на одном из спринтов, у вас допустим была поставлена цель — это отрисовать интерфейс данного мобильного приложения, пока вы разрабатываете свой проект, пока выполняются работы в рамках данного спринта у руководства банка поменялись люди. То есть, руководство полностью сменилось, и у них уже другое видение. Как должен выглядеть интерфейс вашего мобильного приложения? Точнее говоря, их мобильного приложения, ведь это всё-таки их банк. Соответственно, уже что-то делать в рамках текущего спринта, не имеет никакого смысла, поменялись изначально установленные требования, нужно вовремя остановить спринт. Происходит аварийная остановка спринта, получаются новые исходные данные, их естественно формирует заказчик, то есть тот самый владелец продукта и затем уже, планируется новый спринт, в рамках которого, выполняются другие работы, с уже новыми полученными исходными данными.



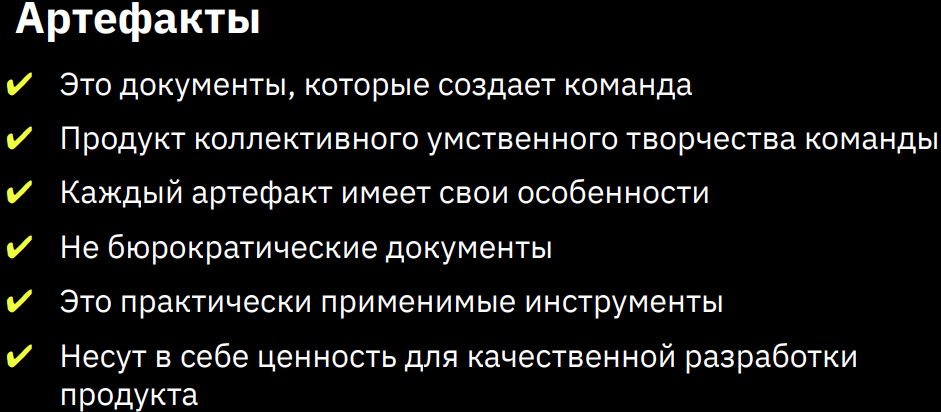
Следующий важный момент — это подведение итогов спринта, который происходит в конце каждого абсолютно спринта. Что мы здесь делаем? По окончании спринта, команда обычно тестирует свою часть разработанного it-продукта. В тестирование, принимают участие потенциальные заказчики, потенциальные пользователи, так называемая фокус-группа, которая будет использовать уже в дальнейшем, наш it-продукт, смотрят, насколько получилась работоспособная версия, небольшая версия, в рамках нашего спринта. Владелец продукта у нас, собирает обратную связь, принимается решение, для успешной работы в будущем, формируется backlog следующего спринта. Соответственно, приступаем уже потихоньку к новому спринту.



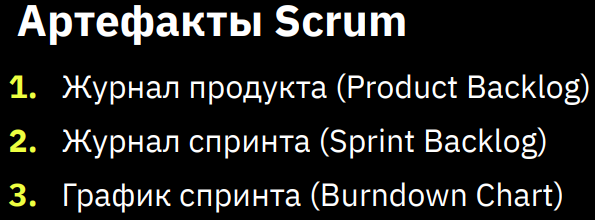
Ещё важный один момент — это ретроспектива спринта. Чем она отличается от подведения итогов спринта? Это скрам мероприятие, которое нарушать нельзя, предназначено для обзора завершённых этапов, то есть складывается большая общая картинка, после выполнения каких-то нескольких спринтов, после достижения определенных целей, создания части какого-то нашего it-продукта. Команда записывает определённые результаты, извлекает определённый урок и смотрит, как не допустить текущих ошибок в будущем. Это очень важный момент, который помогает совершенствоваться и команде, и в целом разработке того самого it-продукта. Основная задача такого мероприятия, это привлечь внимание всех членов команды разработки, в том числе скрам-мастера, владельца продукта к основным проблемам, каким-то успехам и неудачам, которые были выполнены в рамках нашего спринта. Важно отметить, что это мероприятие не носит какого-то наказательного характера. То есть, у вас не должна быть акцентирована, именно какая-то ошибка команды, здесь важно верхне-уровнево и где-то даже, детально проанализировать, что способствовало успеху, а что наоборот привело вас к неудаче какой-то. Выработать определённые решения, как вы будете делать дальше в дальнейшем, в последующих спринтах и соответственно вынести ценный опыт, из этих полученных уроков. То есть ни в коем случае нельзя акцентировать внимание на ошибках, кого-то из вашей команды, это не будет способствовать совершенствованию. Важно отметить, что можно сделать лучше, именно такой должен быть посыл, на проведение данного собрания.



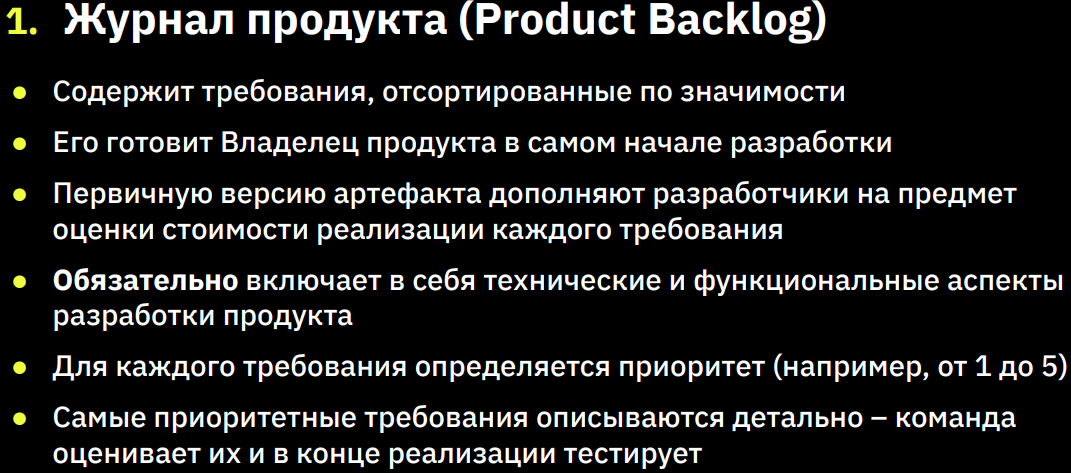
На данном слайде вы видите графическое представление: «Как происходит работа по методологии скрам». Формируется product backlog, то есть тот самый backlog нашего продукта, в котором описывается полностью, все технические требования к нашему продукту, оттуда уже выбираются задачи для бэклога спринта, стартуют спринты. Ежедневно происходит дейли скрам митинг, те самый стендапы, после завершения спринта, мы делаем его обзор, так называемое собрание по обзору спринта, затем проводим Sprint review, после нескольких спринтов, смотрим насколько успешно мы двигаемся, какие ошибки допустили, что улучшить и как действовать соответственно дальше. Идём дальше, планируем новые спринты и так до момента, пока не получим готовый it-продукт, на протяжении всей работы нашей команды. Естественно появляются, определенные документы, называются они артефакты.



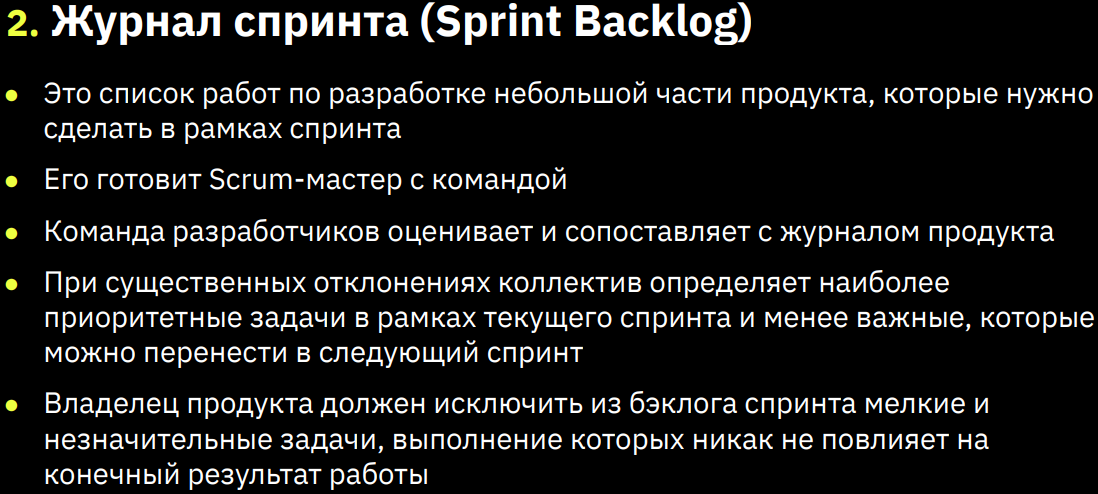
Что такое артефакты? Это продукт коллективного умственного творчества нашей скрам команды, по сути дела каждый артефакт, имеет свои особенности, это не бюрократические документы — это прямо практически применимый документ, который несёт в себе определённую ценность, совершенствовать разработку, и в целом, улучшать, весь процесс и наш конечно it-продукт.



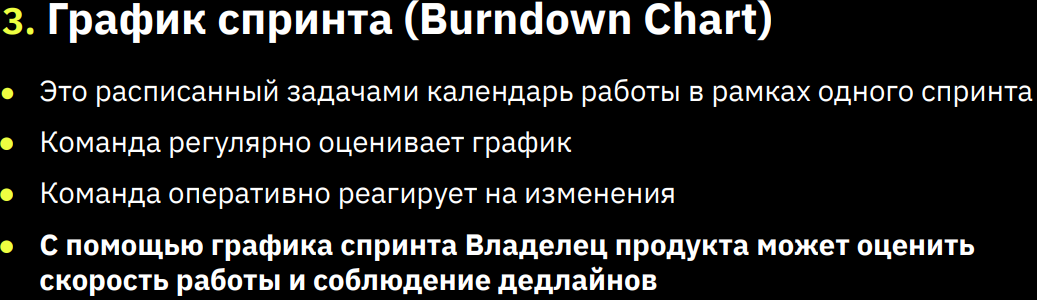
Есть три основных артефакта в скраме: это журнал нашего продукта product backlog, журнал спринта Sprint backlog и конечно график спринта Burndown Chat, по которому мы двигаемся. Давайте, посмотрим каждый по отдельности.



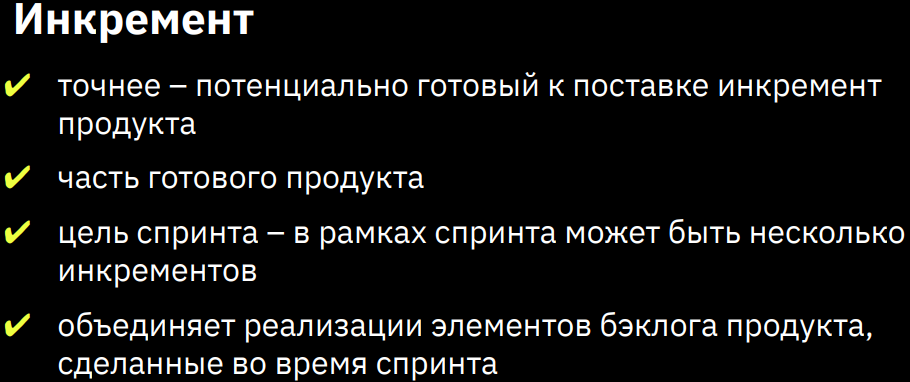
Что такое журнал продукта product backlog? Содержит в себе требования, которые отсортированы по приоритетности к нашему готовому it-продукту, его обычно готовит владелец продукта и обсуждает уже с командой, со скрам командой. Первичную версию такого продукта бэклога, обязательно смотрят разработчики и оценивают, насколько действительно реально можно это сделать, эти самые задачи, реализовать практически. Важно отметить, что product backlog обязательно включает в себя технические и функциональные особенности разработки, также внутри этого бэклога, задачи приоретезируются от одной до пяти. Соответственно, самые приоритетные задачи, максимально глубоко детализируются, прописывается по ним основные технические требования и соответственно те самые метрики, по которым будут происходить тестирования, выполнения этих задач.



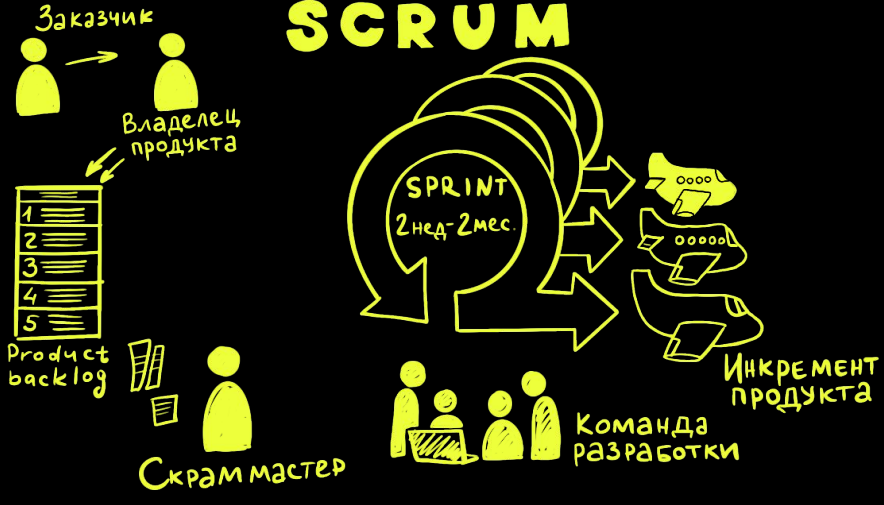
Следующий документ — это журнал спринта, из того самого бэклога всего продукта, мы выбираем приоритетные задачи, включаемых их спринты и начинаем естественно разработку. Журнал спринта готовит обычно скрам-мастер, естественно не только в одном лице, а с командой разработчиков, потому что очень важно понимать техническое видение реализации этих задач, команда разработчиков оценивает данный бэклог, сопоставляет его с журналом продукта, понимает, что где правильно, что где нет, вносит какие-то корректировки, и затем уже приступает непосредственно к выполнению этих поставленных задач, в рамках спринта, при этом над журналом спринта работает не только команда вместе со скрам-мастером, но ещё и владелец продукта. Что он делает? Обычно, если он видит, что в рамках спринта, отобраны какие-то незначительные мелкие задачи, которые в целом не способствует успеху, он их просто убирает оттуда, то есть смотрит, на предмет значимости и ценности наш журнал спринта.



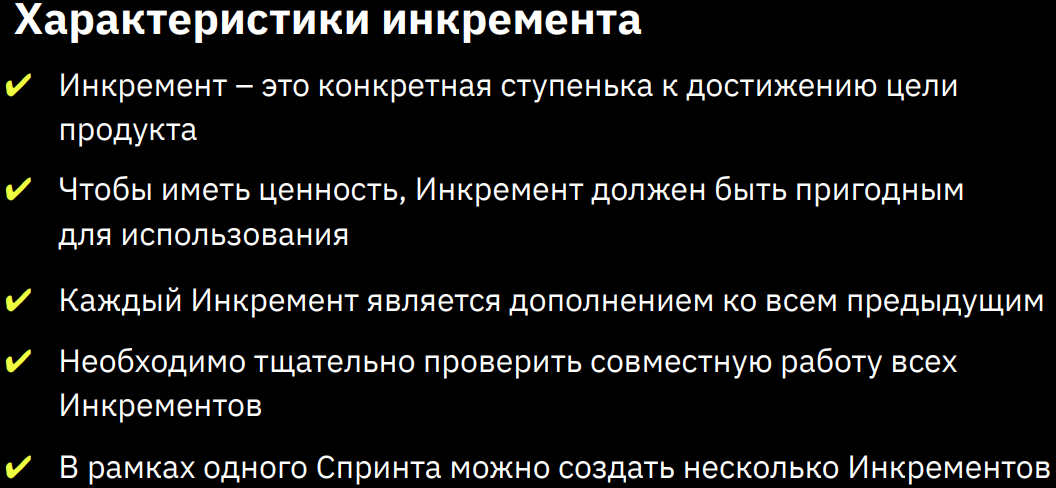
Следующий артефакт – это график, по которому мы двигаемся в рамках спринта, это расписанный задачами, календарь работы. Естественно — это практически применимый инструмент, с помощью которого, наш владелец продукта, уже контролирует весь процесс разработки, понимает, где происходит отставание, где необходимо что-то изменить, скорректировать, какие-то задачи сдвинуть, и так далее. Также с помощью данного графика спринта, наш владелец продукта, может в целом оценить скорость разработки, посмотреть на сколько она удовлетворяет тем самым требованиям заказчика, где необходимо это скорректировать и улучшить.



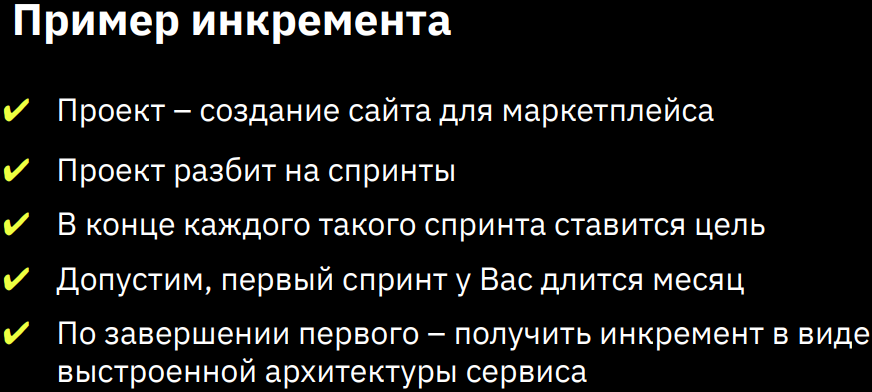
Инкремент, что это такое? В рамках каждого спринта у нас получается определённая часть нашего продукта. Это та самая цель, к которой мы стремимся в рамках спринта, называется инкремент. То есть — это потенциально, готовая, часть к поставке, то есть потенциально готовая часть, нашего общего большого продукта, для поставки нашему заказчику, её можно протестировать, оценить, и проверить насколько она работоспособна, также инкремент, в целом объединяет реализацию элементов бэклога продукта, когда мы в целом работаем в рамках одного спринта.



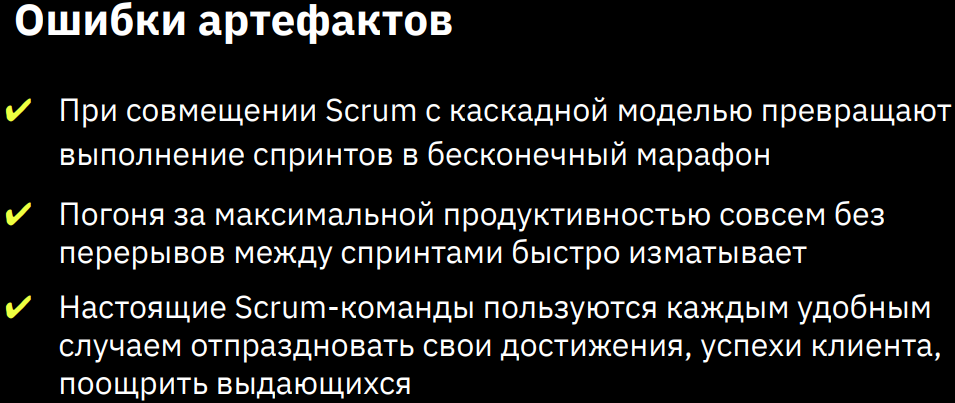
Давайте посмотрим на вот этом графическом примере, как это выглядит? Допустим у нас есть заказчик, есть владелец продукта (специалист, представитель со стороны заказчика), есть скрам-мастер, есть команда, которая уже непосредственно руками что-то делает. Есть product backlog с теми самыми задачами по всему продукту, из него мы разбили наш весь проект, на определённые спринты, взяли какие-то задачи. В рамках первого спринта допустим сделали, только лишь каркас самолёта. В рамках второго спринта мы уже пробурили определённые отверстия, где будут люки, где будут те самые окна. В рамках третьего спринта, мы уже практически сделали малую версию продукта, то есть полностью построили каркас, возможно даже уже туда внедрили двигатель самолёта, то есть всё это в рамках каждого спринта, мы получаем инкремент продукта, маленькую часть нашего одного большого продукта. Важно отметить, что в рамках спринта, может быть получен не только один инкремент, но и несколько, такое тоже имеет место быть.



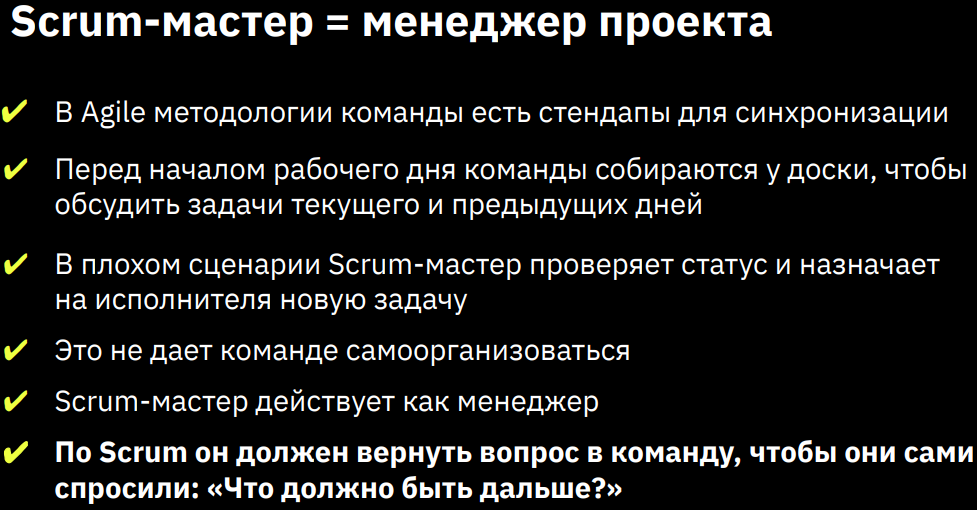
Основные характеристики инкремента, инкремент — это конкретная ступенька к достижению цели, чтобы иметь ценность, наш инкремент должен быть работоспособным, соответственно, каждый инкремент является дополнением к предыдущему. То есть это не отдельно взятая какая-то часть, это прямо наполнение нашего готового продукта, тоже самое, что мы видели сейчас на рисунке, где был сначала каркас самолёта. Затем он уже был покрашен, потом он уже стал с двигателем, и в конце мы получили готовый продукт, который может летать.



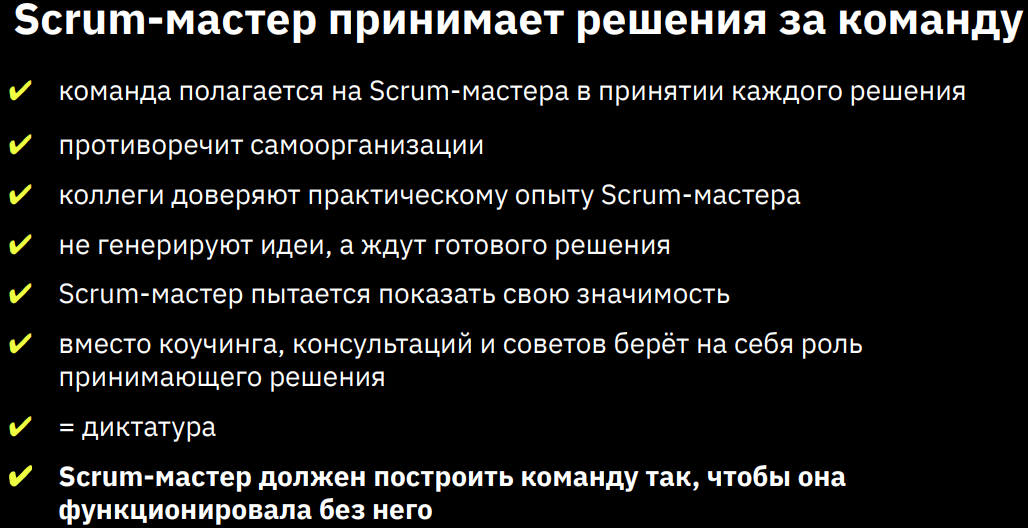
Пример инкремента, давайте посмотрим, как это выглядит, допустим вы создаёте сайт для маркетплейса, у вас проект разбит на спринты, в рамках каждого спринта, вы ставите определённую цель, то есть получаете тот самый инкремент, допустим первый спринт у вас длится всего месяц, по завершению первого спринта вы получаете инкремент в виде выстроенной архитектуры вашего сайта, для маркетплейса.



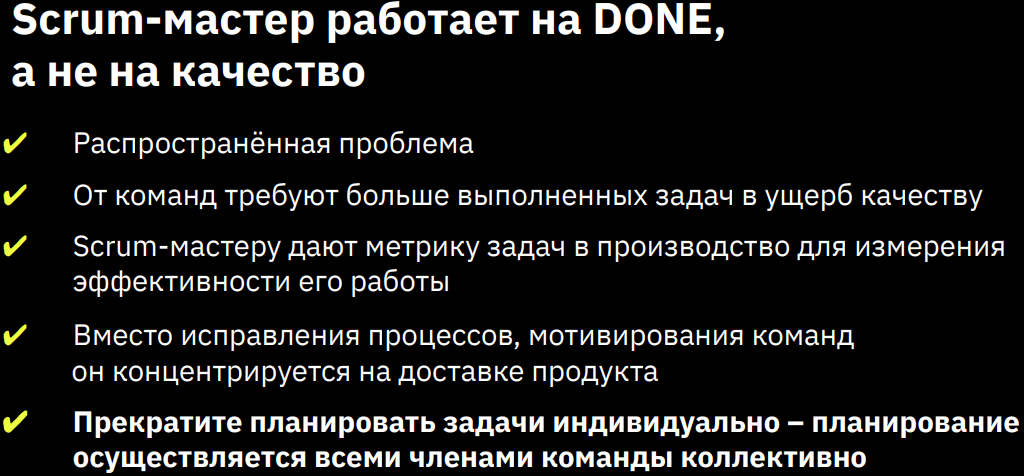
Теперь давайте поговорим, про часто встречающиеся ошибки? Что это такое? Обычно бывает ошибки артефактов, как они выглядят, когда пытаются совместить скрам, допустим с какой-то другой методологией, например, с каскадной, то начинается гонка за выполнение каких-то задач, то есть, происходит погоня за максимальной производительностью, это очень сильно изматывает, команды не успевает отдохнуть, не успевают понять, что она сделала правильно, что, где допустила ошибки, и так далее. Соответственно хорошие скрам команды, обязательно отмечают свои успехи, выделяя тех, кто отличился анализирует прошлые уроки, прошлые свои ошибки, и только после этого, уже идут дальше.



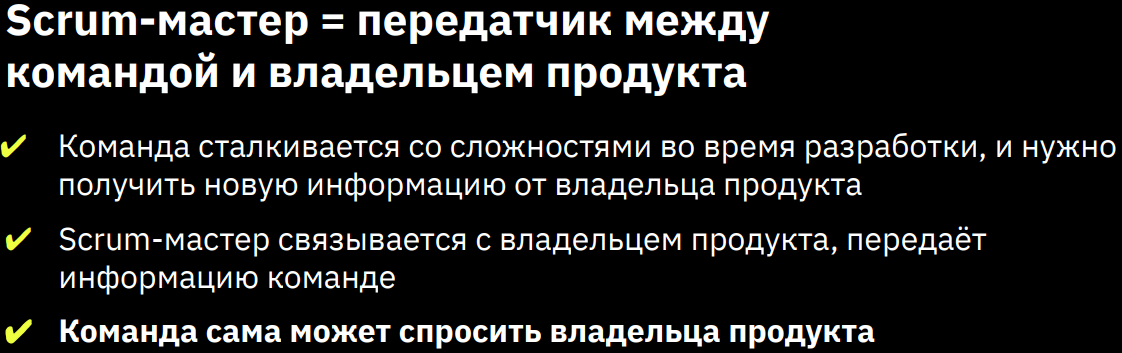
Скрам-мастер — это не менеджер проекта, это тоже очень часто встречающаяся ошибка, когда скрам-мастера, выставляют в качестве менеджера проекта. В agile методологии команда использует стендапы, для синхронизации, где они все совместно обсуждают задачи, которые перед ними ставятся, это происходит перед началом каждого рабочего дня, буквально на 10-15 минут. В плохом сценарии скрам-мастер, во время этого совещания, проверяет статус исполнения задач, на самом деле, это делать должны, сами члены команды, то есть каждый, на кого возложена определённая задача, говорит сделал он её или нет, и какие были перед ним препятствия. Скрам-мастер не должен действовать как менеджер, он не управленец, в целом фасилитирует полностью работу, над разработкой it-продукта. По скрам методологии, скрам-мастер, при возникновении какого-то вопроса, должен вернуть его обратно в команду. Чтобы произошло обсуждение и соответственно все члены команды пришли к какому-то общему мнению, a скрам-мастер, только может направлять их для того, чтобы, они соответственно пришли к этому мнению.



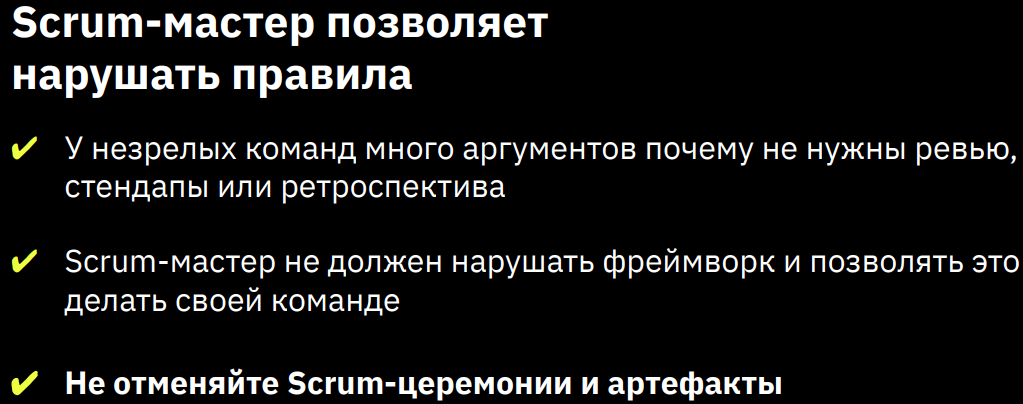
Следующая часто встречающиеся ошибка, это когда скрам-мастер принимает решение за команду, команда полагается на скрам-мастера абсолютно в принятии всех решений, то есть она видит в нём опять же менеджера проекта, это противоречит той самой методологии нашему скраму, коллеги начинают доверять практическому опыту скрам-мастера и перестают генерить какие-то идеи, перестают самостоятельно работать, над решением какой-то проблемы. Что мы видим в таком случае? Здесь мы видим, что скрам-мастер начинает принимать решения за команду, диктовать как работать, что делать, отвечает на вопросы и постепенно мы скатываемся к диктатуре. Этого ни в коем случае не должно быть. Если мы работаем по скраму.



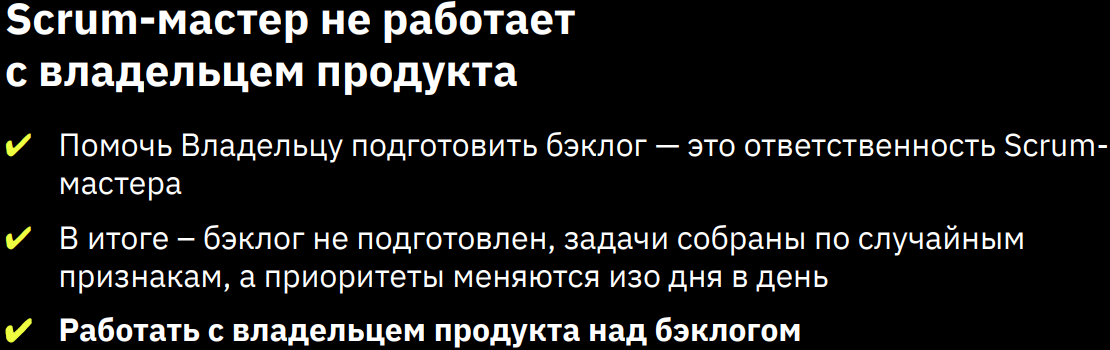
Скрам-мастер, должен построить работу команды таким образом, чтобы она справлялась даже без него, то есть самостоятельно могла решить какие-то проблемы и при необходимости, обратиться к другим участникам команды. Скрам-мастер работает на результат, а не на качество, это тоже очень часто распространённая ошибка, от команд требует большого выполнения задач, то есть как можно больше докидывают работ, задач, в рамках спринта, скрам-мастер начинает трудиться над тем, чтобы всё это успеть. В такой гонке, у нас теряется качество, вместо исправления процессов, мотивирование команды, он следит за исполнением поставленных задач, это ни в коем случае делать нельзя. Нужно прекратить планировать задачи за команду, важно вернуть вопрос обратно в команду, спросить за сколько времени они смогут это сделать, соотнести с теми требованиями заказчика, которые у нас есть и при необходимости скорректировать их.



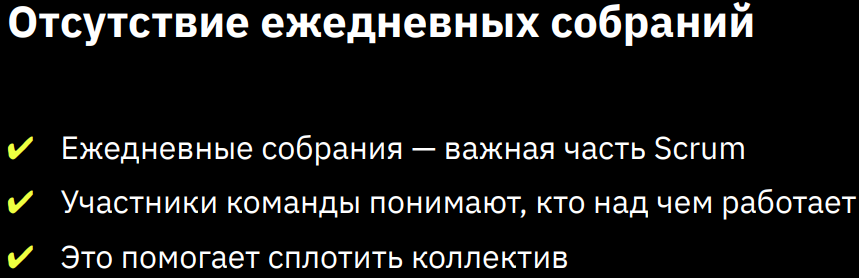
Скрам-мастер, как передатчик между командой и владельцем продукта, опять же, важно отметить, что, когда команда, сталкивается со сложностями в процессе разработки, она может напрямую спросить, что-то, что её интересует у владельца продукта, не нужно идти к скрам-мастеру, чтобы он уже спрашивал у владельца продукта, если вам что-то не понятно, идите и спрашивайте сами. Это очень важный момент, в таком случае, если вы будете обращаться сначала к скрам-мастеру, а тот уже к владельцу продукта, может произойти определённый, глухой телефон, когда кто-то кого-то не до поймёт, либо тот же скрам-мастер не сможет передать всю сущность вашего вопроса и всю сущность той самой технической проблемы, которая у вас возникла.



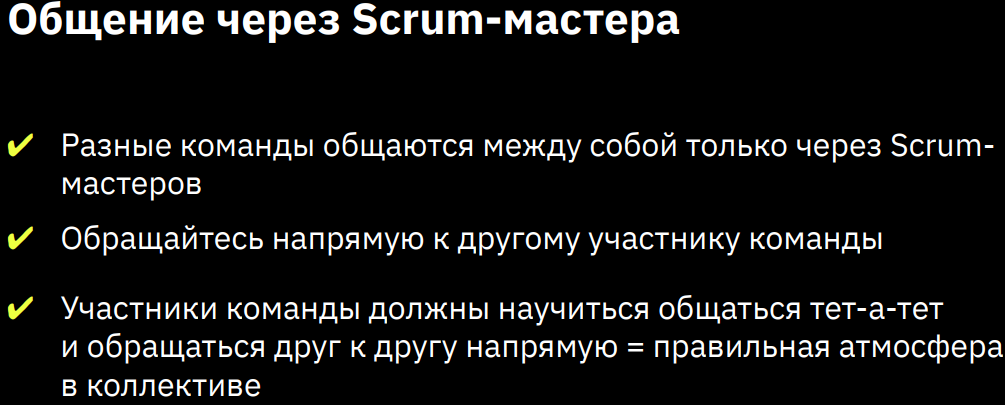
Скрам-мастер позволяет нарушать правила, когда у нас незрелая команда, и она пытается выстроить свою работу, она зачастую задаётся вопросом. Зачем нам нужны спринты? Зачем нужны какие-то review, подведение итогов, проведения стендапов, и так далее? И принимает решение, что у неё всё получится, даже без этих определенных ритуалов, которые нужны по-скраму. Скрам-мастер позволяет, это сделать своей команде, соответственно, в результате мы видим некачественный процесс разработки, потому что нарушается сам фреймворк, ни в коем случае не отменяйте правила скрама, проводите стендапы, проводите различные review, полностью, следуйте, тому самому фреймворку.



Скрам-мастер не работает с владельцем продукта. Одна из основных задач скрам-мастера — это помочь владельцу продукта, распределить задачи по спринтам, зачастую, когда ещё только-только молодая команда начинает работать по данной гибкой методологии, скрам-мастер забывает про это, и перекладывает всё это на владельца продукта. В результате, что мы видим, backlog бывает неподготовленный, либо задачи у нас собираются по каким-то случайным признакам, без достаточно глубокого и детального анализа, поэтому очень важно чтобы скрам-мастер всегда работал, вместе с владельцем продукта над backlog.

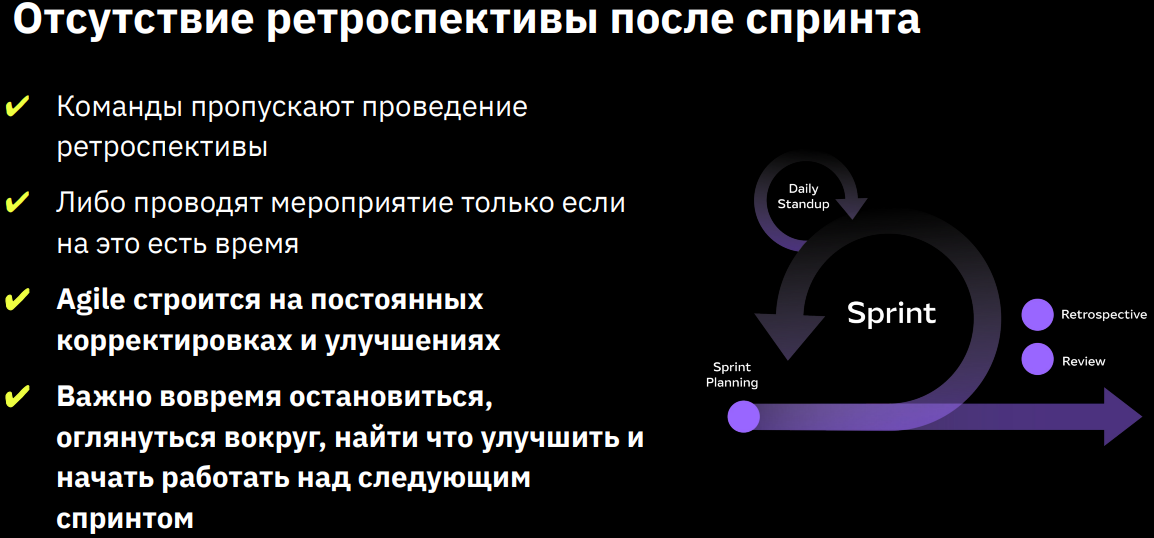


Отсутствие ежедневных собраний? Часто люди думают, что каждый день собираться на 10-15 минут, это очень сложно и не несёт в себе никакой ценности. На самом деле, это не так, ежедневные собрания — это очень важная часть нашего скрама. Почему? Потому, что на таком собрании участники, получают обратную связь, они видят, кто над чем работает? Какие задачи выполнены? Какие наоборот застряли и где нужна помощь? Это очень хорошо помогает сплотить коллектив, чтобы они качественно работали над нашим продуктом.

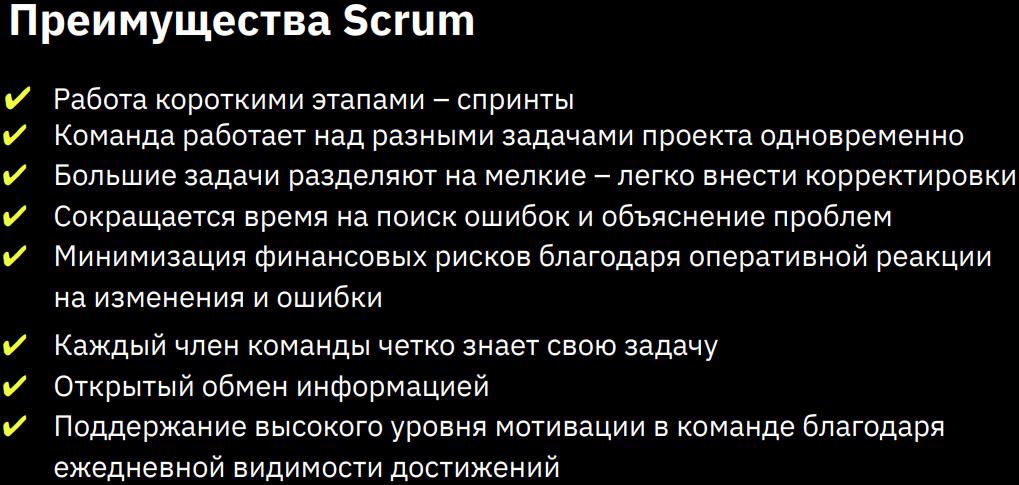


Следующий момент, это общение через скрам-мастера, допустим внутри команды разработчиков, возник какой-то вопрос или они хотят что-то спросить, у другой команды, этого ни в коем случае допускать нельзя.

Важно отметить, что любой участник команды, может обратиться к другому участнику совершенно другой команды, важно налаживать коммуникацию, важно действовать всем сообща, не нужно, превращать скрам-мастера, в определённого передатчика, и только через него связываться, с другими участниками, других команд.

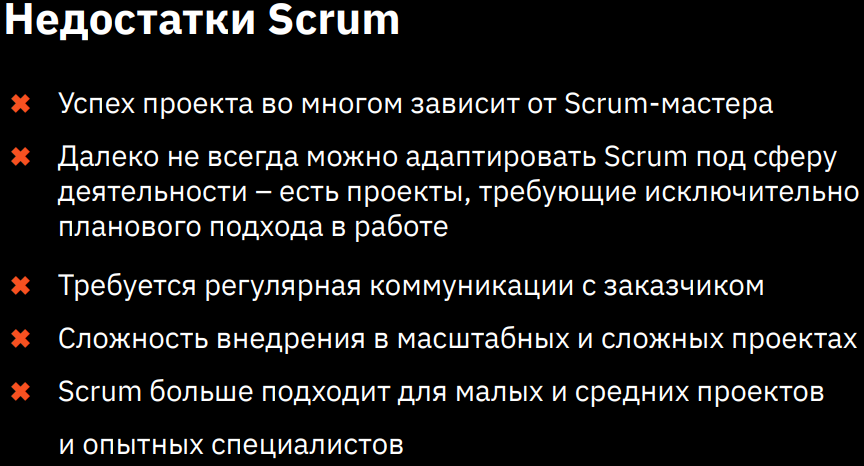


Отсутствие ретроспективы, после спринта? Зачастую люди думают, что достаточно просто, провести ревью спринта, то есть подвести определённые итоги, посмотреть, протестировать, часть продукта, которая получена в рамках одного спринта, и не нужно делать никаких ретроспектив. Почему? Команды пропускают это мероприятие, и допускают достаточно большую ошибку. Либо проводят такое мероприятие, как ретроспектива, вот только есть-ли на это у нас есть время. Этого делать нельзя потому, что наш эджайл, наш скрам, строится на принципе постоянного совершенствования, если не вы не будете анализировать, насколько успешно, конечно, вы двигаетесь в целом по бэклогу всего продукта, а не только в рамках спринта, вы не получите действительно качественный, интегрированный it-продукт. То есть, вы не сможете вынести определённые уроки, не сможете скомпоновать те самые инкременты, те самые результаты, которые получаются в рамках спринтов. Важно вовремя остановиться, взглянуть, то, что вы сделали неправильно, насколько ваша часть it-продукта интегрируется в предыдущую часть, насколько это всё взаимно, хорошо работает.



Теперь давайте поговорим, про преимущества и недостатки скрама, во-первых, преимущества, конечно, команда работает спринтами, то есть выполняет все свои работы определёнными итерациями, может работать над разными задачами одновременно, когда большой слон разделён на маленькие части, очень легко контролировать в целом работу, и в таком случае, серьезно снижается стоимость нашей ошибки, то есть, если что мы сделаем в рамках спринта неправильно, это будет достаточно легко исправить, и скорректировать свои дальнейшие действия, как с технической точки зрения, так и с функциональной. Также каждый член команды, четко знает поставленную перед ним задачу, понимает чётко, что он должен сделать в рамках следующих трех недель, к примеру.

Открытый обмен информацией, здесь соблюдаются правила коммуникации и уважения внутри команды, ещё важный момент, это те самые стендапы, они дают нам преимущества того, что каждый член команды ежедневно, на небольшом собрании, на небольшом митинге, видит свой результат, видит результат своего коллеги, понимает куда в целом двигается вся команда. Успех всего проекта, при применении скрама, во многом зависит от скрам-мастера. Насколько качественно, он организует работу команды, насколько он их сплачивает, насколько поддерживает хороший эмоциональный фон, внутри команды.



Далеко не всегда скрам, можно адаптировать под сферу абсолютно любого проекта. Есть проекты, где необходимо, всё-таки придерживаться, какой-то плановости и по-другому там, к сожалению, никак. При работе по скраму, нам требуется постоянное взаимодействие с нашим заказчиком, чтобы получать обратную связь, формировать список задач, и вместе двигаться к нашей цели. Также скрам, достаточно сложно внедрить в больших проектах, где задействовано очень много людей, ещё в самом начале, мы говорили, что скрам команда, должна состоять максимум из 5-9 человек, тогда достаточно комфортно, удобно, и легко, будет контролировать работу этих людей. Скрам, подходит больше для малых проектов, и конечно же, для тех, которые несут в себе какую-то уникальность, там, где заказчик, изначально не может видеть все, абсолютно технические требования к своему продукту. То есть у него есть только определённое представление, как это будет выглядеть. Как мы уже говорили в уроке по эджайл — это больше всего применимо в стартапах и в каких-то инновационных проектах, по разработке it-продуктов. Также важно отметить, что скрам, больше подходит для опытных специалистов, потому что, они могут, самостоятельно организовать свою работу, а это один из важных принципов работы по данному фреймворку. Сегодня мы разобрали такой интересный фреймворк, поговорили про то, как его практически можно применить. Из каких ролей состоит команда, которая будет работать над it-продуктом, и в целом, как построить разработку, с помощью данного фреймворка, какие есть определённые правила, которых нарушать нельзя.