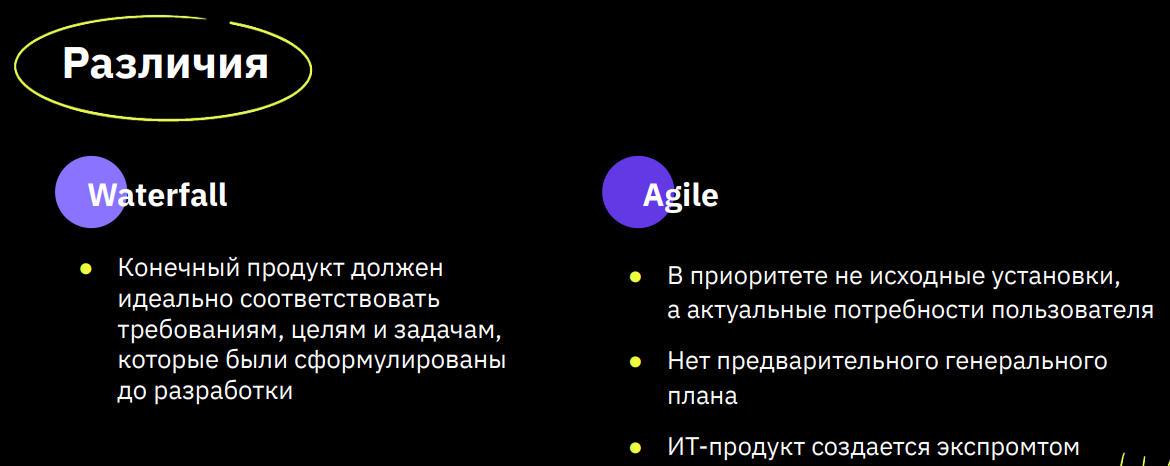
Лекция № 2. Agile

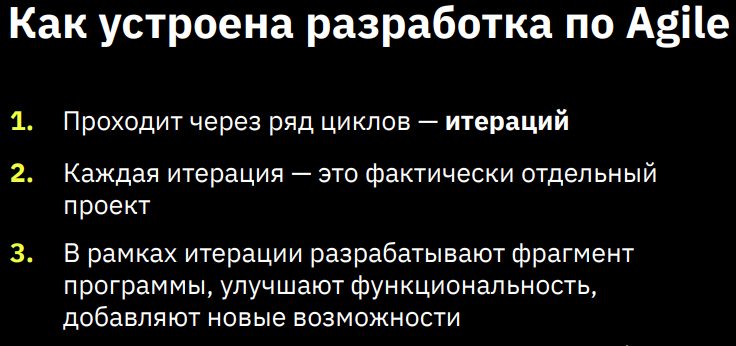
Поговорим про то, что такое Agile? Как это переводится? С какого языка? И как это применимо к IT-индустриям? Рассмотрим историю появления, возникновения, данной гибкой методологии? Поговорим в целом, про то какие в себя включает данная методология подходы, фреймворки и так далее? Обсудим недостатки и преимущества Agile? Также важный момент — это ценности, практики и принципы, которые вобрала в себя Agile философия?



Agile с английского языка переводится как гибкий, подвижный, быстрый, проворный, можно ещё сказать. Если мы говорим про IT-сферу, то конечно же правильный перевод слова Agile — это гибкие методологии. Agile методология позволяет нам динамично организовать разработку IT-продукта, при этом поэтапно определёнными итерациями поставлять какую-то ценность конечного продукта нашему заказчику. Для начала вспомним, что такое waterfall и каким образом строится разработка IT-продукта нашего ПО, сайта или любого другого IT-продукта. Как это всё строится с помощью waterfall, с помощью каскадной методологии? С начала выдвигаются требования к нашему продукту к нашему ПО, разрабатываются технические основы, технические характеристики. Затем все это воплощается в нашем коде. После того как мы получаем готовый продукт, мы можем его протестировать, передать нашему заказчику и получить обратную связь. То есть здесь мы видим, что обратную связь мы получаем только когда мы закончим всю разработку и у нас будет готовый продукт, который можно уже тестировать.

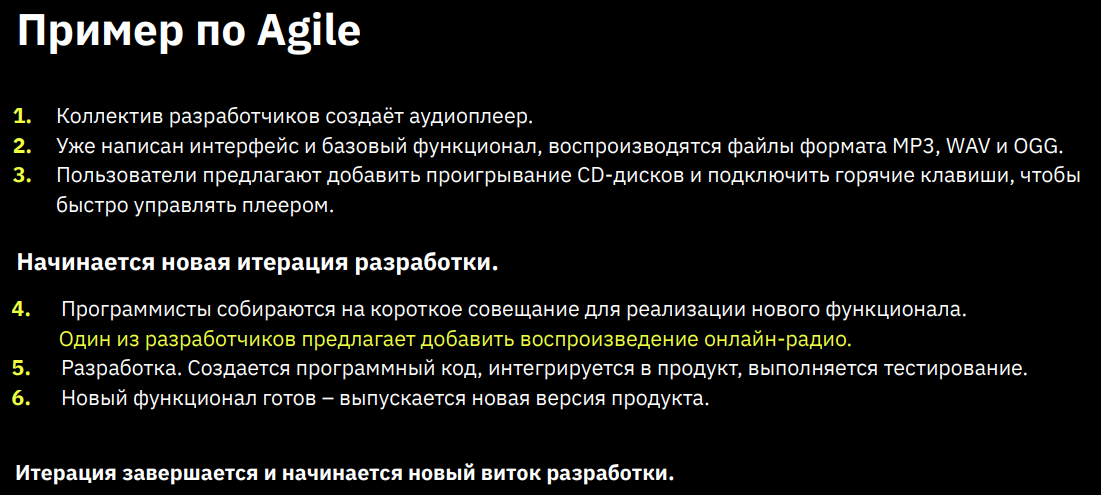


Естественно, есть такое основное различие между waterfall и Agile. Waterfall наша каскадная методология, предполагает то, что проект будет успешен, если мы будем следовать чётко нашему плану. Чутко, разработанному в самом начале техническому заданию. Если мы говорим про Agile, то здесь уже совсем всё по-другому, в приоритете находятся не те установки, которые мы ставим перед собой в самом начале, а актуальные потребности наших конечных пользователей, нашего заказчика. То есть те, кто будет в дальнейшем использовать наш продукт. Соответственно у нас нет какого-то предварительного большого генерального плана, согласно которого мы будем идти. И IT-продукт создаётся экспромтом, то есть проводится определённые итерации, в рамках которых разрабатывается части нашего общего большого IT-продукта.



Каким образом строится разработка по Agile? Во-первых, важно понять, что вся разработка делится на определённые итерации, то есть разработка происходит циклично, второй момент каждая итерация это не просто какой-то виток и набор работ в рамках каждой итерации у нас создаётся определённая версия продукта, то есть это фактически отдельный готовый продукт, который можно уже использовать. Очень часто это ещё называют минимальная версия продукта, либо последующие версии, последующие итерации нашего продукта. В рамках итерации у нас разрабатывается всегда целый полноценный фрагмент, то есть мы уже после завершения итерации можем конкретно протестировать, насколько это маленький фрагмент нашего, большого IT-продукта работоспособный. Насколько он соответственно содержит в себе какие-то баги, дать обратную связь команде разработчиков и затем уже запустить его в свет.

Теперь давайте рассмотрим пример, каким образом строится разработка IT-продукта по Agile? У нас есть какой-то определенный аудиоплеер, который уже включает в себя какой-то минимальный базовый функционал, он воспроизводит такие форматы как MP3, wave, ogg. Соответственно коллектив разработчиков собирается на небольшое собрание, дальше мы поговорим как это правильно называется с точки зрения Agile философии. И обсуждается предложения наших пользователей, которые говорят, что хотелось бы видеть в вашем аудио плеере: проигрывание CD дисков, была возможность подключить какие-то горячие клавиши, чтобы в целом было удобно пользоваться.

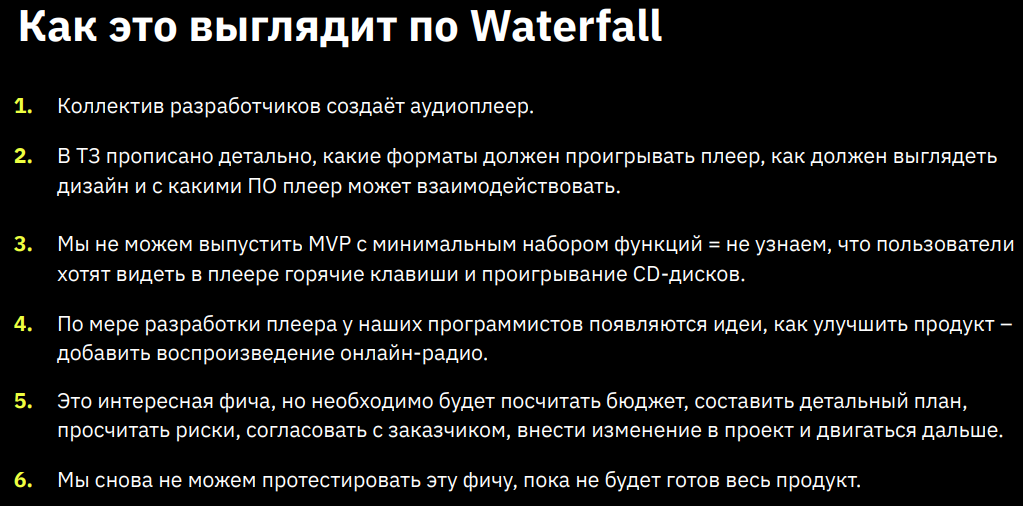


Разработчики всё это обсудили, приняли какое-то решение, выявили каким образом будет писаться код, как в целом будет происходить разработка. У нас стартует итерация, в рамках которой мы соответственно добавляем этот функционал в наш аудиоплеер.

При этом во время того, как программисты собираются на какие-то собрания, обсуждают то, как происходит разработка, у одного из них рождается классная идея это добавить онлайн-радио в наш аудиоплеер. Согласитесь, наверняка многим пользователям это очень сильно зайдёт? Предложение принимается у нас происходит разработка, то есть внедряется ещё помимо тех самых функций, которые предложили наши пользователи ещё и онлайн-радио. Мы получаем готовый продукт, затем его тестируем и выпускаем в свет.

Таким образом, мы можем получить обратную связь от конечных пользователей, посмотреть, насколько им зашёл наш новый функционал, новая фича и соответственно собираем данную обратную связь, проводим какой-то анализ и стартуем в следующей уже итерации, улучшаем постепенно наш продукт!

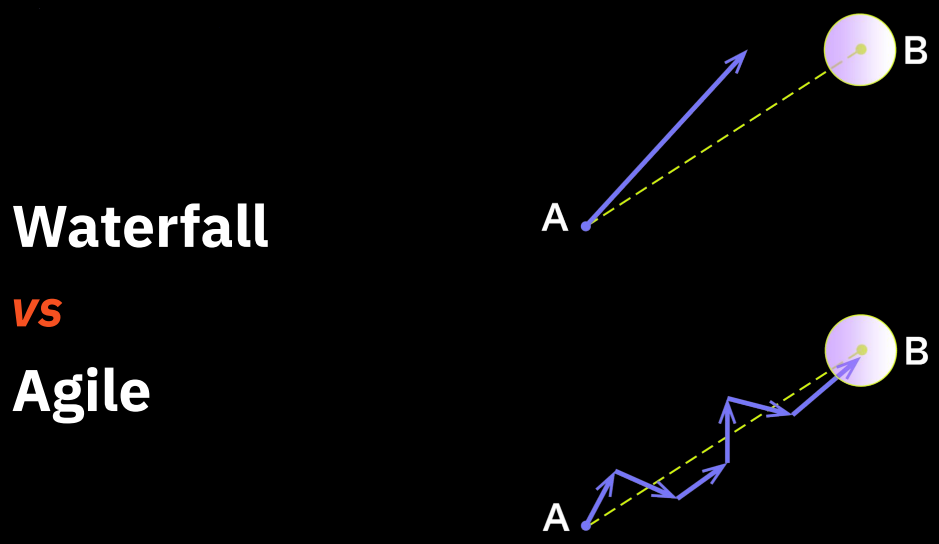
Как бы это выглядело по waterfall? Давайте смотреть и разбирать. У нас есть коллектив разработчиков, есть команда, которая создаёт аудиоплеер. Соответственно в самом начале необходимо достаточно детально проработать техническое задание, прописать какие форматы должен данный аудиоплеер воспроизводить. Можно ли туда подключать CD диски, можно ли включить туда какой-то дополнительный функционал и так далее?



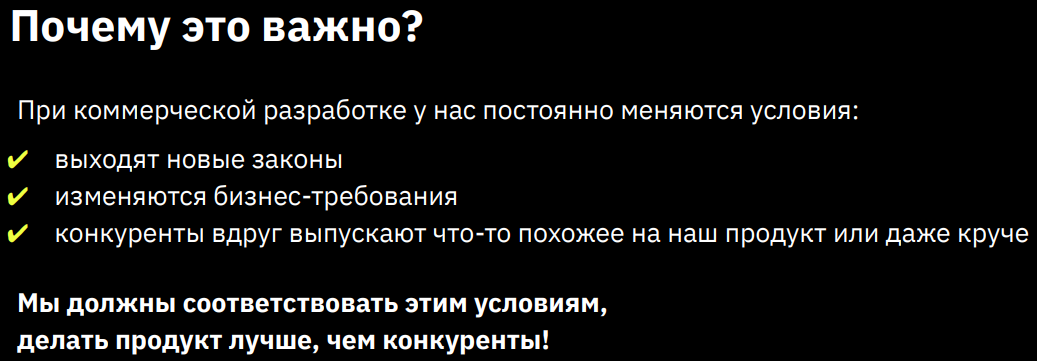
То есть всё это необходимо определить в самом начале. Согласитесь, что сделать это в современном мире не представляется всегда возможным. Почему? Потому что, возникают новые хотелки, новый запрос, у наших конечных пользователей. И соответственно наш продукт должен идти в ногу со временем. Идём по waterfall! Во-первых, мы не сможем выпустить сразу же минимальную версию продукта, то есть получить быструю обратную связь от конечных пользователей до того момента, пока мы не разработаем весь аудиоплеер, весь продукт со всеми его функциями, которые он должен в себя включать, который мы прописали изначально в своём техническом задании.

При этом по мере разработки у наших программистов обсуждается классная идея. Допустим это фича внедрить в данный аудиоплеер возможность подключения онлайн-радио согласитесь круто? Но для того, чтобы внедрить данную фитчу, по методологии waterfall, нам необходимо внести изменения в изначально установленное техническое задание, проработать все риски, рассчитать бюджет, пройти массу различных согласований, чтобы перестроить нашу разработку уже таким образом, чтобы получить функционал внедрение онлайн-радио. Зачастую бывает это сделать очень сложно, но тем не менее, фича интересная, хорошо! Тестировать эту фичу и получить обратную связь от конечных пользователей, сразу же мы не сможем. Мы должны её внедрить, закончить разработку всего нашего IT-продукта.

И только уже после этого его протестировать, собрать обратную связь и возможно мы увидим, что что-то из этого функционала не совсем зашло нашим пользователям. Согласитесь в современном мире это не совсем правильно, потому что рынок идёт вперёд таким образом мы можем потратить кучу денег, при этом не получить должного результата, которые хотели изначально. Давайте посмотрим на то, каким образом можно представить графически итеративность наших работ при разработке с помощью Agile методологий и с помощью waterfall, здесь вы видите, что на слайде изображены точки А и точки Б сверху мы видим, каким образом мы будем двигаться по каскадному методу методологии, разработки по waterfall, а снизу уже представлена Agile.

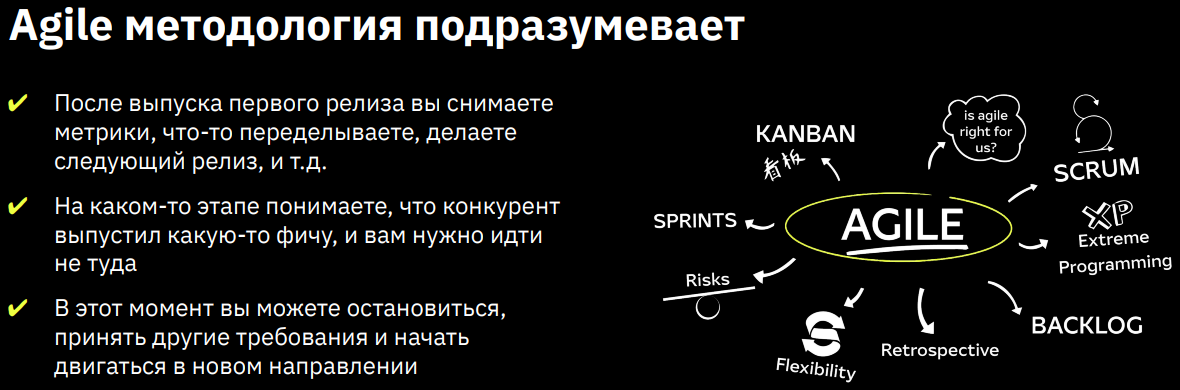


Давайте представим, что мы находимся в лесу. Мы находимся в точке А и должны дойти до точки Б, чтобы выйти из данного леса в которым мы оказались. Что мы делаем если мы идём Каскад Мы сразу же определяем нашу точку Б, где мы должны оказаться, берём в руки GPS и идём. Возможно, мы придём как раз-таки в правильную точку Б, но зачастую бывает не так, мы можем заблудиться можем ошибиться, как будет происходить, если мы будем двигаться по Agile. Мы будем делать небольшие шаги, сначала в одну точку, смотрим на наш компас, собираем метрики. Видим, что мы идём в правильном направлении, делаем следующий шаг, снова останавливаемся смотрим правильно ли мы двигаемся. Если нет корректируем свой путь, идём дальше, таким образом строится разработка любого IT-продукта по этим двум методологиям. Если мы находимся в точке А и принимаем решение разрабатывать по waterfall (каскадной методологии), мы должны изначально быть уверены, что наша конечная точка – это точка Б. Но в виду того, что, наша технологии очень сильно быстро развиваются соответственно мы не всегда можем предположить, что правильная конечная точка, это Б. Возможно она будет В, Г и так далее. То есть это совсем не то куда мы хотим прийти. Если мы смотрим на Agile. У нас тот же самый промежуток времени, мы находимся в той же точке А, мы разрабатываем малую версию продукта MVP, выбрасываем её на рынок. Собираем обратную связь, то есть останавливаемся, собираем метрики, делаем обзор и принимаем решение, каким образом стоит доработать наш продукт. Идём дальше дорабатываем, снова снимаем метрики, получаем обратную связь, из новой точки уже идём дальше. Таким образом мы можем прийти даже не к точке Б, а уже совсем в другую точку. Почему? Потому что внимание пользователей концентрируется на новых фичах. У нас на рынке есть очень много конкурентов, которые что-то выпускают и соответственно мы должны соответствовать этому рынку. Почему это важно?



При коммерческой разработке очень часто меняются условия нам важно быстро поставить продукт на рынок, чтобы получить ту самую необходимую обратную связь и не тратить попусту наш бюджет. Соответственно могут выходить новые законы особенно если мы проводим разработку в рамках какого-то финансового проекта, либо медицинского и так далее. Конкуренты очень часто нас опережает, к сожалению, это рынок и никуда от этого не деться, поэтому мы должны соответствовать этим условиям и постепенно разрабатывая какой-то определенный фрагмент нашего продукта, собирать метрики, собирать обратную связь от наших конечных пользователей, чтобы улучшить наш продукт и сделать его лучше, чем наши конкуренты.

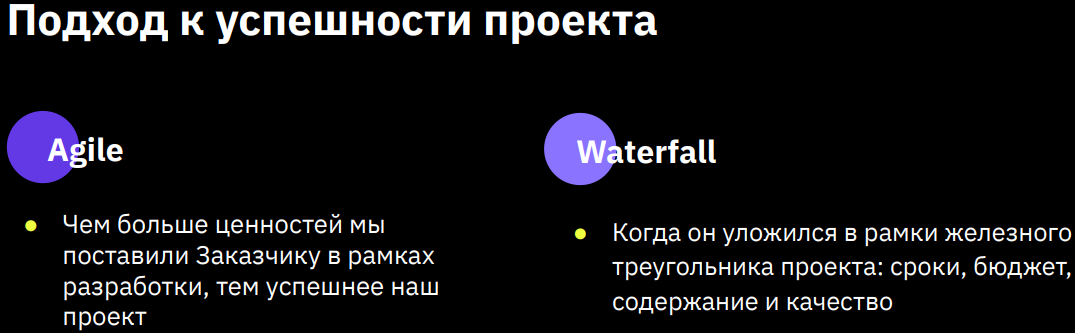
Что у нас подразумевает Agile методология, во-первых, после того как вы выпустили первый фрагмент продукта, обязательно соберите метрики, обязательно получите обратную связь и примите решение, что вам необходимо переделать, доделать, либо усовершенствовать, возможно исправить какие-то баги это очень часто помогает. Потому что мы не можем протестировать абсолютно всё и какие-то моменты вылезут уже у пользователей, которые непосредственно юзают ваш IT-продукт. Когда мы видим, что наш конкурент выпускает какой-то более лучший продукт, усовершенствованный, нам важно остановиться, поменять наши изначально установленные требования к нашему продукту. Понять куда мы движемся? И затем уже идти вперёд, дорабатывать этапами, фрагментами, итерациями наш IT-продукт.



Давайте посмотрим на подход успешности проекта, если мы говорим по Agile и, если мы говорим по waterfall, каким образом относятся данные методологии к успешности нашего IT-проекта, потому что любой продукт создаётся в рамках какого-то определённого проекта.

Если мы говорим про Agile, то здесь мы больше уделяем внимания той ценности, которую получает заказчик, то есть наш проект будет успешен будет успешен если мы поставим большую ценность нашему конечному заказчику.

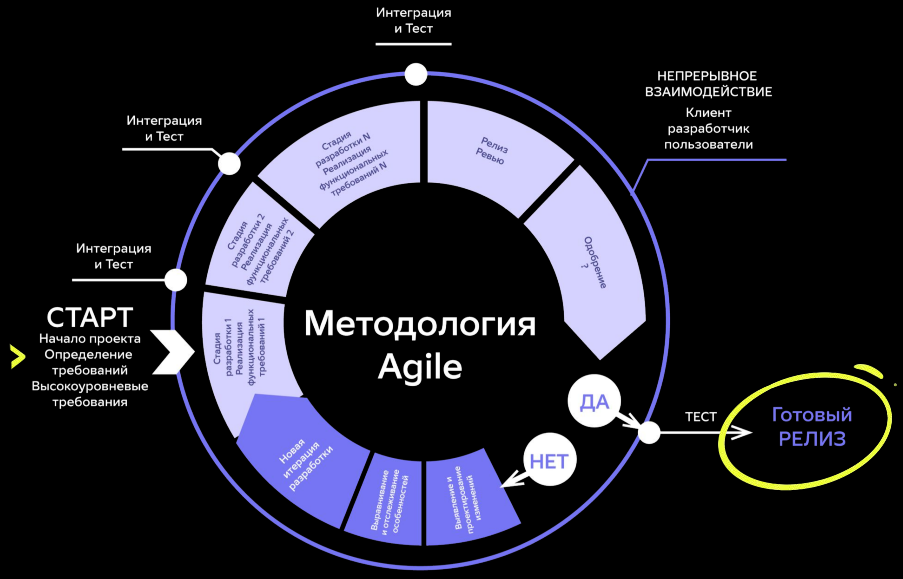
Если мы говорим про waterfall, в данном случае проект считается успешным, если мы следовали четко установленному плану, то есть изначально установленному техническому заданию.



Но согласитесь, как ранее мы уже говорили это не всегда представляется возможным прописать достаточно чёткие технические требования и предугадать куда будет двигаться рынок и чего захотят наши конечные пользователи.

Соответственно здесь становятся ярко выражены различные подходы к тому, как мы можем оценить успешность нашего продукта. То есть по Agile мы говорим, что нам важно поставить какую-то конечную ценность, для нашего заказчика, чтобы наш продукт был успешным, чтобы его полюбили пользователи и он был бы удобен в использовании. Waterfall уделяет больше внимания именно четкому следованию, каким-то правилам, определённому шаблону, установленному техническому заданию.

Сейчас вы видите на данном слайде графическое представление каким образом происходит разработка IT-продукта по методологии Agile.



В самом начале, когда ещё заказчик только говорит, что он хочет что-то получить, какой-то продукт, при этом не имеет чёткого представления как это должно функционировать, у него есть только лишь идея и какие-то минимальные требования к этому продукту.

Мы собираем все эти требования, стартуем нашу итерацию, разрабатываем небольшой фрагмент, проводим тестирование, поставляем данный фрагмент заказчику, он его смотрит, мы получаем обратную связь, стартуем следующую итерацию. После каждой итерации мы можем данный продукт по решению заказчика выпускать уже на рынок и конечно же собирать определённую обратную связь наших пользователей. Затем после того, как все итерации будут завершены, мы проводим релиз-ревью получаем одобрение и соответственно запускаем готовый релиз уже на рынок. Это может быть минимальная версия продукта, это может быть уже полностью готовый IT-продукт, который можно полноценно использовать.

История возникновения, откуда появился Agile и почему он так плавно пришёл в нашу жизнь, почему его так хотят многие разработчики использовать, при создании этих продуктов?

В феврале 2001 года 17 независимых программистов, которые использовали различные методологии свои разработки в продукт собрались в США, штате Юта отдохнули, расслабились и решили, что необходимо что-то модернизировать и создать какой-то документ, в котором прописать, основные понятия. Каким образом можно строить разработку и к чему это нас приведёт? Именно таким образом родился небольшой документ, который называется Agile манифест.

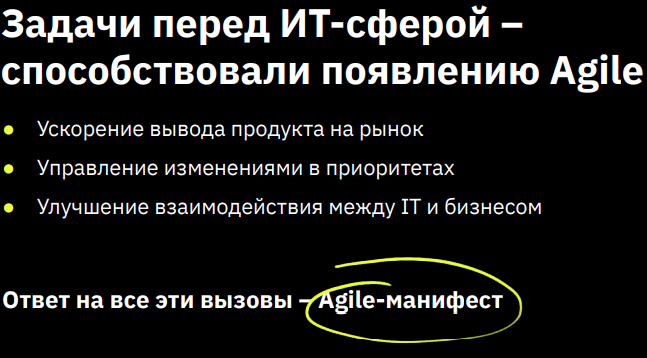
На слайде на данном вы введите ссылку, по которой можете перейти и посмотреть, как выглядят данный документ. Сейчас вы можете посмотреть на создателей, на эти счастливые лица программистов, которые стали про-родителями данной философии, философии Agile.



Agile манифест, что это за документ и что он в себя вобрал? Во-первых, это является основной такой документ, грубо говоря устав нашей Agile философии, включает он в себя 4 основные ценности, 12 принципов и соответственно данный документ был переведён на 50 языков, чтобы абсолютно любые специалисты со всех стран мира смогли его прочитать и применить уже на деле именно. Задачи, какие стояли задачи перед IT-сферой, когда произошло появление Agile, потому что они не просто так эти люди собрались, не просто так появился данный документ?

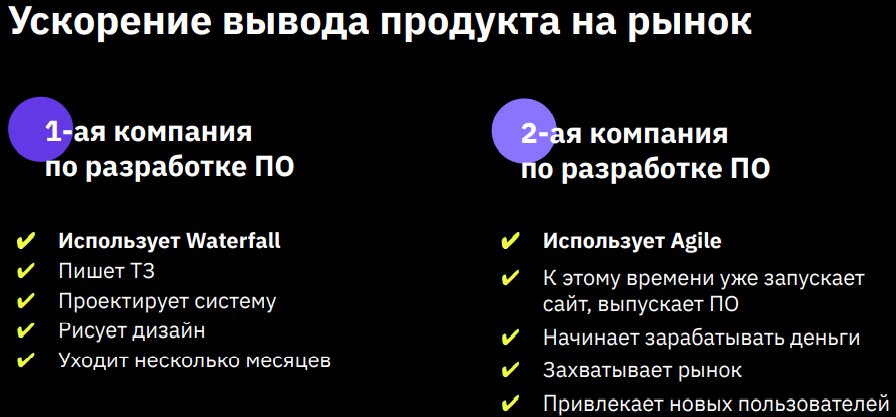


В то время перед IT-сферой стояли следующие задачи, во-первых, это ускорение вывода готового продукта на рынок. В управлении изменениями должно быть в приоритетах, нельзя всегда следовать какому-то шаблону, потому что технологии растут, технологии развиваются. И те времена, когда мы создавали эти продукты только для Нефтегаза, для какой-то оборонки они уже канули в лету, соответственно появляются новые стартапы, возникают новые идеи и их нужно претворять в жизнь.

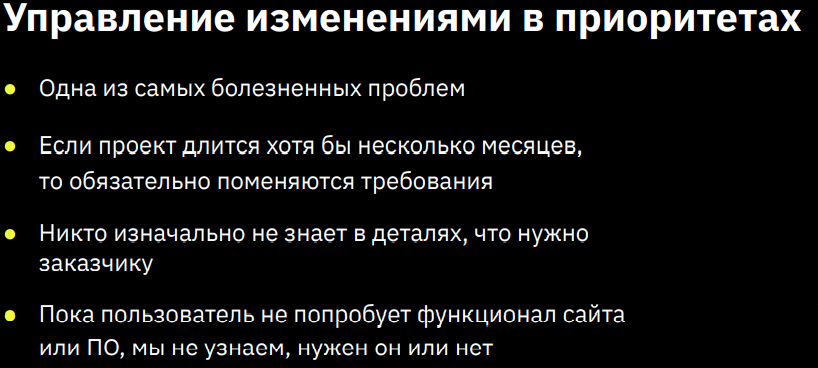


Следующая третья задача, это улучшение взаимодействия между ИТ и бизнесом. Теперь давайте посмотрим подробнее на все эти задачи и соответственно разберём, почему именно они привели к тому, что появился Agile манифест. Такое своеобразное решение этих всех 3 задач. 1 задача: это ускорение вывода продукта на рынок. Давайте представим, что у нас есть две компании которые занимаются разработкой программного обеспечения. Первая компания у нас выбирает waterfoll, она достаточно детально пишет техническое задание и передает его в разработку, программисты пишут код, проходит тестирование и через определённое время получается продукт. Допустим временные рамки у нас целый год, соответственно пока за этот год разработчики не напишут готовый код, готовый продукт, его невозможно будет выпустить в свет. Вторая компания решила взять Agile, что они делают, у заказчика есть какие-то минимальные требования к продукту, который он хочет реализовать, и соответственно происходит первая итерация. Разрабатывают минимальную версию продукта, запускают его на рынок, допустим этот продукт – это наш сайт. Он готов, его выпустили, затем уже принимается решение сделать приложение для Айфона и Android, соответственно разработчики стартуют данную операцию, получают мобильное приложение, тестируют и снова запускают на рынок.

Таким образом вторая компания начинает постепенно захватывать рынок и всё это происходит в рамках одного года. К тому моменту, когда первая компания (которая выбрала waterfall), получит наконец-таки тот самый готовый продукт, то самое ПО. Вторая компания уже захватит рынок, и она будет, так сказать, лидером и флагманом, который выпустил на рынок данный продукт. Здесь стоит отметить, что ни в коем случае нельзя говорить, что наша Agile философия и вот эти все гибкие методологии позволяют получить быстрее готовый продукт.

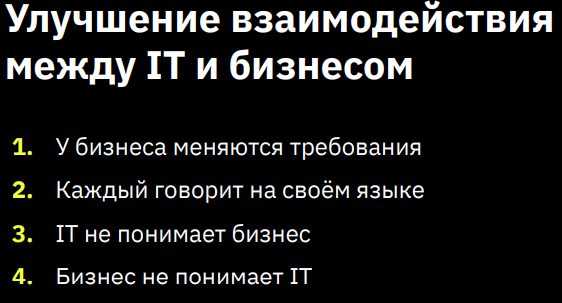


На самом деле и там, и там, временные рамки год, но только первая компания за этот год на рынок ничего не выпускает. Она планомерно постепенно разрабатывает продукт, тестирует и только после того, как будет готовое решение, выпустит его на рынок. Вторая компания идёт постепенно, как я уже говорил в первом примере, с точкой А и точкой Б, она делает шаг, снимает метрики, получает обратную связь, о ней уже слышат на рынке какие-то первые пользователи пусть даже это небольшой какой-то фрагмент продукта, который имеет базовый функционал, но тем не менее он уже вышел в свет. И люди начинают его использовать, затем этот фрагмент дорабатывают, выпускают следующую версию пользователя, ещё больше начинает любить, ваше ПО, ещё больше она получает распространение на рынке. Таким образом вы быстрее, захватываете рынок, вы быстрее получаете обратную связь от конечных пользователей, снимаете те самые метрики, улучшаете свой продукт. И к тому моменту, когда первая компания получит продукт, вы уже сделаете его гораздо совершеннее, гораздо лучше, гораздо круче, чем он получился у первой компании, потому что у вас была возможность сразу же всё тестировать, сразу же всё проверять прямо на ваших конечных пользователях.

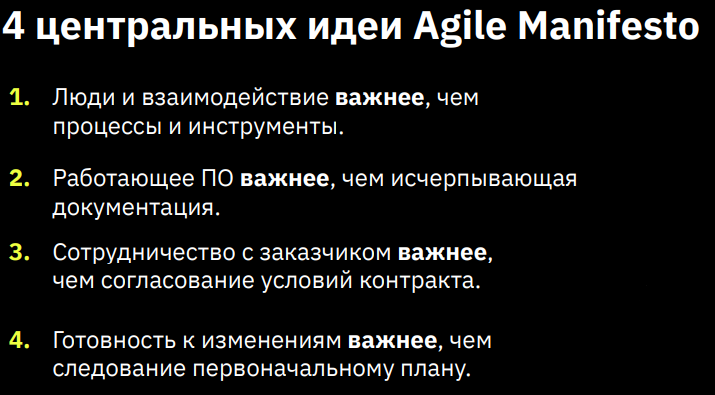


Вторая задача — это управление изменениями, которые должны быть в приоритете, на самом деле это одна из самых сложных задач, которую необходимо решить, как раз-таки waterfall нам не позволяет вносить какие-то изменения в изначально установленное техническое задание. Да, конечно, это возможно сделать если совсем полный аврал, и тот код, который изначально написали или то решение, которое изначально хотел видеть заказчик, оно не способствует тому, чтобы вы продолжали разработку, то есть вы застопорились, вы не можете двигаться дальше, у вас возникла какая-то ошибка, соответственно вам нужно внести изменения. Да, это возможно в методологии waterfall, но это очень болезненно. И это стоит больших денег, потому что необходимо переписывать очень много другого кода, который вы уже написали. Соответственно, если ваш проект длится хотя бы, больше двух-трех месяцев, неизбежны в нём изменения и это будет всегда!

Следующий момент — это то, что никто изначально не может предполагать в современном мире, как должен выглядеть готовый продукт, готовое приложение, готовый сайт? Потому что, постоянно выходит что-то новое, и необходимо адаптироваться под современные реалии, под наш рынок. Следующий момент — это то, что пока пользователь не увидит готовый продукт, не увидит сайт, приложения, ПО. Он не скажет, насколько это хорошо, на сколько это удобно и насколько ему это нравиться? А соответственно до этого момента мы будем, грубо говоря спускать деньги на ветер, если выберем каскадный метод разработки продукта.



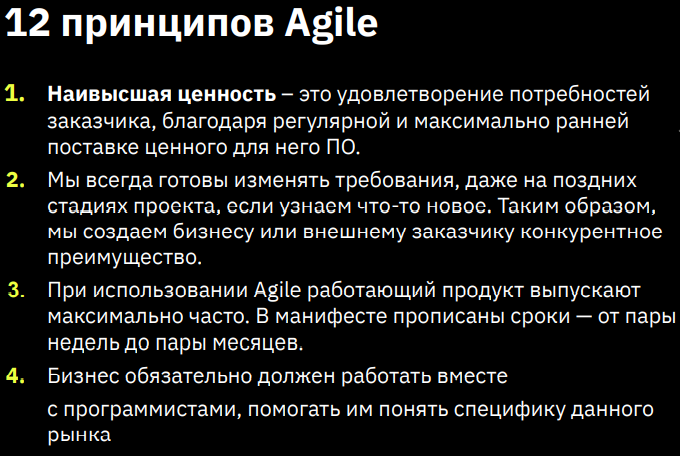
Третья задача — это улучшение взаимодействия между IT и бизнесом, на самом деле это очень распространённый вопрос и большая проблема, даже могу сказать по своей практике, когда мы разрабатывали определённое ПО на базе SAP для финансовых проектов, очень сложно было объединить технарей, финансистов, бухгалтеров и соответственно разработчиков. Поэтому у бизнеса меняются очень часто различные требования, бывает недопонимание между разработчиками и командой конечных пользователей. Именно поэтому важно, чтобы они работали все совместно, работали над созданием какой-то итерации, какого-то фрагмента вашего готового продукта. Ценности Agile? Важный момент, который был прописан у нас в Agile манифесте это четыре центральных ценности, четыре центральные идеи нашей гибкой методологии, даже я бы сказал наши гибкой философии, которая вбирает в себя определённые методологии для разработки ПО. Давайте посмотрим подробнее, что это за четыре центральные идеи, четыре основные ценности нашей гибкой методологии Agile? Даже я бы сказал не методологии, а целой философии, которая убирает в себя определённые практически применимые фреймворки и методологии.



Первая ценность — это люди и взаимодействие важнее чем какие-то процессы и инструменты. Вторая ценность — работающее ПО, важнее чем исчерпывающая документация. Третья ценность — это сотрудничество с заказчиком важнее, чем согласование условий договора. Четвёртая ценность – это готовность к изменениям важнее, чем следование изначально установленному плану. При этом многие новички, которые только начинают осваивать Agile думают, что левая часть, в которой находится до слова важнее, она главенствующая и про правую часть, которая уже находится после слова важнее - совершенно забывают. Нет ни в коем случае, мы ни в коем случае не умоляем значение нашей правой части, то есть мы не умоляем значение процессов, инструментов, значение документации, значение условий контракта, нет мы лишь говорим о том, что, в случае если будут возникать какие-то разногласия то левая часть, работающая ПО взаимодействия команды, следования интересам заказчика, его желаниям, она важнее чем какие-то договорённости, которые были описаны на бумаге в контракте, в нашем изначально установленном техническом задании. То есть если мы видим, что мы не придём к успеху используя наше первоначальное техническое задание, не стоит биться головой об стенку и разрабатывать наш продукт по изначально установленному плану. Важно его изменить, скорректировать и идти уже дальше.

Теперь давайте поговорим, про 12 принципов нашей гибкой философии Agile.

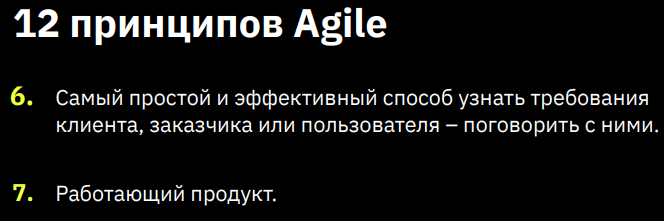
* Первый принцип — это наивысшая ценность удовлетворения потребностей нашего заказчика. Соответственно нам необходимо как можно раньше поставлять какой-то фрагмент ПО нашему заказчику, то есть поставлять определённую ценность и внедрять её на рынке.
* Второй принцип - мы всегда готовы следовать каким-то изменениям, вносить изменения в наш технический IT-продукт. Это необходимо для того, чтобы дать нашему заказчику определённое конкурентное преимущество на рынке.
* Третий принцип — это то, что при условии использования Agile работающий продукт у нас выпускает максимально часто. Обычно в Agile манифесте даже прописали что происходит выпуск какой-то версии продукта в пределах от пары недель до одного месяца. На самом деле на российском IT-рынке происходит всё гораздо даже чаще, бывает, что каждые несколько недель, каждую неделю, выходит что-то новое.
* Четвёртый принцип — это то, что бизнес должен всегда работать в паре с нашими разработчиками, с нашей командой. Чтобы не было никакого противовеса в одну или в другую сторону, чтобы было взаимопонимание между бизнесом и IT.



* Пятый принцип — это команда, команда вообще в Agile философии является таким своеобразным краеугольным камнем, потому что всё в разработке зависит от людей и от их настроя, от их подхода к нашему IT-продукту, к самой разработке, к самому процессу, к самому взаимодействию внутри команды. Здесь важно отметить, что руководитель, который возглавляет IT-разработку, он становится уже не просто менеджером, каким-то управленцем. Он является коучем, он начинает развивать свою команду приносить что-то новое, совершенствовать их работу и в целом отлаживать грамотное, хорошее взаимодействие между специалистами.



* Шестой принцип — это самый простой и эффективный способ взаимодействия с нашим заказчиком, это просто поговорить с ним. На самом деле, просто поговорить зачастую бывает недостаточно. Важно визуализировать требования заказчика то, что он хочет видеть от нас в конце приводить это всё в единую плоскость. Именно для этого есть определённые инструменты в Agile методологиях, такие как доски, такие как стикеры и так далее. Но об этом чуть позже мы поговорим, когда будем разбирать соответствующее практически применимые фреймворки и методологии.
* Седьмой принцип — это работающий продукт. Здесь необходимо отметить, что измерять готовность продукта важно не просто в процентном соотношении, от того сколько сделано работы, то есть допустим мы сделали 50% какой-то работы, на 50 % отрисовали дизайн и так далее. Важно сместится немножко в другую сторону, важно понимать, насколько работоспособную версию, насколько работоспособный фрагмент нашего продукта мы получили.



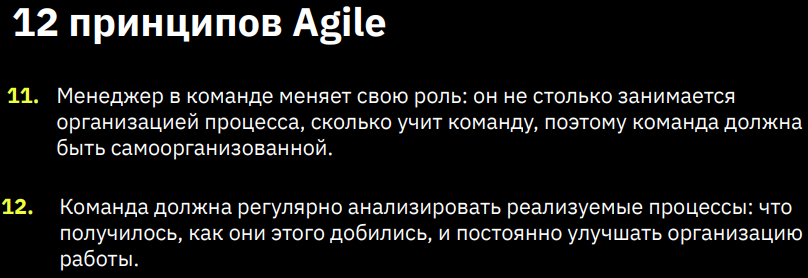
* Восьмой принцип — это то, что в Agile важен ритм, потому что всегда вносить какие-то улучшения изменения в свой продукт нужно двигаться в ногу вместе с нашим рынком, в ногу с теми самыми запросами наших конечных пользователей.
* Девятый принцип заключается в том, что ваш Agile никогда не будет работать, если у вас плохо написан код, если он в целом кривой. Почему это важно? Так как Agile философия предполагает, что у нас вносятся постоянные изменения в наш IT-продукт, в его архитектуру, его дизайн, его код наполнение и так далее, нам важно изначально качественно продумать всю архитектуру, чтобы не приходилось делать какие-то костыли, они так и называются, когда мы проводим разработку. Именно костыли? Временное решение, временное технологические решение, для того чтобы у нас что-то работало.



* Десятый принцип заключается в том, что важно соблюдать простату, если мы хотим поставить нашему конечному заказчику в конечном пользователю какой-то продукт и поставлять его соответственно частями то есть, того самого большого слона по частям. Важно начинать с самый простой, самой малой части ничего не замудрить, ничего не придумывать.



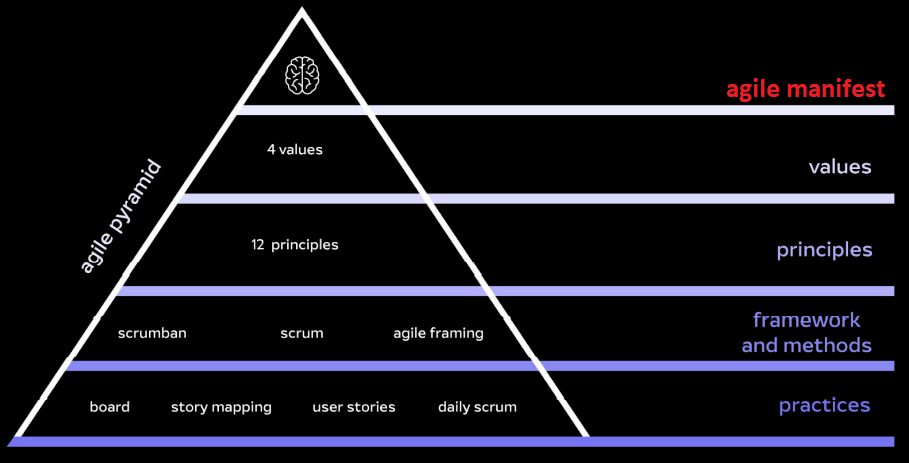
* Одиннадцатый принцип заключается в том, что менеджер с точки зрения Agile, он в целом меняет свою роль, он не столько занимается организацией работы, сколько учит команду, учит как ей необходима самоорганизоваться для того, чтобы получить готовый IT-продукт, именно поэтому зачастую роль менеджера называется немножко по-другому, когда мы используем Agile методологию. Это коуч, это мастер и другие различные названия, которым мы тоже рассмотрим чуть позже, когда будет говорить, про применимые фреймворки, методы подхода и методологии.
* Двенадцатый принцип заключается в том, что команда должна регулярно анализировать свою деятельность для того, чтобы её улучшать, потому что в целом философия предполагает постоянное улучшение. Улучшение не только готового IT-продукта, но и в целом работы команды, чтобы, когда она переходила на последующие уже проекты, она вбирала все те самые полученные уроки с предыдущих проектов, и делала разработку гораздо лучше и качественнее.



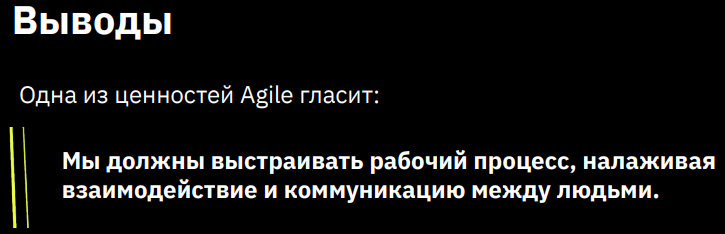
Теперь хотелось бы отметить, что, руководствуясь только этими идеями, принципами и подходами невозможно разработать какой-то IT-продукт. Согласитесь, даже если мы всё это детально пропишем, обговорим со всеми членами команды, с нашим заказчиком, с пользователями, всё равно IT-продукт мы не разработаем. В целом Agile — это не просто методология, это как я уже ранее говорил целая философия, которая вбирает в себя определенный набор практически применимых фреймворков и методологий, таких как Scrum, Canban, XP и так далее. Какие именно, поговорим позже.



При этом важно понимать, что Agile философия предполагает разделение большого слона на маленькие части, то есть большого готового IT-продукта, который мы хотим получить в конце на определенной итерации, в рамках которых мы получаем фрагменты малую версию продукта и поставляем её нашему заказчику. То есть главная цель быстрее поставить заказчику ценный фрагмент нашего продукта, для того чтобы он мог его оценить, мы смогли получить обратную связь, что-то измерить, что-то улучшить, и идти дальше.



Здесь сейчас вы видите ту самую пирамиду Agile, во главе неё находится Agile манифест, который создали те самые наши программисты, собравшись, отдохнув и прописав данный документ, обязательно посмотрите его по ссылке. Изучите, тем более он переведён на 50 языков, не зря это было сделано. Затем у нас идёт четыре ценности, после них двенадцать принципов, после идут практические фреймворки, методики и методологии. Как всё это применять наделе, как применять эти четыре идеи, двенадцать принципов и как в целом построить всю разработку нашего продукта? И конечно же практики, те самые инструменты, те самые подходы, как это прямо практически использовать при разработке, при выстраивании процессов, при выстраивании и совершенствовании своей команды разработки? Давайте как раз-таки поговорим про эти самые практики, без которых у нас не получится что-то разработать. Одна из ценностей гласит, что мы должны выстраивать рабочий процесс, налаживая коммуникацию внутри нашей команды.



Давайте посмотрим пример, как у нас это всё происходит в жизни: утром команда разработчиков собирается на определенное небольшое совещание, в Agile философии оно называется standup, соответственно на данном совещании команда разработчиков синхронизирует свою работу обсуждают, что было сделано вчера, что необходимо сделать сегодня и каким образом двигаться к нашей конечной цели, конечной цели определённой итерации, получить ту самую минимальную версию продукта, небольшой его фрагмент, либо просто улучшенную версию вашего IT-продукта. Можно, конечно, на данных standup или вместо них писать какие-то отчёты, о том, что ты сделал вчера, позавчера, что ты будешь делать завтра, сегодня и так далее. Но в таком случае это будет уже не совсем Agile, это будет не та философия которой мы говорим: «Почему?». Потому что у нас появляется много малозначительной документации, различные отчеты, которые сами по себе не несут в себе какую-то информацию, не несут какую-то практическую применимость. То есть получается лишний документооборот, лишний поток ненужной нам информации. Именно поэтому лучше это сделать, обсуждая с командой, свои цели, задачи, то, что ты сделал, то, что ты будешь делать в рамках такого небольшого standup, длится он обычно минут 15-20, соответственно всё это обсуждается, быстро обговаривается и люди приступают к работе. В следующий момент это часто используемые практики в Agile мы рассмотрели пример, где обговорили, что такое standup.

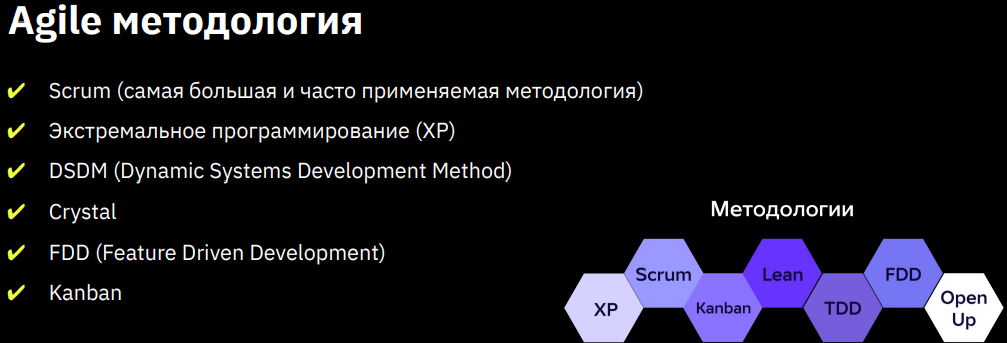
Вторая практика — это планирование итерации, то есть каждый итерация вбирает в себя определенный объём работ, которые необходимо выполнить в рамках неё и получить небольшой фрагмент продукта.

Unit-тестирование, после каждой итерации, после того как мы разработали тот самый фрагмент, его необходимо протестировать, не просто протестировать на работоспособность, а насколько данный фрагмент интегрируется в то что мы уже сделали до этого. Насколько правильный код, насколько правильные были приняты решения.

Четвёртая практика – это планирование релизов, то есть необходимо планировать большими кусками, какую из этих версий какую из этих фрагментов вы будете поставлять на рынок, что у вас пойдёт в свет.



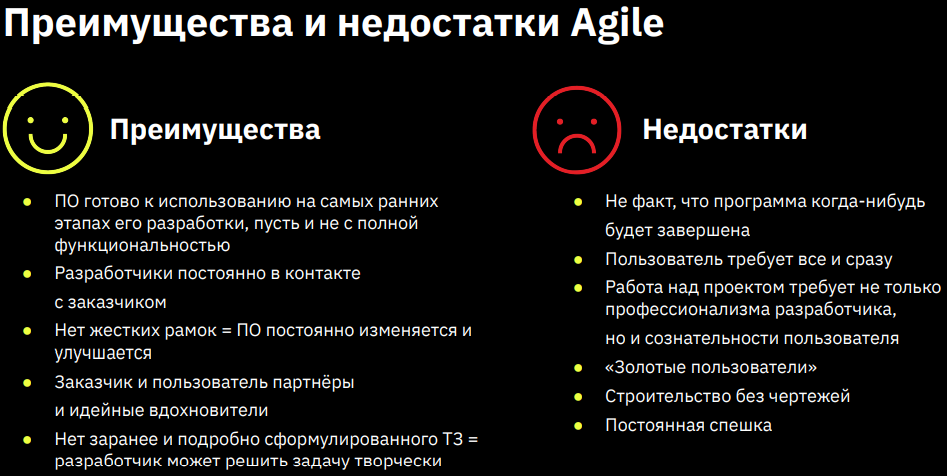
Исходя из этого, уже строить свою работу, по разработке продукта. Какие же есть практически применимые Agile методологии фреймворки и так далее, которые нам позволяют следовать той самой Agile философии, основанной на четырёх идеях, двенадцати принципах, для того чтобы получить качественный продукт.



Во-первых, это Scrum очень распространённая гибкая методология, даже её называют фреймворком, на соответствующем уроке мы поговорим. Какие она включает в себя практики, как её действительно можно применить при разработке?

Второй элемент — это Kanban, это такой своеобразный тренд, который потихоньку входит в нашу жизнь и соответственно используются при разработке.

Также есть XP экстремальное программирование, DSDM метод, Crystal, FDD и так далее на самом деле очень много различных подходов, очень много различных практик. Мы будем рассматривать только наиболее применимые, которые действительно работают при разработке продукта. Теперь хотелось бы отметить, какие есть преимущества и недостатки у нашей Agile философии.



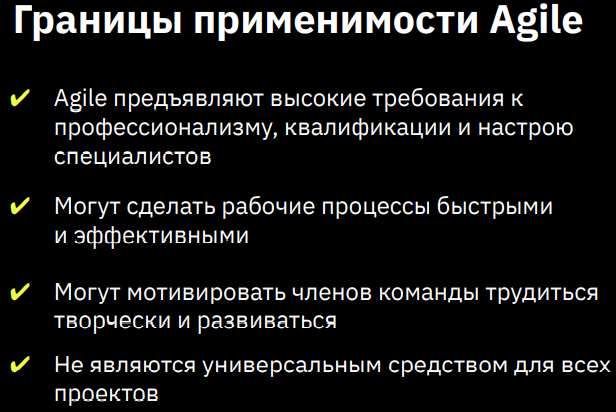
Для начала преимущества:

Самое важное преимущество заключается в том, что наше ПО готово для поставки на рынок, для поставки конечному пользователю ещё на самых ранних этапах, пусть это будет минимальная версия продукта даже с базовым каким-то функционалом. Но тем не менее её уже можно использовать, её уже можно юзать, тестировать и что-то с ней делать. Согласитесь, это достаточно круто!

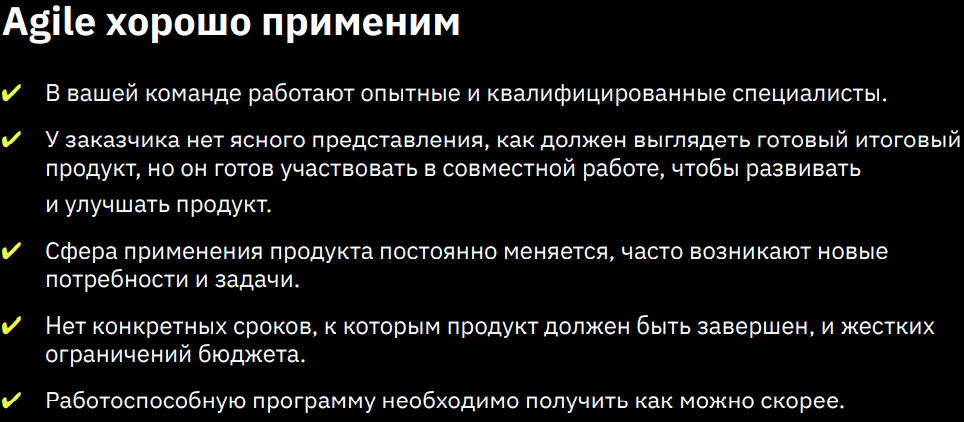
Второе преимущество — это то, что разработчики находятся в постоянном контакте с заказчиком, то есть здесь происходит такое смещение заказчик, конечные пользователи и разработчики. Это не просто определённые роли, это партнёры, которые взаимодействует для того, чтобы получить действительно качественный продукт. Если мы говорим про каскадную методологию, то там заказчик ставит задание, обрисовывает техническое задание, прописывает определённые требования и только после этого уже разработчики начинают что-то делать, при этом происходит отрыв. Пока разработчики не получат готовый продукт они не начинают взаимодействовать со своим заказчиком. Если мы говорим про гибкую методологию, здесь такое взаимодействие происходит постоянно, итерация за итерацией. И даже во время этих самых итераций если возникают вопросы, то команда может обратиться к заказчику, может его спросить? И тем самым сделать правильный выбор с точки зрения технического решения, с точки зрения реализации какого-то момента в своём фрагменте. Следующий момент — это то, что жёстких каких-то рамок ПО постоянно изменяется, постоянно улучшается, то есть у нас главная цель - не следовать изначально установленному плану, а получить действительно, качественный продукт, который будет соответствовать требованиям и ожиданиям нашего заказчика. Следующий момент заключается, в том, что у нас нет изначально сформулированного технического задания. На самом деле, это и плюс и минус — это очень двоякое мнение, потому что когда мы представляем, что мы хотим получить, нам очень трудно куда-то двигаться, но тем не менее, если мы говорим про какие-то новшества, про какие-то нововведения стартапы и так далее, нам совершенно невозможно изначально получить то, что мы хотим получить. У нас есть какая-то идея, мы её реализуем, воплощаем в коде, обрисовываем дизайн, получаем какое-то приложение, сайт, ПО или любой другой IT-продукт, смотрим, тестируем и идём дальше, постоянно итерация за итерацией улучшаем его.

Теперь про недостатки:

Есть большой недостаток, применение Agile философии, Agile гибких методологий, которые зачастую критикуют различные противники, применение данной философии. Заключается он в том, что не факт, что наш какой-то продукт, который мы изначально задумали сделать, всё-таки будет реализован, то есть не факт, что когда-то разработка у нас завершится. Почему? Потому, что мы постоянно итерация из итерацией добавляем что-то новое и мы не видим тот самый конец, когда мы получим совершенный продукт, который будет удовлетворять всем требованиям. На самом деле, невозможно получить такой продукт! Потому что, постоянно возникают новые запросы, новые хотелки, новые ожидания у наших конечных пользователей. Зачастую бывает так, что пользователь требует всё и сразу, то есть изначально хочет, чтобы было создано какое-то ПО, которое вбирает в себя абсолютно все функции с точки зрения Agile философии сделать это всё сразу нельзя, важно идти постепенно. Именно поэтому, когда вы заканчиваете первую итерацию и говорите, что у меня готов допустим, только лишь дизайн. Я бы хотел, чтобы вы посмотрели и оценили его, заказчик начинает негодовать и говорить: «А где же все те самые требования, которые я установил тебе в самом начале, я уже хочу видеть прямо готовое приложение, открыть его со своего смартфона, и что-то там сделать? А Вы ему говорите, извините, но у нас готов только дизайн — это первая итерация. Оцените её, посмотрите, насколько это правильно и дальше мы будем дорабатывать наш продукт». Следующий недостаток заключается в том, что работа над проектом требует большого профессионализма от команды, у вас не получится построить работу разработчиков, работу программистов по Agile, если там будут, грубо говоря только новички. Очень важно уделять внимание именно квалификации специалистов, потому что уже квалифицированный специалист, который имеет определённый бэкграунд, он может выстроить свою работу, самоорганизовываться и получить действительно ценность в рамках данного проекта. Очень часто бывают ещё золотые пользователи их так называют. Почему? Зачастую, когда вы работаете над IT-продуктом. У вас есть несколько заказчиков и обсуждения технических требований, каких-то параметров к вашему продукту, происходит во время большого совещания, где участвует 2-3-4 заказчика, даже бывает, что создают целые форумы, когда хотят реализовать какой-то глобальный проект и получить какой-то действительно глобальный, масштабный продукт. Что здесь возникает, активные заказчики, которые заинтересованы они начинают перетягивать одеяло на себя и соответственно могут повести разработку совсем по-иному пути, реализовывая свои хотелки, свои представления о том, как должен выглядеть продукт. А те самые неактивные заказчики они уходят в тень, где-то с чем-то соглашаются, где-то что-то добавляют, но тем не менее гораздо менее активно участвуют чем те первоначальные заказчики, про которых я говорил. Здесь, конечно, важна работа в целом руководителя команды разработки, чтобы не пустить свою разработку не по правильному пути. А идти действительно к намеченной изначальной цели. Следующий момент — это строительство без чертежей оно прямо так и называется - этот недостаток Agile философии. Почему? Допустим, мы строим какой-то большой небоскрёб, при этом не представляя, как он должен выглядеть, мы идём этапом за этапом, у нас нет какого-то генерального плана, генерального чертежа, как мы должны построить этот небоскрёб? Мы делаем сначала фундамент, первый этаж, второй, третий и так далее. Что здесь плохо? Здесь могут возникнуть такие моменты, что когда-то на определённом этапе, когда мы будем создавать наш продукт у нас могут возникнуть разночтения с изначально прописанным кодом, то есть возникают какие-то нестыковки и нам придётся делать временные костыли, временные решения, для того чтобы всё работало. Часто бывает, что, сделав такой костыль, люди к нему не возвращаются, работает и хорошо переделаем, когда что-то сломается. Почему это происходит? Потому что это Agile философия предполагает ритм, то есть происходит постоянная спешка, у нас постоянно возникают какие-то новые идеи и у нас нет времени их достаточно детально прорабатывать, обговаривать. Мы хотим их практически применить, сделать, разработать, написать какой-то код, внедрить его, отрисовать дизайн, посмотреть и уже получить обратную связь, работает это или нет. Соответственно исходя из данных недостатков и преимуществ важно выделить что Agile философия и Agile гибкие методологии — это не является панацеей, на самом деле они не могут решить абсолютно все проблемы, которые возникают при разработке.

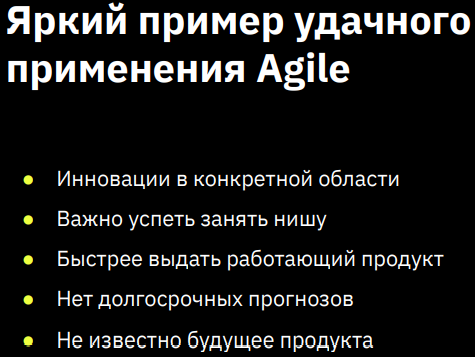


Соответственно есть границы применимости Agile, то есть, где их можно применить в данной методологии, а где лучше не следует. Agile предъявляет высокие требования к профессионализму, квалификации ваших сотрудников, потому что необходимо, чтобы люди самоорганизовывались. Если у вас новички, выстроить работу по гибким методологиям становится гораздо сложнее. В следующий момент Agile методологии могут сделать быстрыми и эффективными в целом ваши рабочие процессы, всё это из-за того, что вы гораздо быстрее получаете какой-то фрагмент продукта и можете получить обратную связь, соответственно улучшать свои рабочие процессы и строить свою дальнейшую разработку исходя из полученной обратной связи, исходя из полученных отзывов, неважно как это называется. Agile методология могут мотивировать членов вашей команды трудиться, творчески развиваться, потому что все новые идеи, нововведения, какие-то новшества, которые могут привести ваши разработчики, люди из вашей команды они только приветствуются. То есть нет каких-то жёстких ограничений и требований, исследования изначально установленному плану. При этом помните, что Agile всё-таки это не универсальный инструмент, решения всех проблем. Да, это действительно круто! Но важно правильно эти методологии использовать, только тогда вы сможете получить действительно качественный IT-продукт. Agile достаточно хорошо применим в целом данная философия, данный подход, если у вас:



Во-первых, есть опытные квалифицированные разработчики, члены вашей команды, у заказчика нет чёткого, ясного представления о том, каким должен конечный IT-продукт, что он хочет видеть в конце у него есть только идея и какие-то минимальные требования. Сфера применимости данных технологий постоянно меняется.

Третий момент, когда применим Agile — это когда мы разрабатываем свой продукт в сфере, которая постоянно меняется, то есть изменяются требования, рынок идет вперед, соответственно в данном случае лучше всего применить Agile философию гибкий подход к разработке. Когда у вас нет конечных сроков, когда заказчик готов с вами работать, готов улучшать продукт, готов взаимодействовать — это действительно хорошо, это круто, если заказчик только за и он готов платить за эту разработку. То почему бы не заняться данным продуктом? Agile методологии очень хорошо применять, когда у вас есть такая необходимость поставить на рынок работоспособную версию продукта гораздо быстрее, то есть захватить тот самый рынок, про который мы говорили во всех наших примерах получить небольшой фрагмент нашего IT-продукта, поставить его, собрать обратную связь, и идти дальше.



Ярким примером, наиболее успешного применения Agile методологии является стартапы, это именно те самые проекты, в рамках которых нужно создать уникальный IT-продукт, быстрее занять рыночную нишу, выйти в свет, получить что-то работоспособное, собрать обратную связь от ваших конечных пользователей, улучшить код, доработать его и идти дальше. Именно поэтому, здесь мы видим, что когда мы делаем какое-то стартап, когда мы создаём уникальный IT-продукт у нас нет времени что-то прогнозировать, мы не видим, каким будет конечный продукт, мы видим лишь только сейчас. Что требует рынок, что мы хотим запустить, какая у нас есть идея? Именно поэтому здесь важно подходить гибко к разработке нашего продукта. Подытоживая всё вышесказанное хотелось бы отметить, что Agile — это целая философия, которая вбирает в себя определённые принципы, определенные ценности и идеи, то есть определённые подходы вашей разработки. Ещё раз хочу заметить, что, применяя только лишь данные принципы, идеи, подходы, невозможно разработать сам IT-продукт. Нужны практические методологии, как раз-таки про них мы и будем говорить на наших последующих уроках.