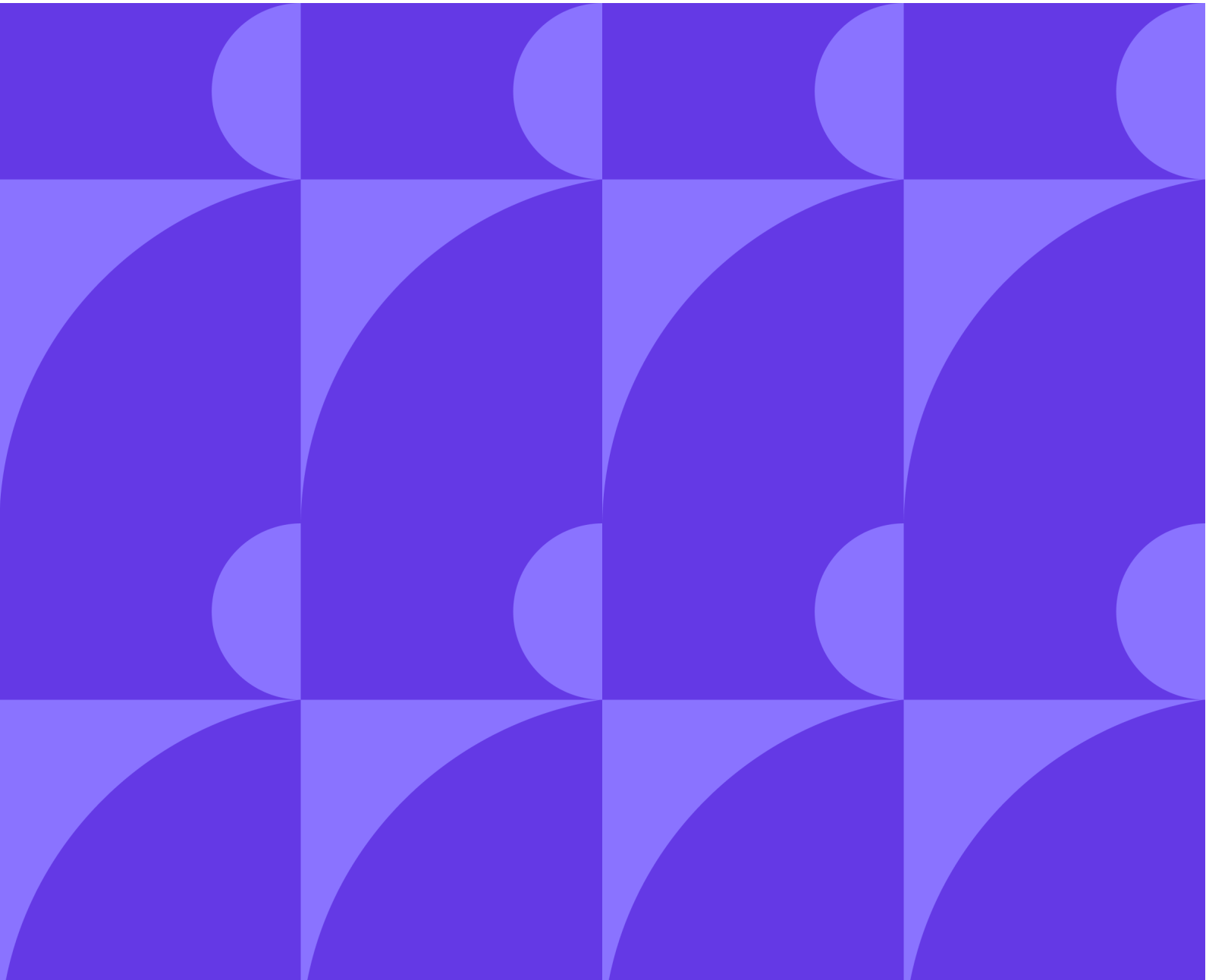
Гибкие методологии

Agile



На этом уроке

1. Узнаем, что такое Agile и историю его появления.
2. Поговорим про ценности и принципы Agile.
3. Разберём российские и международные практики применения Agile.
4. Узнаем границы применимости Agile.

Оглавление

На этом уроке

Оглавление

Глоссарий

Теория урока

Что такое Agile

История Agile

Появление и предназначение Agile

Ценности Agile

Принципы Agile

Практики

Agile-методологии

Преимущества Agile

Недостатки Agile

Границы применимости Agile

Используемые источники

Глоссарий

IT (Information technology) – информационные технологии

ПО – программное обеспечение

ТЗ – техническое задание

Теория урока

1. Что такое Agile
2. История Agile
3. Появление и предназначение Agile
4. Ценности Agile
5. Принципы Agile
6. Практики
7. Agile методологии
8. Преимущества Agile
9. Недостатки Agile
10. Границы применимости Agile

Что такое Agile

С английского *agile* переводится как «подвижный, быстрый, проворный». Но если говорить об IT, лучшим переводом будет слово «гибкие» — «гибкие методологии».

Agile-методология динамично организует разработку IT-продукта, постоянно адаптируя проект к новым обстоятельствам и требованиям. Это мы уже знаем, дальше — больше.

Вспомним, как устроена разработка по методологии Waterfall:

1. Выдвигаются требования к ПО, разрабатывается техническое задание (ТЗ).
2. Поставленные задачи воплощаются в коде.
3. Выполняется тестирование ПО.

4. Готовое ПО внедряется в работу.

Теоретически в Waterfall возможен возврат на предыдущие ступени, например, если оказывается, что ту или иную задачу невозможно выполнить по техническим причинам. В таком случае ТЗ пересматривают, но это скорее чрезвычайная ситуация. В норме конечный продукт должен идеально соответствовать требованиям, целям и задачам, которые были сформулированы до разработки.

⚠️ **Внимание!**

При использовании Agile-методологии в приоритете не исходные установки, а актуальные потребности пользователя. Постоянно вносить изменения в ТЗ, даже в самый разгар разработки, для Agile нормально. В гибкой методологии не предусмотрен предварительный генеральный план, скорее наоборот, IT-продукт создаётся практически экспромтом.

Разработка проходит через ряд циклов — итераций. Каждая итерация — фактически отдельный проект, где разрабатывают фрагмент программы, улучшают функциональность, добавляют новые возможности.

Пример. Коллектив разработчиков создаёт программный аудиоппроигрыватель. Уже написан костяк кода: готовы интерфейс и базовая функциональность. Программа умеет воспроизводить файлы формата MP3, WAV и OGG. Но пользователи предлагают добавить проигрывание CD-дисков и подключить горячие клавиши для быстрого управления плеером.

Начинается новая итерация разработки. Программисты собираются на короткое совещание: обсуждают задачи, распределяют их и вырабатывают способы решения. Один из разработчиков предлагает добавить воспроизведение онлайн-радио.

Следующий этап — непосредственно разработка продукта. Она может занять от нескольких дней до недель. В этот период создаётся и интегрируется в продукт программный код, выполняется тестирование.

Когда новая функциональность полностью готова к работе, формируется очередная версия программы и исполняемый файл отправляется к пользователям. На этом итерация завершается — начинается новый виток разработки.

Давайте посмотрим, как бы это всё реализовывалось по водопадной методологии.

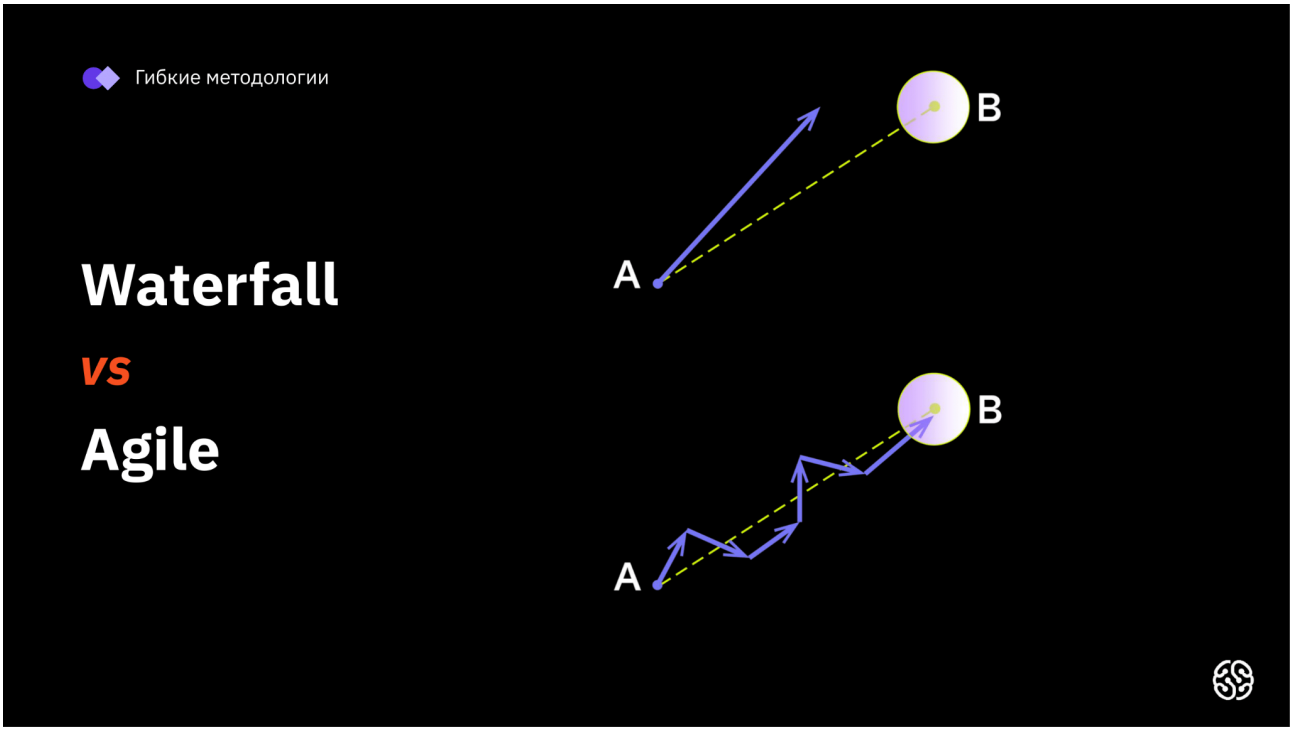
Коллектив разработчиков создаёт аудиоплеер. В техническом задании прописано детально, какие форматы должен проигрывать плеер, как должен выглядеть дизайн и с какими ПО плеер может взаимодействовать.

В отличие от Agile-подхода, руководствуясь водопадной моделью, мы не можем выпустить минимальную версию продукта с минимальным набором функций, а значит, не узнаем,

например, что пользователи хотят видеть в плеере горячие клавиши и проигрывание CD-дисков.

По мере разработки плеера у наших программистов появляются идеи, как улучшить продукт. Например, один из разработчиков предлагает добавить воспроизведение онлайн-радио. Все понимают, что это интересная фича и она может заинтересовать конечных пользователей, но, чтобы её реализовать, необходимо будет посчитать бюджет на её реализацию, распланировать тщательно все сроки, провести анализ рынка, согласовать всё это с заказчиком, внести изменения в проект, что само по себе достаточно сложно сделать в водопадной модели. И только после этого команда сможет приступить ко внедрению этого изменения в проект.

На основании этих примеров можно графически представить, как происходит разработка по методологиям «Водопад» и Agile:



Нам надо дойти из точки А в точку Б. «Дойти» — значит «получить обратную связь». Допустим, у вас есть GPS, вы можете дойти до какой-то точки и свериться, правильно ли вы пришли. Водопадная модель предполагает один большой шаг: вы куда-то пришли, выпустили на рынок продукт и только после этого понимаете, то ли вы сделали.

Итеративная модель — это целая серия шагов. Вы делаете первый шаг, снимаете метрики, что у вас используется в продукте и насколько это интересно конечному пользователю, затем корректируете свои дальнейшие шаги. Пусть вы не придёте сразу в точку Б, но окажетесь в её окрестностях, оглядитесь, поймёте, что движетесь в правильном направлении, и сделаете последующие шаги.

Почему это важно

При коммерческой разработке у нас постоянно меняются условия:

- выходят новые законы,
- изменяются бизнес-требования,
- конкуренты вдруг выпускают что-то похожее на наш продукт или даже круче.

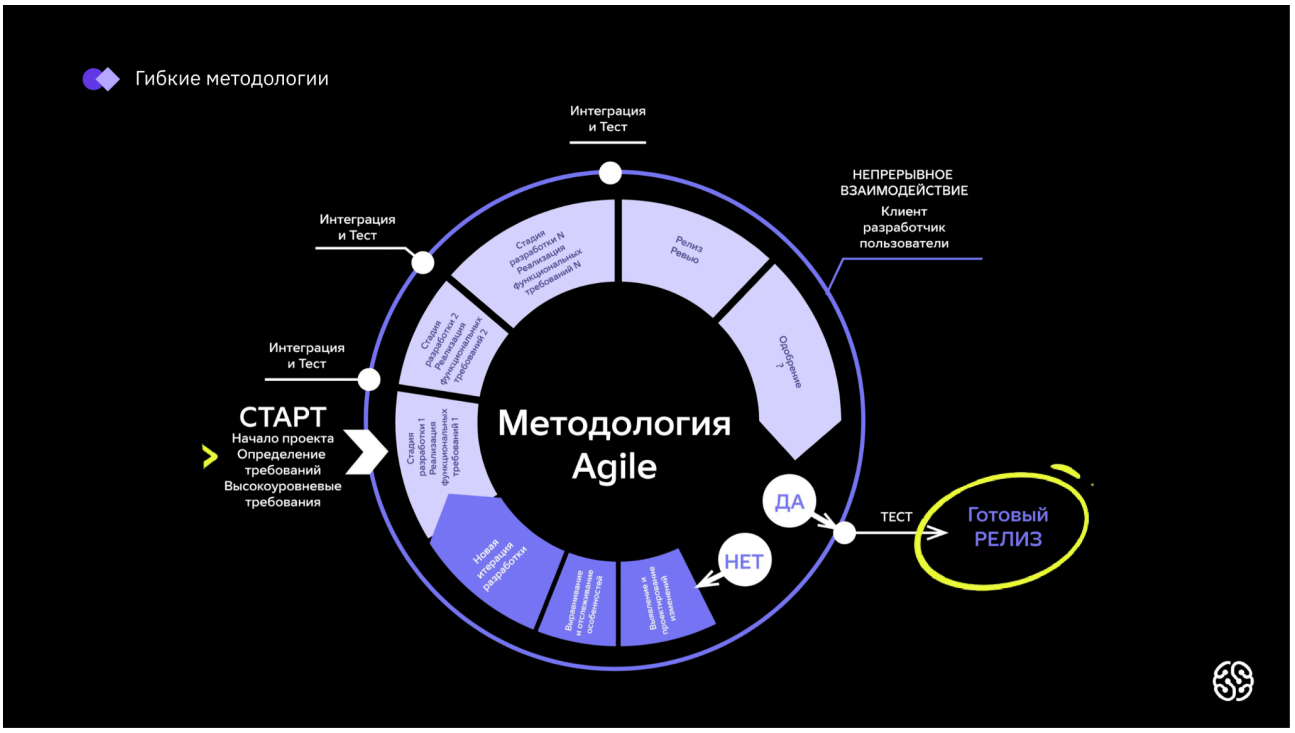
А, значит, мы должны соответствовать этим условиям, делать продукт лучше, чем конкуренты, учитывать новые законодательные акты и в целом отвечать на все вызовы среды, окружающей наш продукт.

В итоге может получиться, что мы из точки А должны дойти не в точку Б, а уже в точку В.

⚠️ **Внимание!**

Agile-методология подразумевает, что после выпуска первого релиза вы снимаете метрики, что-то переделываете, «выкатываете» следующий релиз, и т.д. И на каком-то этапе понимаете, что конкурент выпустил какую-то фичу и вам нужно идти не туда. В этот момент вы можете остановиться, принять другие требования и начать двигаться в точку В. При высокой изменчивости условий в процессе создания продукта вы всё время будете узнавать что-то новое. И в этом одно из преимуществ гибкой методологии.

Согласно Agile, чем больше ценностей мы поставили заказчику в рамках разработки, тем успешнее наш проект. А с точки зрения традиционной водопадной модели, проект считается успешным, только когда он уложился в рамки железного треугольника проекта: сроки, бюджет, содержание и качество. При этом сам продукт не всегда получается качественным и полностью удовлетворяет заказчика, несмотря на полное соответствие изначально прописанному ТЗ.



История Agile

Чтобы лучше понять логику Agile, погрузимся в историю его возникновения.

В феврале 2001 года 17 независимых практиков нескольких методик программирования, именующих себя Agile Alliance, собрались на горном курорте в штате Юта (США), отдохнули, пообщались и составили небольшой документ — Agile-манифест.

Этот документ стал новацией в разработке программного обеспечения и положил начало ряду практических подходов к программированию.

Манифест гибкой разработки программного обеспечения (Agile Manifesto) — основной документ, содержащий описание ценностей и принципов гибкой разработки ПО. Текст манифеста доступен на более чем 50 языках, в том числе на русском, и включает в себя 4 ценности, 12 принципов.

Ознакомиться с Agile-манифестом в оригинале Вы можете по ссылке.

Таким образом, наступила эпоха развития гибкого подхода к разработке IT-продукта.

Появление и предназначение Agile

Agile-методология сделала возможным решение ряда задач, которые стояли перед IT-сферой на момент её появления. Давайте рассмотрим эти задачи и поговорим, почему только Agile их решает.

⚠️ **Внимание!**

Ускорение вывода продукта на рынок. Если вы хотите что-то сделать быстрее, нужно делать это в соответствии с Agile.

Пример. Есть две компании, у них примерно одинаковый бизнес:

- Одна пишет ТЗ, затем проектирует систему и рисует дизайн — это Waterfall, на разработку по которой может уйти несколько месяцев.
- Во второй компании используют Agile, и за те же месяцы запускают сайт, выпускают ПО с минимальным набором функций, начинают зарабатывать деньги, захватывать рынок и привлекать новых пользователей, что самое главное в бизнесе.

При этом хотелось бы подчеркнуть: Agile не сократит сроки разработки полноценного ПО, но позволит гораздо раньше выпустить на рынок минимальную версию этого ПО, возможно, с очень ограниченным набором функций, зато с этими функциями уже будут знакомиться конечные пользователи, которые начнут проявлять к продукту интерес, ждать новых версий с добавлением новой функциональности, предусмотренной нами в ТЗ.

Если же мы используем водопадную модель, выпустить ПО на рынок мы сможем, только когда оно будет полностью готово. А это существенно повышает риск, что нас обгонят конкуренты и к моменту выхода ПО на рынок оно в чём-то уже будет неактуально, учитывая, насколько всё скоротечно в цифровом мире.

Управление изменениями в приоритетах. Это одна из самых болезненных проблем для всех компаний. Если вы делаете проект, который длится хотя бы несколько месяцев, у вас обязательно изменятся требования. Если говорить про коммерческую разработку, проблема в том, что программисты, аналитики, дизайнеры и даже руководитель проекта никогда не знают досконально, что нужно заказчику и конечным пользователям. Обычно все подходят к вопросу так: пока пользователь не опробует возможности сайта или ПО, мы не узнаем, нужна конкретная функция или нет, удалось ли нам сделать именно то, чего хотел заказчик и что найдёт отклик в сердцах конечных пользователей.

Улучшение взаимодействия между IT и бизнесом. Это большая проблема всех компаний, особенно крупных, ведь у бизнеса периодически меняются требования и каждый говорит на своем языке. IT не понимает бизнеса, а бизнес не понимает IT. Вот и получается, что стороны не понимают друг друга.

Ответом на все эти вызовы и стал тот самый Манифест гибкой разработки ПО.

Ценности Agile

Гибкая методология — не единый подход к разработке, а набор идей и принципов, на которых основаны конкретные практические решения. Можно назвать это особой философией, которая задаёт вектор, а не предписывает действия. Именно эти идеи и были сформулированы в Agile-манифесте.

4 центральных идеи Agile Manifesto:

- Люди и взаимодействие важнее, чем процессы и инструменты.
- Работающее ПО важнее, чем исчерпывающая документация.
- Сотрудничество с заказчиком важнее, чем согласование условий контракта.
- Готовность к изменениям важнее, чем следование первоначальному плану.

Очень часто люди, которые стремятся погрузиться в Agile и изучают манифест самостоятельно, допускают грубейшую ошибку: они читают 4 ценности и решают, что левая часть каждой ценности исключает правую. То есть, например, что мы делаем работающий продукт и полностью отказываемся от документации. Это в корне не верно! Всегда стоит помнить о том, что обе части важны, но, в случае конфликта между правой и левой частью, приоритет отдаётся левой.

Принципы Agile

Вышеперечисленные 4 основные ценности (идеи) повлекли за собой **12 принципов Agile**:

1. Наивысшая ценность — это удовлетворение потребностей заказчика благодаря регулярной и максимально ранней поставке ценного для него ПО.
Например, если заказчик хочет получить от нас большого слона и мы можем дать ему часть этого слона не через год, а через три месяца, потом ещё часть через три месяца, а затем ежемесячно выдавать кусочки, то чем чаще мы это будем делать и чем раньше, тем лучше.

2. Мы всегда готовы изменять требования, даже на поздних стадиях проекта, если узнаем что-то новое. Таким образом, мы создаем бизнесу или внешнему заказчику конкурентное преимущество.
Допустим, работают две компании: одна написала ТЗ и за год сделала продукт, а мы сделали концепцию продукта (не важно, в каком виде) и постепенно его выкатываем и раскатываем. Тогда наш продукт будет больше соответствовать требованиям заказчиков, пользователей и рынка в целом.

3. При использовании Agile работающий продукт выпускают максимально часто. В манифесте прописаны сроки от пары недель до пары месяцев. В России чаще всего каждые две недели выпускается что-то новое.

4. Бизнес обязательно должен работать вместе с программистами, помогать им понять специфику данного рынка.

Например, чтобы создать качественный финансово-технический IT-продукт для банка, программисты должны понимать принцип работы банка не только в целом, но в деталях. А всё знать невозможно! Поэтому нужны консультанты со стороны банка, т.е. со стороны самого заказчика.

⚠️ **Внимание!**

Наиболее частой проблемой является недоступность бизнеса, когда разработчик не может получить у сотрудника нужную информацию. При использовании Agile важно избегать возникновения подобных ситуаций.

5.Команда — один из краеугольных камней Agile. Наилучших результатов достигает команда замотивированных профессионалов. В Agile руководитель прежде всего должен создавать условия для команды и обеспечивать всестороннюю поддержку, проводить коучинг, следить за атмосферой в коллективе.

6. Есть много исследований, которые показывают, что лучшее общение — лицом к лицу. Причем желательно во время общения иметь под рукой, на чём писать: лист бумаги, доску со стикерами или другое средство визуализации.

⚠️ **Внимание!**

Из этого следует, что самый простой и эффективный способ узнать требования клиента, заказчика или пользователя — поговорить с ними.

7. Работающий продукт. Степень готовности проекта должна измеряться не словами о том, что ТЗ уже написано и 50% макетов нарисовано, а объёмом функциональности, выпущенной в production.

8. В Agile важен ритм: необходимы постоянные улучшения. Бизнес и программисты всегда должны иметь возможность делать процесс устойчивым, постоянно его улучшать.

9. Agile вообще не будет работать, если вы написали кривой код. У вас должна быть хорошая гибкая архитектура, в которую можно добавлять разные элементы и при необходимости легко их изменять. И если команда не будет уделять максимум внимания техническому качеству (писать хороший код, использовать инженерные практики, автоматизировать процессы), никакого Agile не получится.

10. Простота. Она проявляется в технической составляющей, в дизайне. Это один из принципов экстремального программирования. Простота очень важна также с точки зрения выпуска продукта: когда вы хотите «нарезать слона», лучше начать с простой части.

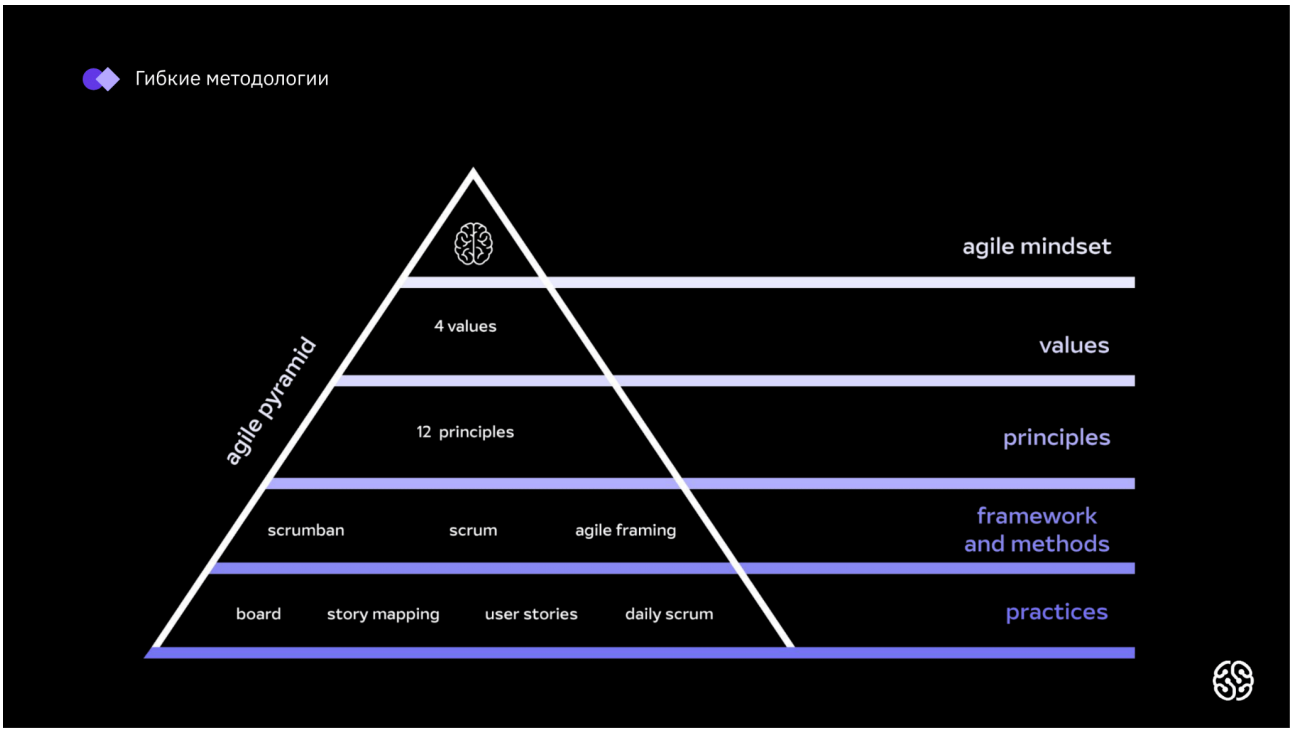
11. Менеджер в команде меняет свою роль: он не столько занимается организацией процесса, сколько учит команду. Поэтому команда должна быть самоорганизованной. Есть даже специальные стратегии, как из группы людей сделать самоорганизованную команду.

11. Команда должна постоянно анализировать свою работу, процессы: что получилось, как они этого добились, и постоянно улучшать организацию работ.

> ⚠️ **Внимание!**
>
> Руководствуясь только этими идеями и принципами, выстроить рабочие процессы нельзя. Поэтому принято считать, что *Agile – это целый класс, в рамках которого существует ряд прикладных методологий.*

Подытоживая, можно сказать, что Agile — это серия подходов к разработке IT-продуктов путём непрерывной и быстрой поставки ценных рабочих возможностей самоорганизованной командой профессионалов в сотрудничестве с заказчиком.

Давайте повторим: у нас получается пирамида, состоящая из четырех ценностей, на которых выстроены 12 принципов. Теперь посмотрим на конкретные практики.



Практики

Одна из ценностей Agile гласит:

> Мы должны выстраивать рабочий процесс, налаживая взаимодействие и коммуникации между людьми.

Это отражается в практических шагах.

1. **Стендапы.** Да-да, именно так это и называется.

Проходят они довольно просто: каждый день в определённое время команда собирается и синхронизирует свою деятельность.

Пример. Утром команда разработчиков проводит совещание и синхронизирует свою деятельность на предстоящий день. Но это не привычное для всех жёстко регламентированное совещание, а скорее ***стендап*** без шуток 🙂.

На таком стендапе каждый может выступить и рассказать самое важное и актуальное, что он сделал и что планирует сделать для продукта.

Почему это Agile? Потому что нет потока малозначимой документации в виде бумажных отчётов о проделанной за неделю работы.

Если в вашей команде не получается использовать стендапы, значит, у вас совсем не организованная команда.

1. **Планирование итераций.** Команда планирует объём работ на ближайшую итерацию. И, как оказалось, в Agile тоже присутствует планирование.

Например, если итерация идёт две недели, то формируется план, декомпозиция и даже разбивка бизнес-задачи на технические задачи.

1. **Unit-тестирование**. Это очень полезная Agile-практика и одна из самых простых инженерных практик, которую можно достаточно быстро начать использовать. Если есть проблемы с качеством кода или IT-продукта, почти 60—70% багов можно отловить unit-тестированием.

Unit-тестирование (его ещё называют «модульное тестирование») — проверка на корректность отдельных единиц работы исходного кода программы. Минимальными единицами работы для нас являются процедуры и функции.

1. **Планирование релизов.** Здесь уже происходит планирование большими кусками: что и когда будем выпускать в свет.

Agile-методологии

Agile — это не одна какая-то методология, а **гибкий подход** к разработке и управлению проектом, который включает в себя целое семейство гибких методологий, из которых наиболее распространены следующие:

- Scrum — самая большая и часто применимая методология).
- Экстремальное программирование (XP).
- DSDM (Dynamic Systems Development Method) — метод разработки динамических систем.

- Crystal.
- FDD (Feature Driven Development).

Более половины компаний, применяющих Agile-подход, используют Scrum.

На втором месте по частоте использования находится комбинация Scrum и XP, когда берутся управленческие практики из Scrum и добавляются инженерные. А вот комбинация Scrum и Kanban не пользуется такой популярностью, её использует примерно 8% компаний, занимающихся разработкой. Сейчас это один из трендов, мода.

Подробнее эти методологии мы рассмотрим на следующих уроках. А сейчас выделим преимущества, недостатки и границы применимости Agile на основании всего вышесказанного.

Преимущества Agile

- Программный продукт готов к использованию на самых ранних этапах его разработки, пусть и не с полной функциональностью.
- Разработчики постоянно в контакте с заказчиком и пользователем и всегда знают, что именно требуется от программы, могут оперативно реагировать на новые потребности пользователя и пожелания к продукту.
- Нет жёстких рамок, поэтому программный продукт постоянно изменяется и улучшается, становится эффективнее и привлекательнее для пользователя.
- Заказчик и пользователь в Agile — партнёры и идейные вдохновители. И это действительно помогает разрабатывать качественный продукт, потому что не всегда программист ясно представляет, что именно хочет получить пользователь. А пользователь, в свою очередь, далёк от разработки и не понимает какие возможности он может получить с помощью программы, какие процессы можно автоматизировать и на каком уровне. Объединив усилия и общаясь, пользователь и разработчик способны создать продукт, превосходящий аналоги по эффективности и эргономичности.
- Нет заранее и подробно сформулированного технического задания — значит, разработчик может решить задачу творчески. Но не слишком: пользователь не позволит ему оторваться от реальности и наплодить ненужного кода. Каждый фрагмент программы будут обсуждать и продумывать совместно.

Недостатки Agile

- Не факт, что программа когда-нибудь будет завершена. Ведь после каждой итерации и у разработчика, и у пользователя будут возникать новые идеи, как сделать продукт

ещё мощнее и полезнее. В таком случае появляется риск, что разработка может растянуться на годы. Но если вы планируете долговременное сотрудничество с заказчиком и он готов платить за всё время разработки, почему бы и нет?

- Пользователь требует всё и сразу. Большинство участников проекта со стороны пользователей могут на ранних этапах сформулировать множество требований к программе и ожидать, что всё это будет реализовано в ближайших итерациях. Значительная часть требований получит наивысший приоритет, и разработчику придётся определять, какие задачи выполнить сейчас, а какие отложить. А пользователи будут недовольны: они хотят всё и сразу.

- Работа над проектом требует не только профессионализма разработчика, но и сознательности пользователя. И сразу же возникает вопрос, часто ли встречаются адекватные, понимающие пользователи, готовые ждать исправления багов, разработки продукта и т.д. Очевидно, что нет. В таком случае, заказчик и пользователи могут быть недовольны подходом.

- «Золотые пользователи». Если в обсуждении участвуют несколько заказчиков (пользователей), их вклад в проект часто разномасштабен. Один более внимателен и вносит много предложений, а другой сидит молча. Обсуждение проекта с широким охватом может и вовсе проходить на форуме. Фактически это приведёт к тому, что небольшая группа активистов будет уводить разработку по интересному лично им пути, формировать программу под себя. Мнения других пользователей уйдут в тень.

- Строительство без чертежей. Ещё одна проблема, на которую обращают внимание критики гибких методик — отсутствие генерального плана, концепции программы, единой структуры. Код такого программного продукта может напоминать небоскрёб, который построили без чертежей и плана коммуникаций. Решения о нововведениях принимаются буквально на ходу, о долговременном планировании и речь не идёт. В результате оказывается, что уже реализованные участки кода не вписываются в архитектуру, которую подразумевает новая функциональность. Их приходится дорабатывать и добавлять «костыли», а то и полностью переделывать.

С каждой новой итерацией количество «подпорок» растёт катастрофическими темпами, делая внутреннюю структуру программы нелогичной и малоэффективной. А тестирование на каждом этапе проводится только для вновь созданной или доработанной функциональности. И нет гарантии, что, поправив код в одном месте, не сломаешь в другом.

- Постоянная спешка. Работа по Agile очень интенсивна и не располагает к медитации. Нововведения изобретаются на лету и их нужно так же быстро реализовывать. Приходится реагировать моментально и оперативно работать. Нет времени обдумывать все аспекты, неторопливо взвешивать за и против. И это, конечно, сказывается на качестве кода: бывает, фрагменты программы пишутся по принципу «и так сойдёт», без попыток сделать их более изящными или эффективными. Разработчик осознаёт, что такой код годится только как временное решение, но

вернуться и переписать получается редко. «Работает и ладно. До тех пор, пока работает... А там переделаем».

Но, несмотря на критику, подход Agile и гибкие методологии разработки успешно используется при создании IT-продуктов.

Границы применимости Agile

Гибким методологиям приписывают невероятные достижения и чудодейственную способность решать любые проблемы разработки. Но в действительности ничего фантастического в них нет. Это хорошие инструменты: если ими грамотно пользоваться, они могут сделать рабочие процессы быстрыми и эффективными, а ещё — мотивировать членов команды трудиться творчески и развиваться.

⚠️ **Внимание!**

Agile-методологии предъявляют высокие требования к профессионализму, квалификации и настрою специалистов. Важны сплоченность коллектива, взаимное уважение и обмен опытом.

Вы можете сказать, что уже работали по похожей схеме, хоть и не знали об Agile. Для разработчика естественно писать код небольшими фрагментами, время от времени собираясь с коллегами у кофейного автомата, чтобы обменяться мнениями и информацией. Разумно разбивать разработку на итерации, в ходе которых устранять баги и добавлять фичи, а в конце выкатывать новую версию. В этом прелесть методологий Agile: они интуитивно понятны и близки каждому программисту.

Гибкие Agile-методологии хорошо применять в следующих случаях:

- В вашей команде работают опытные и квалифицированные специалисты, умеющие действовать сообща и помогать друг другу.
- У заказчика нет ясного представления, как должен выглядеть готовый итоговый продукт, но он готов участвовать в совместной работе по развитию и улучшению продукта.
- Сфера применения продукта постоянно меняется, часто возникают новые потребности и задачи.
- Нет конкретных сроков, к которым продукт должен быть завершен, и жёстких ограничений бюджета.
- Работоспособную программу необходимо получить как можно скорее.

Яркий пример применения Agile — **стартапы**. Практически все вышеперечисленные условия подходят под этот пример. Стартап подразумевает инновации в конкретной области, и важно успеть занять нишу, выдав работающий продукт. При этом нет долгосрочных прогнозов, в каком направлении будет развиваться этот продукт.

Используемые источники

1. Project Management Body of Knowledge (PMBOK).
2. Эндрю Стеллман, Дженнифер Грин «Постигая Agile. Ценности, принципы, методологии».
3. Стивен Деннинг «Эпоха Agile».
4. https://blog.ganttpro.com/ru/metodologiya-agile-methodology/
5. https://rb.ru/opinion/agile-practices-russia/
6. https://infostart.ru/1c/articles/1056335/