Как правило, процесс разработки любого программного продукта средней и высокой сложности ведётся в составе команды (или нескольких команд) специалистов, которая включает в себя разработчиков, аналитиков, менеджеров, тестировщиков, системных администраторов, дизайнеров и представителей других направлений. Каждая из этих групп принимает участие на тех или иных стадиях разработки проекта, которые вместе составляют жизненный цикл программного продукта.

Fullstack-разработчик — универсальный солдат, который может самостоятельно реализовать проект «под ключ», охватив и backend, и frontend.

Важные компетенции в fullstack-разработке — это и языки программирования и фреймворки для вёрстки и веб-дизайна. Кроме того, нужно понимать, как устроен **жизненный цикл** любого ПО.

Говоря простыми словами, **жизненный цикл ПО** — это некоторый период времени, который начинается с момента принятия решения о необходимости разработки конкретного программного продукта и заканчивается в момент его полного изъятия из эксплуатации.

Несмотря на весьма простое определение, правильное построение жизненного цикла ПО — очень сложный процесс, который достаточно строго регламентирован различными общепринятыми документами, например, международным стандартом ISO/IEC 12207:2008 или его отечественным аналогом ГОСТ 34.601-90. Эти документы в различных терминах, но в целом весьма близко по своей сути устанавливают общую структуру процессов жизненного цикла программного продукта, определяют виды деятельности и задачи, которые выполняются в ходе различных этапов жизненного цикла, описывают возможные типы документации, которые должны быть использованы в ходе решения этих задач, ставят в соответствие каждой активности людей, обладающих той или иной специализацией.

Основываясь на этих стандартах, отдельные организации создают свои подробные руководства по тому, как правильно и эффективно строить процесс разработки ПО и производить управление такими проектами.

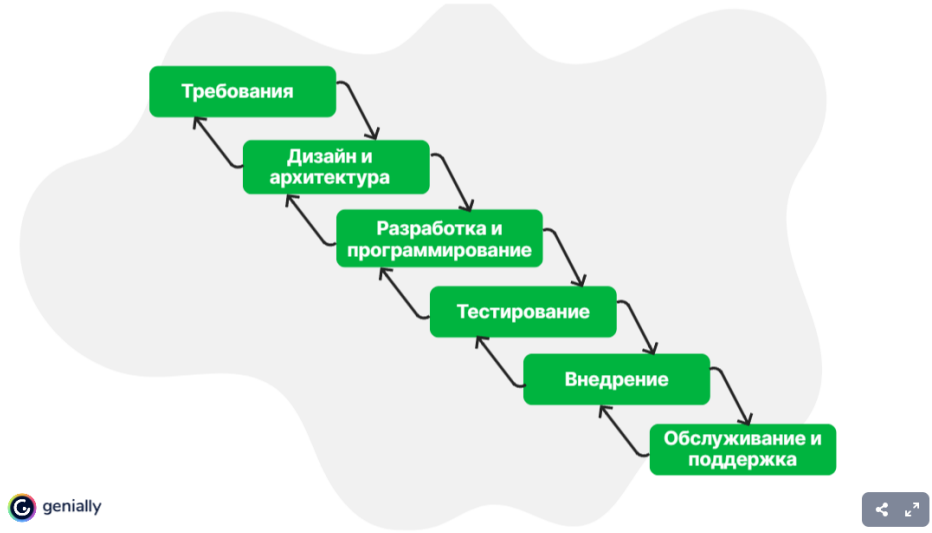
Наиболее известным из таких трудов является **PMBOK** (Project Management Body Of Knowledge, Свод знаний по управлению проектами) — достаточно

большой и регулярно обновляемый документ, который описывает суть процессов управления проектами, взаимодействия между ними и их конечные цели.

И стандарты, и PMBOK с подобными ему документами являются достаточно многословными и сложными для восприятия, кроме этого, обычно знание этой информации не нужно разработчикам ПО в полном объеме и рекомендуется лишь руководителям организаций и менеджерам проектов, поэтому в данном юните мы не будем сильно углубляться в текст этих документов и рассмотрим жизненный цикл ПО в упрощенном виде, узнаем, какие именно этапы он в себя включает и какие процессы выполняются на каждом из них.

Этапы жизненного цикла ПО

Основные этапы жизненного цикла ПО приведены на следующей диаграмме:



Далее подробнее рассмотрим, что включает в себя каждый из этих этапов и кто принимает в них участие.

Требования

На данном этапе производится определение детальных требований к желаемой для разработки системе. В процессе этого могут производиться дополнительные уточнения, выявляться и разрешаться возможные противоречия, устраняться различные пробелы в корректном понимании

тех или иных аспектов. Конечным результатом данного этапа являются точные, систематизированные и хорошо задокументированные требования.

Основными действующими лицами на данном этапе являются: бизнес-аналитики, дизайнеры, архитекторы, менеджеры и представители заказчика.

Дизайн и архитектура

В ходе этого этапа на основе полученных на предыдущем шаге требований архитекторы и разработчики разрабатывают высокоуровневый дизайн системы. Дополнительно может потребоваться участие системных администраторов и дизайнеров. Результатом работы команды на данном этапе является некоторое техническое описание архитектуры конечной системы и её отдельных элементов — дизайн-документ или дизайн-спецификация. Данный документ содержит как текстовое описание системы и её частей, так и различные визуализации: макеты, блок-схемы, UML- и ER-диаграммы.

Разработка и программирование

Данный этап представляет собой написание кода в соответствии с составленными ранее требованиями и спецификациями к системе. В ходе данного этапа производится настройка окружения для разработки, написание кода программы, отладка, сборка исходного кода программы в готовое приложение и базовое тестирование силами разработчиков как ручное, так и путем написания модульных (юнит-) тестов.

Обеспечение качества и программное тестирование

На данном этапе команда тестировщиков получает на вход разработанное приложение и производит более тщательную проверку и тестирование программного продукта на его соответствие установленным требованиям и наличие возможных ошибок в процессе его работы. Данный этап состоит из нескольких шагов – составление тестовых сценариев, сбор необходимой информации, подготовка и развертывание тестового окружения, выполнение непосредственно тестовых сценариев, составление отчетов об успешном тестировании и об ошибках, выполнение повторного тестирования в случае внесения исправлений. При этом сам процесс тестирования может быть как ручным, так и автоматизированным, в

зависимости от возможностей и ресурсов команды, также эти два подхода могут быть скомбинированы.

Внедрение

На данном этапе выполняется финальная сборка конечного решения, готового к использованию и дистрибуция его заказчику. При необходимости, например, в случае разработки SaaS-платформ, производится развертывание рабочего (production) окружения, которое будет доступно конечному пользователю. Также данный этап включает в себя процесс составления различной документации как технической, которая подробно описывает принципы работы программного продукта и его отдельных частей, так и пользовательской, которая включает в себя различные справочные и поясняющие текстовые, графические и видеоматериалы, необходимые для обучения конечного пользователя работе с разработанной системой.

Ключевыми лицами на данном этапе являются: системные администраторы, разработчики, технические писатели.

Обслуживание и поддержка

Данный этап выполняется непрерывно после выпуска финальной версии программного продукта и включает в себя такие шаги, как:

- сбор отзывов конечных пользователей о работе программы;
- сбор и анализ отчетов об ошибках, внесение исправлений в исходный код программы;
- дистрибуция обновленной версии приложения пользователям;
- обновление документации;
- при необходимости — вывод программного продукта из эксплуатации.

В данном процессе принимают участие все члены команды, поскольку он включает в себя активности из всех предыдущих этапов, также сюда добавляются и члены команды пользовательской поддержки.

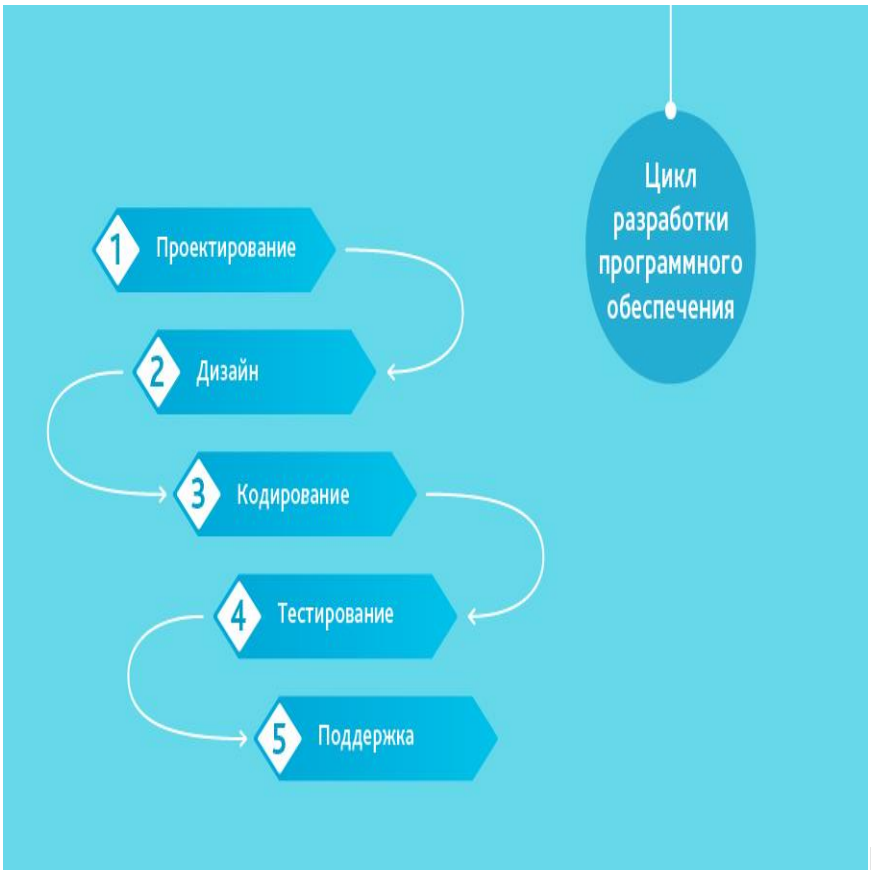
Модели жизненного цикла ПО

Перечисленные выше этапы жизненного цикла программного продукта необязательно следуют в строгом порядке друг за другом. В ходе развития информационных технологий и подходов к разработке программного

обеспечения были созданы различные модели жизненного цикла, которые задают определенный порядок выполнения этапов и позволяют решить те или иные проблемы.

Далее рассмотрим несколько наиболее известных моделей жизненного цикла ПО, разберем их преимущества и недостатки и узнаем, в проектах какого типа они могут эффективно применяться.

Каскадная или водопадная (Waterfall) модель



Источник: habrastorage.org

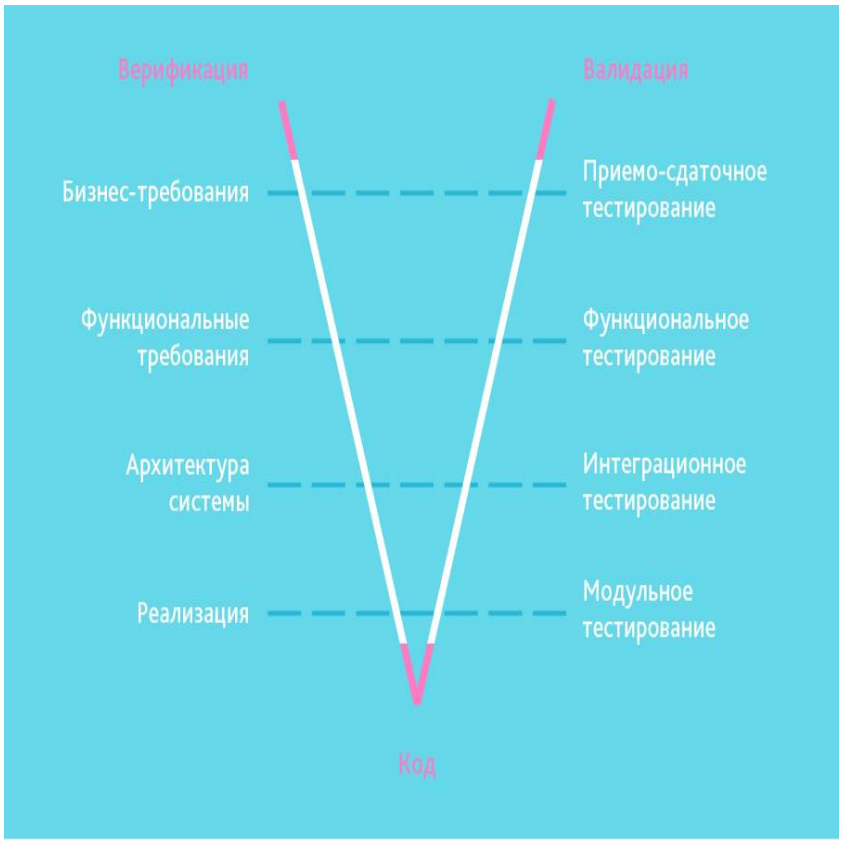
При использовании данной модели все этапы жизненного цикла выполняются друг за другом в строгом порядке, каждая следующая стадия начинается только после окончания предыдущей. Эта модель является одной из наиболее старых моделей процесса разработки ПО и применяется с 1970-х годов.

Ключевым преимуществом данного подхода является прогнозируемая продолжительность и стоимость проекта, которая известна заранее, а также точные требования, которые не меняются в ходе разработки, что позволяет разработчикам сконцентрироваться на реализации желаемой функциональности.

Основным же недостатком является высокая сложность формулировки четких требований при начале работы над проектом. Кроме этого, строгая последовательность выполнения этапов работы означает, что тестирование будет проведено только после полной разработки приложения, а заказчик увидит готовый продукт только в самом конце разработки без возможности дать обратную связь в процессе работы. Таким образом, при возникновении необходимости вернуться к одному из предыдущих этапов, сроки и стоимость работ резко возрастают.

Исходя из вышеперечисленного, можно сказать, что каскадная модель хорошо подходит для разработки проектов в хорошо известных предметных областях, которые уже содержат большое количество различной документации (стандарты, патенты, спецификации), на основе которых могут быть написаны максимально точные и подробные требования к продукту.

V-образная модель



Источник: habrastorage.org

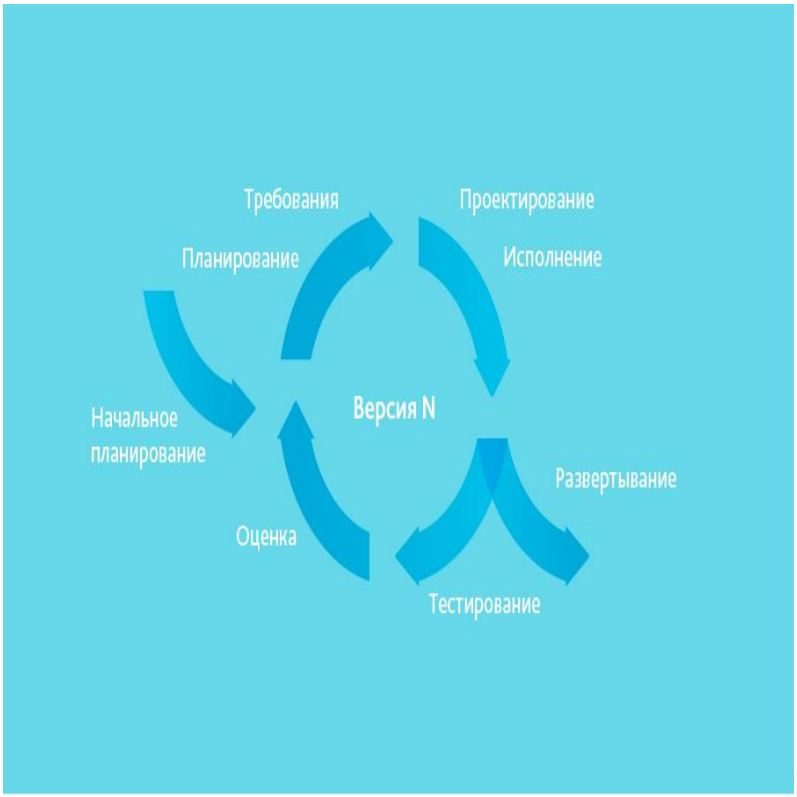
Данная модель является развитием идей каскадной модели, ключевым усовершенствованием является то, что параллельно с каждой фазой разработки выполняется соответствующая ей стадия тестирования.

Например, модульные тесты пишутся во время написания кода самого продукта, интеграционные разрабатываются при построении архитектуры и т.д.

Таким образом, основным преимуществом данной модели является то, что она сфокусирована на тщательной проверке качества исполнения каждого этапа разработки. Что касается недостатков, то она наследует все слабые стороны каскадной модели, также при использовании данной модели может потребоваться большее количество единовременно доступных человеческих ресурсов.

Данная модель отлично подходит для разработки проектов с очень строгими требованиями к качеству финального продукта, некорректная работа которого может привести к значительным финансовым потерям или нанести ущерб здоровью или жизни людей, например, в сфере разработки медицинского оборудования, а также ПО для автомобилей, самолетов и космических летательных аппаратов.

Инкрементная модель



Источник: habrastorage.org

Данная модель представляет собой каскадную модель, выполнение этапов которой повторяется в течение нескольких циклов, в ходе которых ведется

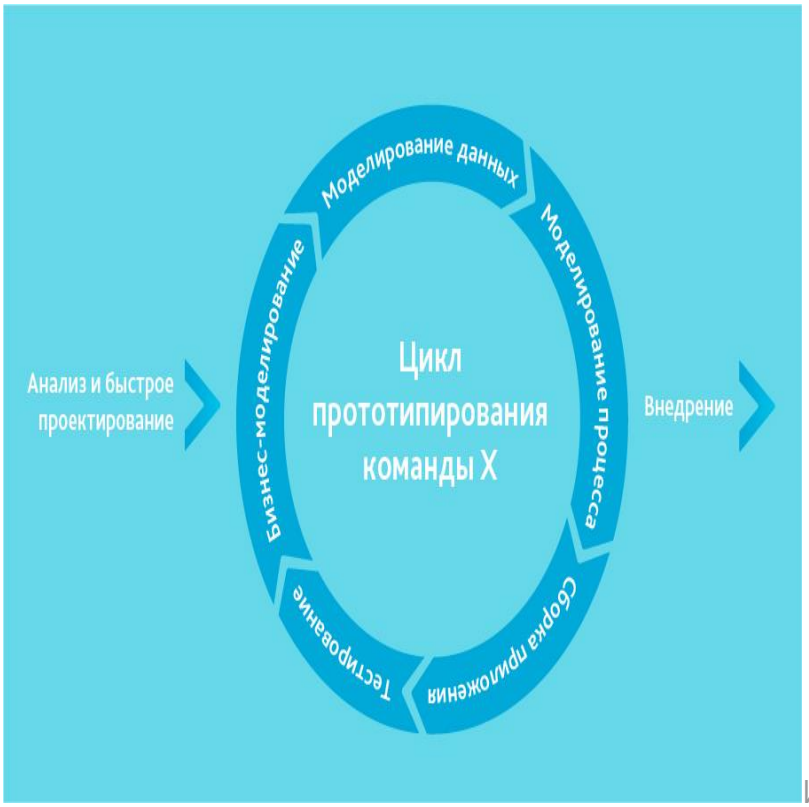
разработка заранее определенных модулей приложения. На выходе каждого из циклов заказчику представляется очередная версия разработанного программного продукта.

В качестве первой версии модель предполагает выпуск продукта с базовой функциональностью, которая может быть использована (MVP — Minimum Viable Product). Затем собирается и анализируется обратная связь, уточняются возможные дополнения в требованиях и в следующих циклах добавляется новая функциональность.

Ключевым преимуществом данной модели является возможность предоставить конкретные версии программного продукта заказчикам и пользователю в заранее определенные сроки, а также возможность внести в требования некоторые детали, которые не были учтены при изначальном планировании. Недостатки данной модели наследуются у каскадной модели, но актуальны только в рамках одного цикла.

Данная модель может быть актуальна при разработке приложения, основные требования к которому уже четко определены и понятны, однако незначительные детали могут меняться в течение времени. Кроме этого, разработка программного продукта по данной модели позволяет обеспечить вывод рабочего решения на рынок в относительно короткие сроки.

RAD (Rapid Application Development — быстрая разработка приложений)



Источник: habrastorage.org

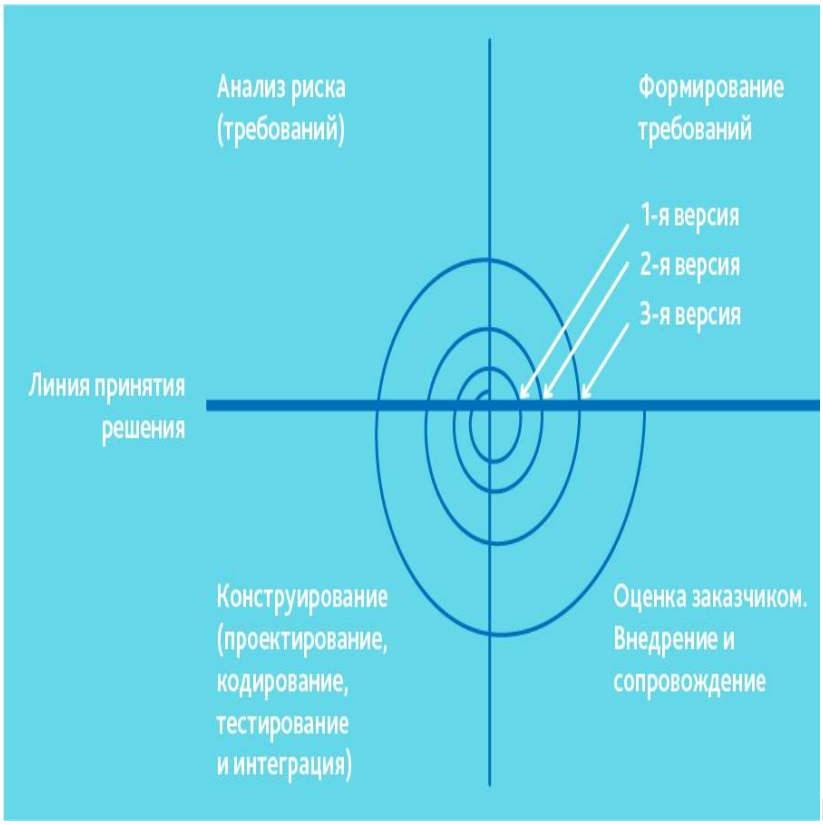
В рамках данной модели разрабатываемый программный продукт разбивается на несколько независимых или слабо связанных модулей, каждый из которых разрабатывается параллельно отдельной командой в сжатые сроки. Затем эти модули объединяются в готовое решение, которое может быть представлено заказчику.

Основным преимуществом этой модели является возможность вывести продукт на рынок в кратчайшие сроки, а также постоянная вовлеченность заказчика и всех команд в процесс разработки, что позволяет быстро получать обратную связь и вносить необходимые изменения.
Главным недостатком данной модели является предположительно высокая стоимость проекта, поскольку выпуск продукта в кратчайшие сроки за счет параллельной работы множества команд подразумевает одновременную работу большого количества высококвалифицированных и высокооплачиваемых специалистов.

Данная модель может быть использована при разработке программного продукта, который может потерять свою актуальность или потенциальную аудиторию через определенный промежуток времени (например,

демонстрационное приложение для стенда на крупной конференции), либо продукта, от которого могут зависеть здоровье и жизнь людей (например, приложение для отслеживания состояния больных в условиях пандемии).

Спиральная модель



Источник: habrastorage.org

Эта модель заимствует идею инкрементной модели, однако фокусируется прежде всего на начальных этапах: составлении требований и проектировании, уделяя большое внимание анализу возможных рисков и принятии решения о необходимости выполнения следующей итерации.

Важным преимуществом данной модели является возможность свести к минимуму возможные сложности и проблемы в процессе разработки и поддержки следующей версии программного продукта.
Однако, данный процесс анализа рисков является наиболее затратным в плане временных и финансовых ресурсов, что может оказаться и минусом данной модели.

Данная модель хорошо подойдет при разработке крупных программных систем, для которых может требоваться регулярное обновление, которое, в свою очередь, может вызвать различные сложности в дальнейшей

поддержке, что может привести к значительным финансовым потерям, например, электронные системы документооборота.

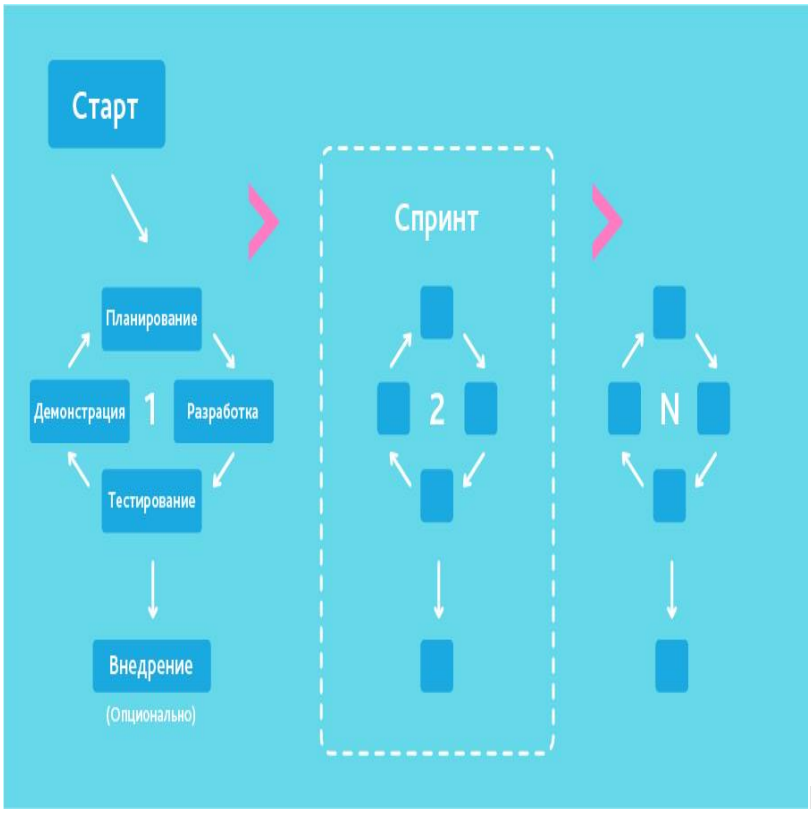
Agile (Гибкая модель разработки)

Гибкая модель разработки основана на так называемом Agile-манифесте, который при разработке программного обеспечения ставит во главу угла необходимость частого взаимодействия между членами команды и с заказчиком, готовность уточнять требования и вносить изменения по ходу разработки и важность регулярно предоставлять работающий продукт.

Agile-модель строится на одной или нескольких методологиях или практиках, которые могут быть использованы при организации процесса управления проектом разработки программного продукта, среди них: **Scrum** («схватка»), **XP** (eXtreme Programming — экстремальное программирование), **FDD** (Feature Driven Development — функционально-ориентированная разработка), **Lean Development** (бережливая разработка), **Kanban** и т.д.

На сегодняшний день Agile-модель и методология Scrum являются одними из наиболее популярных методов для управления проектами в компаниях, занимающихся продуктовой и заказной разработкой ПО, в частности, современных веб-приложений.

Рассмотрим подробнее, что представляет из себя данная методология.



Источник: habrastorage.org

Scrum предполагает деление процесса разработки на короткие промежутки времени — спринты, длящиеся от 1 до 4 недель (в среднем 2 недели), в которые предполагается выполнение определенного набора задач.

Первоначально все задачи находятся в **бэклоге (backlog)** — своего рода журнале, куда заносятся все созданные задачи, которые необходимо будет реализовать в процессе разработки. Активные задачи переносятся из бэклога на **доску спринта** (Sprint Board) и в процессе работы над ними перетаскиваются из одного состояния в другое, например: **Open** → **In Progress** → **On Review** → **Done.** Данные состояния необязательные, их число и названия могут варьироваться в зависимости от предпочтений команды.

Scrum четко разделяет членов команды на отдельные роли, среди таких ролей:

1. **Scrum Master** — человек, который руководит корректным процессом Scrum внутри команды, назначает совещания и проводит их, разрешает противоречия.

2. **Product Owner (PO)** — человек, разбирающийся в предметной области и представляющий интересы конечных пользователей и заказчиков.
3. **Developers** — команда разработчиков проекта, которая включает в себя представителей разных специальностей: аналитики, программисты, тестировщики, дизайнеры и т. д. Предполагается, что команда разработчиков состоит не более, чем из 9-10 человек, если их больше, такую команду могут разделить на несколько меньших.
4. **Stakeholders** (стейкхолдеры) — заказчики или иные лица, финансово или другим образом заинтересованные в проекте.
5. **Users** — конечные пользователи.

Стейкхолдеры и пользователи не являются непосредственными участниками Scrum-команды и не принимают прямого участия в Scrum-процессах.

Важной особенностью Scrum также является большое количество совещаний, на каждом из которых команда обсуждает актуальные и насущные вопросы:

6. **Daily Meeting** (дейли) — ежедневное совещание, на которой члены команды рассказывают о своем прогрессе, делятся своими проблемами и могут обсудить пути их решения или договориться о назначении отдельного совещания.
7. **Sprint Planning** (планнинг) — совещание, проводящееся перед началом следующего спринта, на котором обсуждается, какие задачи будут взяты, кто будет за них ответственен. Также задачи оцениваются в относительных величинах по своей сложности, чтобы команда могла взять на себя адекватную нагрузку в течение спринта.
8. **Sprint Retrospective** (ретроспектива) — совещание, которое проводится после окончания спринта, на которой команда делится своими впечатлениями о прошедшем спринте, отмечая на специальной доске следующие пункты: что удалось выполнить, где возникли проблемы и что можно улучшить. Затем выделяются некоторые действия (action points) для реализации этих улучшений и назначаются ответственные за их выполнение люди.

9. **Demonstration** (демо) — совещание, на котором результат работы команды в течение спринта в виде непосредственно работающего продукта представляется PO, обсуждаются возможные вопросы и, при необходимости, дополнительные исправления и улучшения.
10. **Backlog Grooming** (груминг) — необязательное совещание, на котором может проводится анализ существующих задач в бэклоге, какие-то из них получают приоритет для попадания в спринт, в каких-то может выставляться предварительная оценка или уточняться некоторые требования, какие-то могут удаляться как более неактуальные.
11. **Scrum of Scrums** — необязательное совещание, которое может проводиться в случае работы нескольких Scrum-команд над некоторым проектом. На данном совещании представители разных команд делятся информацией о текущем состоянии и прогрессе.

Для оценки эффективности работы команды обычно используется **диаграмма сгорания задач** (BurndownChart), которая показывает, какое количество задач было закрыто в спринте и с какой скоростью. Если часть задач остается незакрытыми, это позволяет провести дополнительный анализ и поменять подход к оцениванию задач или брать меньше задач в следующий спринт.

Исходя из перечисленных выше особенностей, можно заметить, что при работе по Scrum команда разработчиков действительно является очень гибкой — постоянно находится в контакте с представителями заказчиков, может быстро реагировать на возможные изменения требований и в короткие сроки предоставлять корректно работающую функциональность. Такой процесс весьма удобен при разработке большого количества современного ПО, в частности, многих веб-приложений, поскольку оно не требует строгих соответствий тем или иным регламентам, а требования могут регулярно меняться, реагируя на различные события в индустрии и мире.

1/1 point (ungraded)

Укажите порядок выполнения этапов ЖЦ ПО в каскадной модели (в качестве ответа введите соответствующие буквы без пробелов и запятых):

a. Проектирование

b. Внедрение

c. Поддержка

d. Сбор требований

e. Тестирование

f. Разработка

dafebc ✔

Show answer

Отправить

1/1 point (ungraded)

Какая модель ЖЦ ПО рекомендуется при разработке ПО со строгими регламентными требованиями к его функциональности?

- (●) V-образная
- ( ) Инкрементная
- ( ) RAD
- ( ) Agile

✔

Show answer

Отправить

Какие модели ЖЦ ПО могут быть использованы при разработке ПО для быстрого запуска на рынок?
Несколько верных вариантов ответа

- [ ] 1. Каскадная
- [x] 2. RAD
- [ ] 3. Спиральная
- [x] 4. Agile

✔

Show answer

Отправить

1/1 point (ungraded)

Какая аббревиатура используется для определения ранней версии приложения, готовой к выходу на рынок?

- ○ XP
- ○ FDD
- ◉ MVP
- ○ PO

✔

Show answer

Отправить

Какой может быть длина спринта по методологии Scrum?

Несколько верных вариантов ответа

- ☑ 1 неделя
- ☑ 2 недели
- ☑ 4 недели
- ☐ 6 недель

✔

Show answer

Кто отвечает за представление интересов заказчика в Scrum-команде?

- ○ Scrum Master
- ○ Stakeholder
- ◉ Product Owner
- ○ Project Manager

✔

Show answer

Какой вид совещаний не описан в Scrum?

- ○ Sprint Planning
- ◉ Teambuilding Lunch
- ○ Backlog Grooming
- ○ Daily Meeting

✔

Show answer

Отправить

Если фронтенд-разработчик отвечает за внешний вид и пользовательский интерфейс, то чем в команде занимается веб-дизайнер?

Современный веб-дизайн тесно связан с фронтенд-разработкой (frontend development). Коммуникация и синхронизация дизайнера и фронтенд-разработчика очень важна, чтобы готовый проект максимально точно воплотился из идеи в продукт для пользователя.

В фокусе внимания веб-дизайнера находится именно визуальная часть интерфейса приложения. И главная задача дизайнера — визуально воплотить идею проекта, чтобы пользовательская история была максимально удобна и понятна.

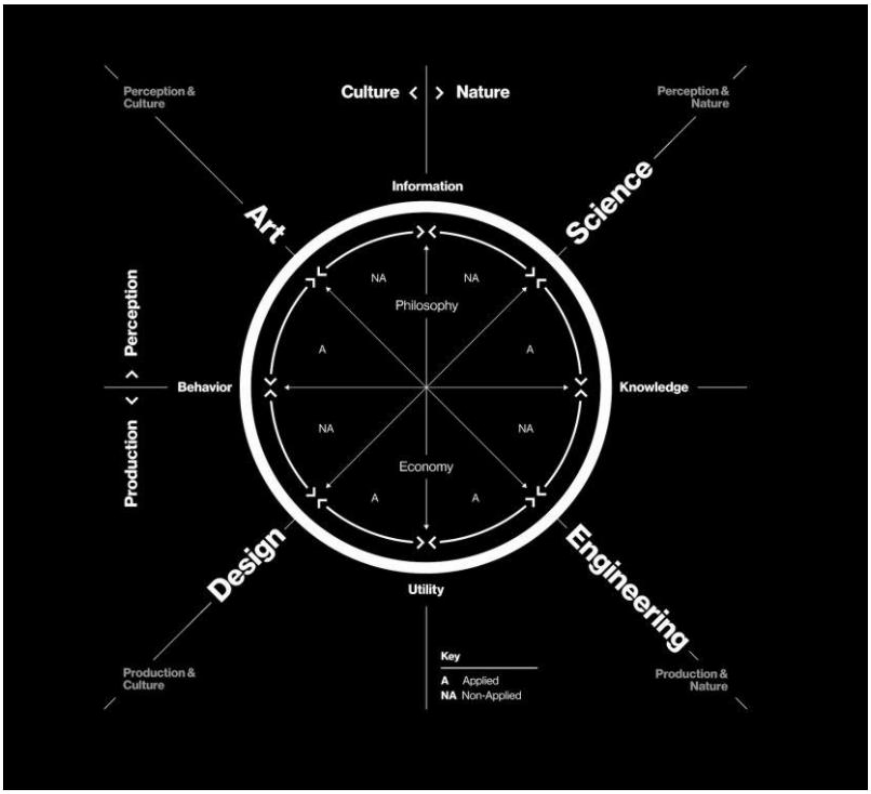
Таким образом, веб-дизайнер разрабатывает макет сайта или приложения и его интерфейс, чтобы разработчик собрал по макету работающий проект.

Интерфейс (interface) — это совокупность средств, методов и правил, предназначенных для взаимодействия элементов системы (или целых систем) между собой.

Веб-дизайнеры разрабатывают один из видов интерфейсов — **пользовательский** (взаимодействие пользователя (человека) с программно-аппаратными компонентами компьютерной системы). То есть находят баланс между визуальным отображением сайта и удобством его использования человеком.

Дизайн — это комбинация эстетических свойств, логики и возможностей.

В сериале Netflix «Абстракция: Искусство дизайна», дизайнер Нери Оксман представляет интересный взгляд на дизайн с помощью диаграммы процесса проектирования своей команды:



Дизайнеры по интерфейсам проектируют внешний вид сайтов, Интернет-сервисов, мобильных приложений, компьютерных программ. В нашем примере с сервисом Яндекс.Такси весь пользовательский интерфейс также спроектирован дизайнерами. Почитать, как создавали дизайн сервиса в связи с появлением беспилотников, можно здесь.

UX-эксперимент: создали впятером интерфейс беспилотного такси, чтобы переосмыслить командный дизайн — Дизайн на vc.ru

Для проектирования дизайнеру необходимо быть и аналитиком, который создаёт простые и понятные продукты на основе анализа поведения пользователей.

Веб-дизайнер в своем профессиональном развитии может углубиться в одно из конкретных направлений (моушн-дизайн, брендинг и создание логотипов и т.д.) или расширить свои компетенции, чтобы работать над всем продуктом более глобально.

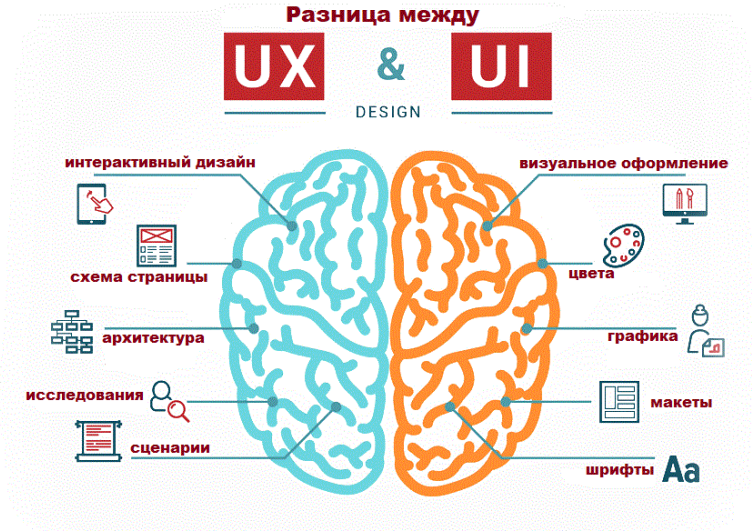
Более масштабное направление развития веб-дизайна — UX/UI-дизайн.

UX — это User Experience (дословно: «опыт пользователя»)

Какой опыт/впечатление получает пользователь от работы с интерфейсом? В этом направлении внимание уделяется пользователю и тому, какое впечатление он получает от работы с интерфейсом, как переходит по страницам, достигает ли он своей цели и насколько ему сложно это сделать.

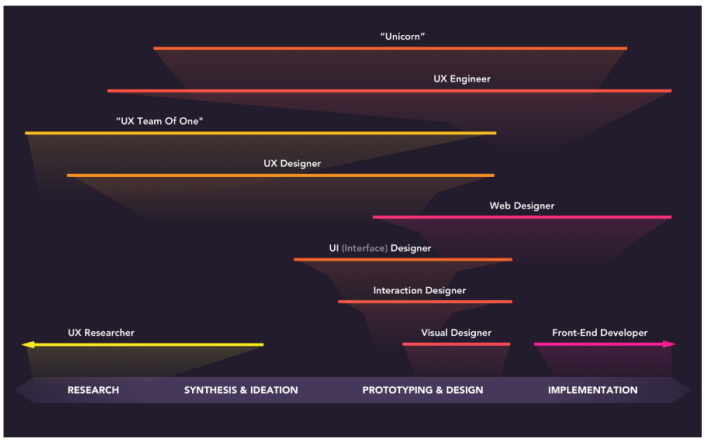
UI — это User Interface (дословно «пользовательский интерфейс»)

Как внешне выглядит интерфейс? Здесь важны визуальные и физические свойства. Проектируются цветовые акценты, размер, расположение элементов, так как от этого зависит удобство пользователя.



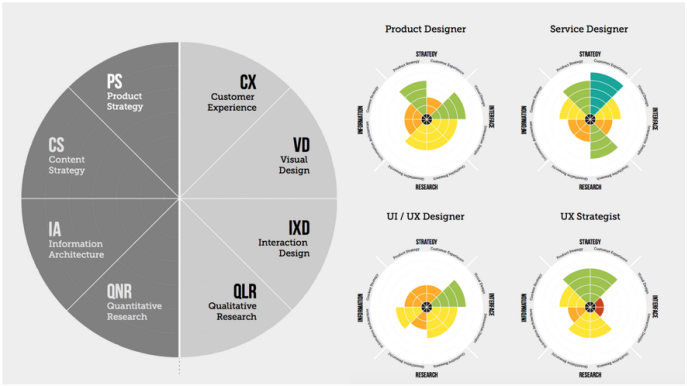
Грань между UX и UI постепенно стирается, поэтому специалисты всё чаще соединяют оба направления и пишут в резюме «UX/UI», и это уже более широкий профиль, чем веб-дизайн.

Как появилось понятие UX — видео от школы Contented



Спектр дисциплин и ролей дизайна. Источник: The spectrum of design roles in 2018 | UX Collective

Дизайнеры участвуют в разработке проекта на всех этапах. Посмотрите на карту , чтобы соотнести этап разработки с ролью дизайнеров в нём.



Карта специальности дизайнеров Джаспера Стивенсона. Источник: Vitamintalent

Задание для UX-дизайнера

Попробуйте самостоятельно оценить интерфейс сервиса Яндекс.Такси и, например, Wildberries / Ozon (сайт) с точки зрения **UX-дизайнера.**

Попробуйте ответить на вопросы:

- Какие бизнес-задачи решает продукт?
- Какие потребности пользователей удовлетворяет?
- Какие могли быть технические ограничения при создании продукта?

Полезные материалы от школы Contented

1. Книги для знакомства с веб-дизайном

- Стив Круг, «Не заставляйте меня думать»
- Дональд Норман «Дизайн привычных вещей»
- Алан Купер «Об интерфейсе»
- Уильям Лидвелл, Критина Холден, Джилл Батлер, «Универсальные принципы дизайна»
- Джесс Гарретт, «Веб-дизайн. Элементы опыта взаимодействия»

2. Телеграм-каналы для развития насмотренности и связи с веб-дизайном

- @pdigest — канал дизайнера интерфейсов Юрия Ветрова, урок которого вы только что посмотрели. Дайджесты с подборками самых свежих и полезных статей, инструментов, паттернов, кейсов.
- @duiux — авторские заметки, статьи, новости, кейсы, портфолио. Всё о дизайне интерфейсов. Для UX/UI дизайнеров и всех, кто в теме.
- @dangry — статьи и заметки о работе с интерфейсами.
- @thedesigner — переводы статей, полезные ссылки на статьи и сервисы, скрипты, портфолио, профессиональные советы, новости
- @uifail — наглядная подборка неудачных интерфейсов с объяснением проблем и предложениями по улучшению.
- @nowhow — примеры сайтов с отличным дизайном со всего мира, советы по продуктивности и образовательные материалы.
- @uxnotes — заметки о проектировании и UX-дизайне. Полезные ссылки на видео, статьи и литературу для юзабилити-специалистов.
- @mosinkru — канал о дизайне, интерфейсах, пользовательском опыте: интервью с дизайнерами, обзоры новых инструментов и сервисов, лекции и статьи.
- @uxidesign — переводы интересных статей зарубежных специалистов, UX-анализ топовых компаний, последние новости из мира пользовательских интерфейсов и ссылки на полезные ресурсы.

Задание для UX-дизайнера

Есть разные подходы к оценке продукта с точки зрения дизайна интерфейса и понимании того, какие факторы стоит учесть при его создании. Предлагаем оценить сервис Яндекс.Такси по пяти основным критериям:

12. **Обучаемость.** Насколько легко пользователь решает задачу в первый раз?
13. **Эффективность.** Как быстро пользователь выполняет типовые задачи после обучения?

14. **Запоминаемость.** Пользователю легко взаимодействовать с интерфейсом после перерыва?
15. **Предотвращение ошибок.** Интерфейс способен предотвратить ошибки пользователя? Есть возможность их быстро исправить?
16. **Удовлетворенность.** Насколько позитивно ощущение от использования системы?

Задание на mind map

Изучите доступные дополнительные источники и добавьте на mind map **сферу веб-дизайна.**

Отобразите на карте следующую информацию:

- задачи, которые решает веб-дизайн;
- части, из которых состоит веб-дизайн;
- специалисты, которые могут работать в сфере дизайна;
- 5 инструментов веб-дизайнера;
- отобразите на карте, как связан дизайн с остальными сферами;
- зафиксируйте способы взаимодействия фронтенд-разработчика с дизайнером.

Мы разобрались, как происходит общение клиентской и серверной части приложения на программном уровне, но как этот процесс организован на уровне передачи данных?

В этом юните мы поговорим о **сетевой инфраструктуре** и попробуем понять, какой путь проходит информация от одного устройства к другому, а также перечислим технологии, которые используются в этой сфере.

Что происходит после того, как вы сделали заказ в приложении Яндекс.Такси? Данные о вашем заказе всего за несколько мгновений проделывают огромный путь от вашего смартфона до Wi-Fi роутера, затем от роутера к вашему интернет-провайдеру, от него — с помощью сервера DNS определяется IP-адрес назначения, и данные идут по цепочке маршрутизаторов к интернет-провайдеру владельца сервиса, и уже оттуда попадают на сервер приложения Яндекс.Такси, где маршрутизатор принимает запрос и отправляет его на обработку конкретному программному сервису.

Схематически этот процесс можно представить так:

Источник: oyla.xyz

3/3 points (ungraded)

Определите IP-адреса следующих сайтов. Для этого можно воспользоваться одним из множества онлайн-сервисов в Интернете.

habr.com

178.248.237.68 ✔

skillfactory.ru

185.129.100.113 ✔

forbes.ru

178.248.232.187 ✔

Show answer

Чем же занимаются специалисты по сетевой инфраструктуре? Как правило, их усилия сосредоточены на обеспечении работы локальной сети офиса или предприятия.

Специалисты по сетевой инфраструктуре решают следующие задачи:

17. Поддержка физической инфраструктуры сети: закупка и установка оборудования и комплектующих, обслуживание кабельной системы.
18. Обеспечение бесперебойной работы программного обеспечения: правильная настройка серверов, организация резервного копирования, равномерное распределение нагрузки на сервера и т.д.
19. Администрирование рабочих станций (компьютеров, за которыми работают пользователи системы): установка ОС и

различных программ, борьба с вирусами, поддержание «железа» компьютера в рабочем состоянии.
20. Проектирование архитектуры локальной сети в соответствии с потребностями организации.
21. Контроль за безопасностью в сети.
22. Поддержка работоспособности сети в случае нештатных ситуаций (поломки, сбои).

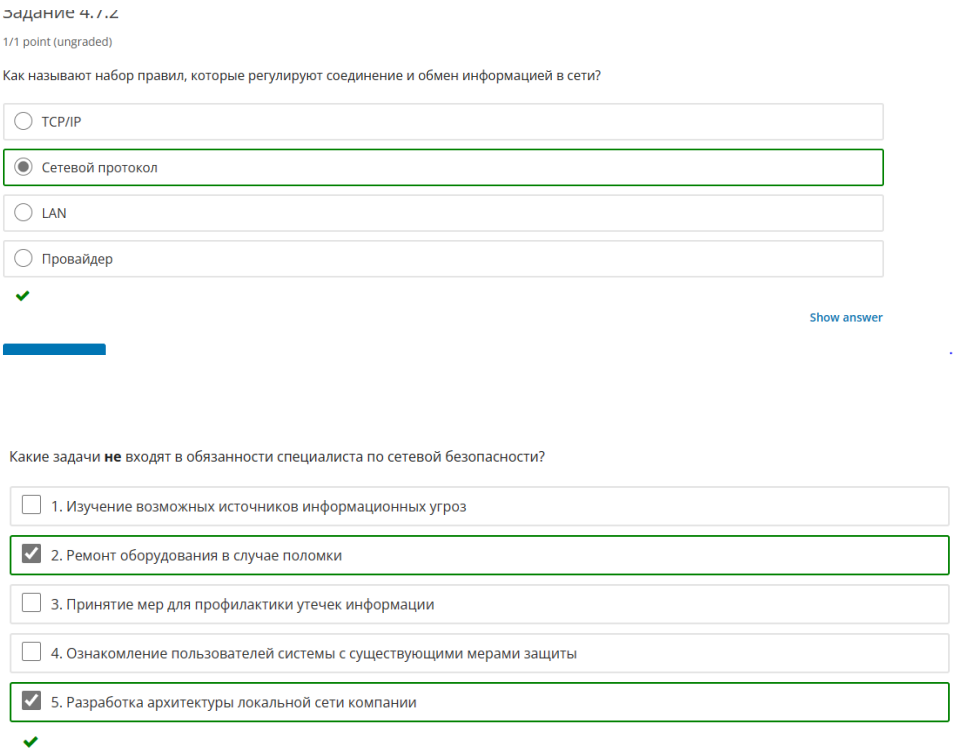
В сфере компьютерных сетей может быть задействовано несколько специалистов:

- **Сетевой техник**
  Обычно отвечает за «физическую» часть сети — оборудование, кабели, комплектующие.
- **Системный администратор**
  В основном занимается проектированием и обеспечением сетевого ПО, регулирует нагрузку на сеть и поддерживает её в работоспособном состоянии. В небольших фирмах может выполнять также обязанности техника и специалиста по безопасности.
- **Специалист по сетевой безопасности**
  Изучает и анализирует риски безопасности, проводит профилактические мероприятия, принимает меры в случае взломов или утечек информации.

Можно выделить ещё двух специалистов, которые хотя и не относятся напрямую к компьютерным сетям, но всё же часто имеют с ними дело:

- **Software engineer**
  Проектирует и разрабатывает крупное программное обеспечение — от интерфейса и взаимодействия с пользователем до взаимодействия с другим ПО в сети. Как правило, не занимается непосредственно программированием или занимается в меньшей степени, больше сосредоточен на построении архитектуры.
- **Hardware engineer**
  Проектирует и разрабатывает цифровые устройства. В отличие от software-инженера, больше работает с «железом» — процессоры, микросхемы и т.д. Так как большое количество устройств сейчас предполагают выход в сеть и взаимодействие

с другими устройствами, hardware-инженеру тоже необходимы знания в сфере компьютерных сетей.



Задание на mind map

Давайте добавим на mind map сетевую инфраструктуру.

Отобразите на карте следующую информацию:

- какие два вида сетей существуют;
- какой общепринятый регламент регулирует передачу данных в сети;
- какие задачи решаются специалистами по сетевой инфраструктуре;
- какие специалисты могут быть задействованы в этой сфере;
- какие знания и навыки необходимы этим специалистам для работы с сетями;
- как связаны сферы бэкенд, фронтенд и сетевая инфраструктура.

Также вам нужно будет самостоятельно углубиться в тему протоколов: найти в Интернете, за что отвечает каждый из слоёв стека TCP/IP,

обозначить на mind map все четыре слоя и кратко описать, какую задачу решает каждый слой и какие конкретно протоколы в них задействованы.

В этом юните вы узнаете, что такое **бизнес-аналитика,** и поймёте, для чего крупные компании собирают о нас столько данных, а также как эти данные можно использовать на пользу бизнесу.

Пример. Представьте ситуацию, что в один прекрасный день вы открываете своё любимое приложение (к примеру, всё то же Яндекс.Такси) и замечаете, что кнопка заказа изменила цвет или переехала в другую часть экрана. Казалось бы, мелочь, но на самом деле за каждым таким изменением стоит тонна проанализированной информации и часы работы аналитиков. Ни одно изменение не вносится просто так — оно всегда решает определённую задачу или проблему: например, пользователю было неудобно попадать по слишком маленькой кнопке, или же она сливалась с фоном, её было сложно найти на экране.

В повседневной жизни мы часто сталкиваемся с тем, что услуги и продукты, которыми мы пользуемся, не стоят на месте: меняется ассортимент продуктов в магазинах, интерфейсы сайтов и приложений, появляются новые виды услуг, а какие-то наоборот исчезают.

На основе чего бизнесы принимают все эти решения? В этом им помогает Business Intelligence.

Business Intelligence (BI, бизнес-аналитика) — это направление, которое занимается анализом и обработкой информации, переводит её в осмысленную и удобную форму и затем использует для решения задач бизнеса.

Задание «на подумать»

Согласно опросам пользователей популярных социальных сетей, многие из них жалуются на то, что их лента перегружена и в ней сложно найти тот контент, который хочется в данный момент: обновления друзей, новости с информационных аккаунтов, полезные посты, развлекательный контент и т.д. Какое решение можно предложить для этой проблемы?

BI применяется во множестве сфер: финансовые организации, банки, логистические компании, ритейл, ресторанный бизнес, нефтедобывающая промышленность и многие другие.

Как правило, услугами бизнес-аналитиков пользуются крупные компании, которые имеют дело с огромными масштабами данных.

Сложно представить, какие объемы данных генерируют люди каждый день: покупки в магазинах, банковские операции, данные о пользовательском поведении на сайтах и в приложениях, информация с различных датчиков и камер видеонаблюдения. Все эти данные заботливо собираются и обрабатываются компаниями. Зачем им это нужно?

Существует много причин, почему такие данные могут быть полезны:

- дают наглядную картину поведения потребителей и их предпочтений;
- отображают изменения на рынке и позволяют быстро отреагировать на них;
- помогают лучше определить целевую аудиторию;
- показывают, как можно оптимизировать бизнес-процессы и сократить издержки;
- помогают принимать стратегические решения о развитии бизнеса.

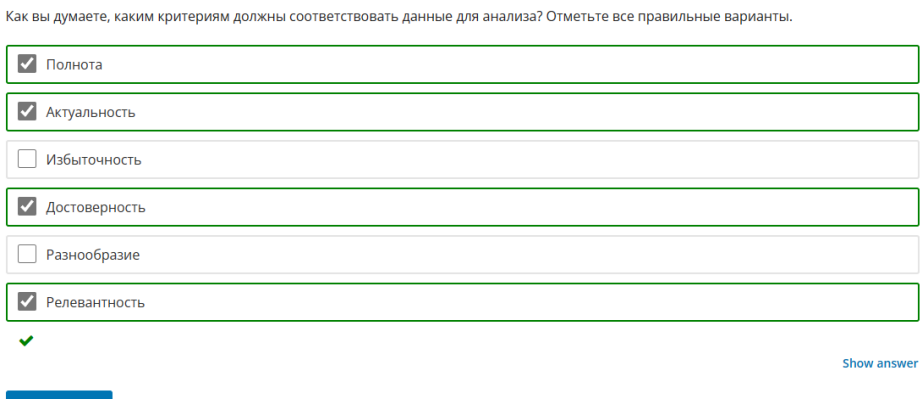
Простой пример: у многих из нас есть умные часы, которые собирают информацию о том, сколько шагов мы прошли за день, как много пробежали, сколько часов спали ночью и как глубоко. При этом в приложении, как правило, содержится базовая информация о вас: возраст, пол, вес. Представьте, как ценна была бы эта информация, к примеру, для производителей спортивной обуви или лекарств от проблем со сном. Без особых усилий они могли бы получить огромный сегмент целевой аудитории.

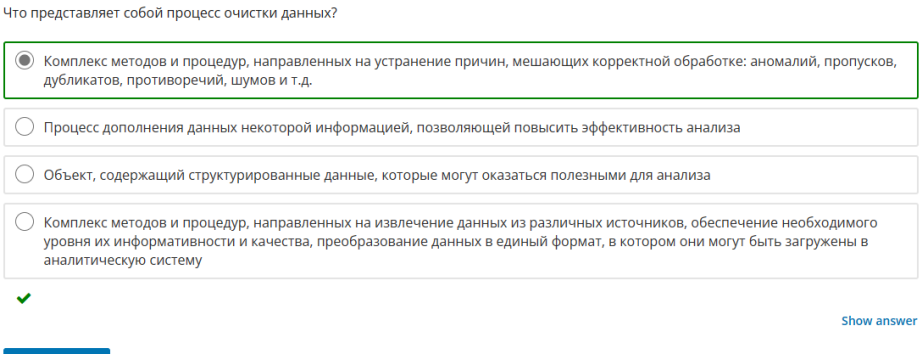
Основная цель BI — обработать большой объем неупорядоченных данных, выделить в них главное, представить в читабельном виде, а в некоторых случаях ещё и смоделировать дальнейшее развитие событий.

Процесс работы бизнес-аналитика схематически выглядит так:

На этапе **«сырых» данных** аналитик собирает данные из всех доступных источников. Это могут быть различные отчёты, выгрузки из баз данных, статистические данные с сайтов и социальных сетей и так далее. Проблема этих данных в том, что они не формализованы (не приведены к единому формату), могут содержать дублирующуюся информацию и ошибки.

На следующем шаге — этапе **«чистых» данных** — аналитик сводит все данные из разных источников в одну формализованную таблицу, фильтрует данные, избавляясь от ошибочных или неполных записей, и приводит их в удобный для работы вид.





На этапе **стандартного анализа** аналитик составляет стандартные типы отчётов, которые не требуют сложных математических операций. Например, посчитать средний чек за месяц, эффективность работы рекламной кампании или сколько сделок было заключено каждым менеджером. Если аналитик использует в своей работе специализированное ПО, этот этап может быть частично или полностью автоматизирован.

Этап **интеллектуального анализа** ставит перед аналитиком уже гораздо более сложную задачу. Если на предыдущем этапе ему нужно было описать то, что фактически произошло (упали продажи, увеличилась текучка кадров, снизилась эффективность менеджеров), то на этом этапе главная задача — понять причину произошедшего. Для этого уже может быть недостаточно данных, собранных внутри компании.

Представим себе, что у фирмы, продающей заграничные туры, упала выручка. Почему это могло случиться? Причин может быть масса: падение качества услуг, удачная рекламная кампания у конкурентов, падение доходов населения, меры государства по поддержке внутреннего туризма, погодные условия.

Задача аналитика — проверить различные гипотезы, подтвердить или опровергнуть их. Для этого используются различные методы математического и статистического анализа, OLAP-кубы и Data Mining. Вкратце разберём, что это такое.

OLAP-куб — это инструмент работы с данными, который позволяет сгруппировать данные по одинаковому признаку (категория, имя, дата) и далее делать с ними дополнительные действия или вычисления. По своей сути OLAP-куб похож на сводную таблицу.

Пример: у нас есть данные о результатах рекламной кампании. Фирма продвигала кампанию по трём каналам: на своём сайте, в Инстаграм* и с помощью холодных звонков. По каждому каналу поступали заявки, но только часть заявок приводила к покупке. Сможете ли вы что-то понять об эффективности кампании, глядя на такую таблицу?

	A	B	C
1	**№ заявки**	**Канал**	**Покупка совершена**
2	1	Реклама в Instagram	нет
3	2	Сайт	да
4	3	Холодные звонки	нет
5	4	Холодные звонки	да
6	5	Реклама в Instagram	да
7	6	Холодные звонки	нет
8	7	Сайт	да
9	8	Реклама в Instagram	нет
10	9	Сайт	нет
11	10	Сайт	нет
12	11	Реклама в Instagram	да
13	12	Реклама в Instagram	да
14	13	Сайт	да
15	14	Холодные звонки	да
16	15	Реклама в Instagram	нет
17	16	Реклама в Instagram	да

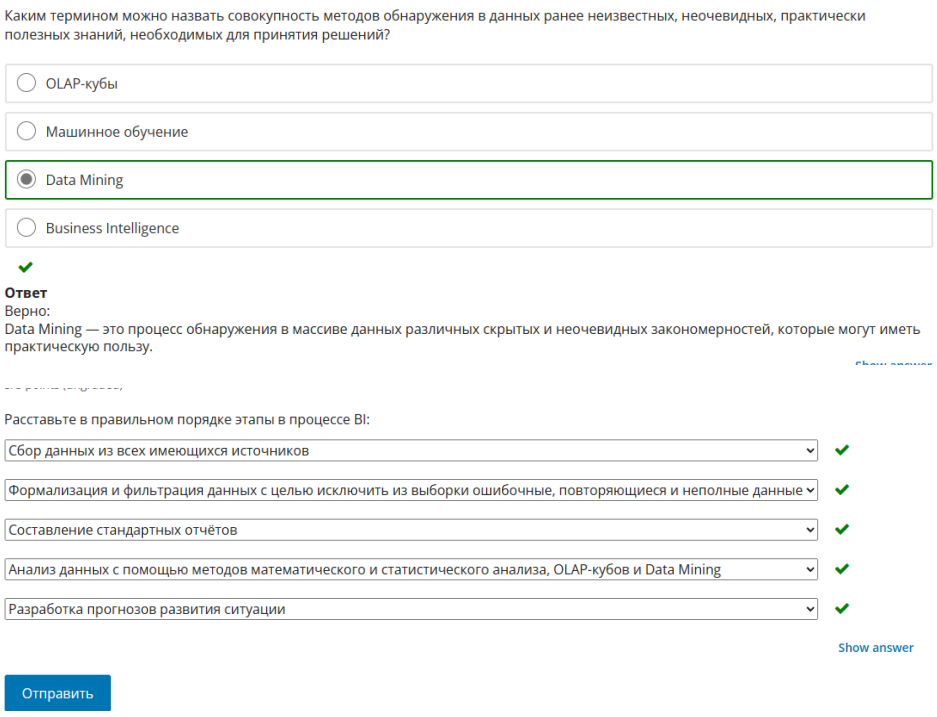
Однако, если сгруппировать заявки по каналам и выделить отдельно заявки, приведшие к покупке, данные принимают краткий и удобный для восприятия вид:

E	F	G	H	I
	Реклама в Instagram	**Сайт**	**Холодные звонки**	**Всего**
Кол-во заявок	18	22	10	50
Кол-во покупок	12	7	2	21

Data Mining — это процесс обнаружения в массиве данных различных скрытых и неочевидных закономерностей, которые могут иметь практическую пользу. Data Mining часто используется, например, в банковской сфере для определения случаев мошенничества.

На последнем этапе задача специалиста — **предсказать** возможные варианты развития ситуации на основе имеющихся данных. Обычно разрабатывают три сценария: пессимистичный, оптимистичный и реалистичный. Для этого используют сложные математические модели и **машинное обучение (Machine Learning).** Машинное обучение будет подробнее разобрано в дальнейшем, но в двух словах это понятие можно охарактеризовать как способность искусственного интеллекта делать предсказания о будущем развитии событий, основываясь на информации о том, как события развивались в прошлом.

Все пять этапов бывают задействованы только в крупных компаниях, потому что у них имеются достаточные объёмы данных для анализа, а также ресурсы на его проведение, так как этот процесс довольно дорогостоящий. Более мелким бизнесам обычно достаточно результатов, достигнутых на третьем этапе.



Для анализа могут быть использованы самые различные инструменты:

- **Excel** (который содержит в себе большое количество продвинутых инструментов);
- **Язык Python** (язык программирования, одно из самых популярных решений для анализа данных сегодня);
- **Язык R** (язык программирования, предназначенный для статистической обработки данных);

- **Специализированное ПО.**

Результаты аналитики красиво оформляют, выделяя только самую важную информацию и делая акцент на визуальное представление. Обычно итоги анализа выносятся на так называемый **дашборд**, т.е. набор графиков и диаграмм с краткими пояснениями. В идеале он должен занимать не больше 1–2 страниц.

Пример дашборда

 Источник: pinterest.ru

В последнее время становится всё более распространённым **self-service** подход к аналитике.

Работа бизнес-аналитика частично автоматизируется (на основе существующего аналитического ПО или пишется свой собственный программный продукт), и разные заинтересованные лица — руководители отделов, директора, маркетологи — могут сами сформировать нужные отчеты несколькими кликами мыши. Это бывает полезно в случаях, когда есть частая необходимость в строго формализованных отчетах, и помогает разгрузить аналитический отдел для более сложных задач.

Обычно специалиста, который занимается Business Intelligence, называют **бизнес-аналитиком**, иногда используется термин **data scientist**. Но в фирмах, где работают с большими объемами данных и решают задачи более высокой сложности, могут быть заняты и более узконаправленные специалисты:

- **Специалисты по Big Data**
  Big Data — это данные огромных объемов, которые сложно обрабатывать стандартными методами.
- **Специалисты по Data Mining**
- **Специалисты по машинному обучению**

Задание на mind map

Отметьте на mind map знания о сфере BI, полученные в данном юните.

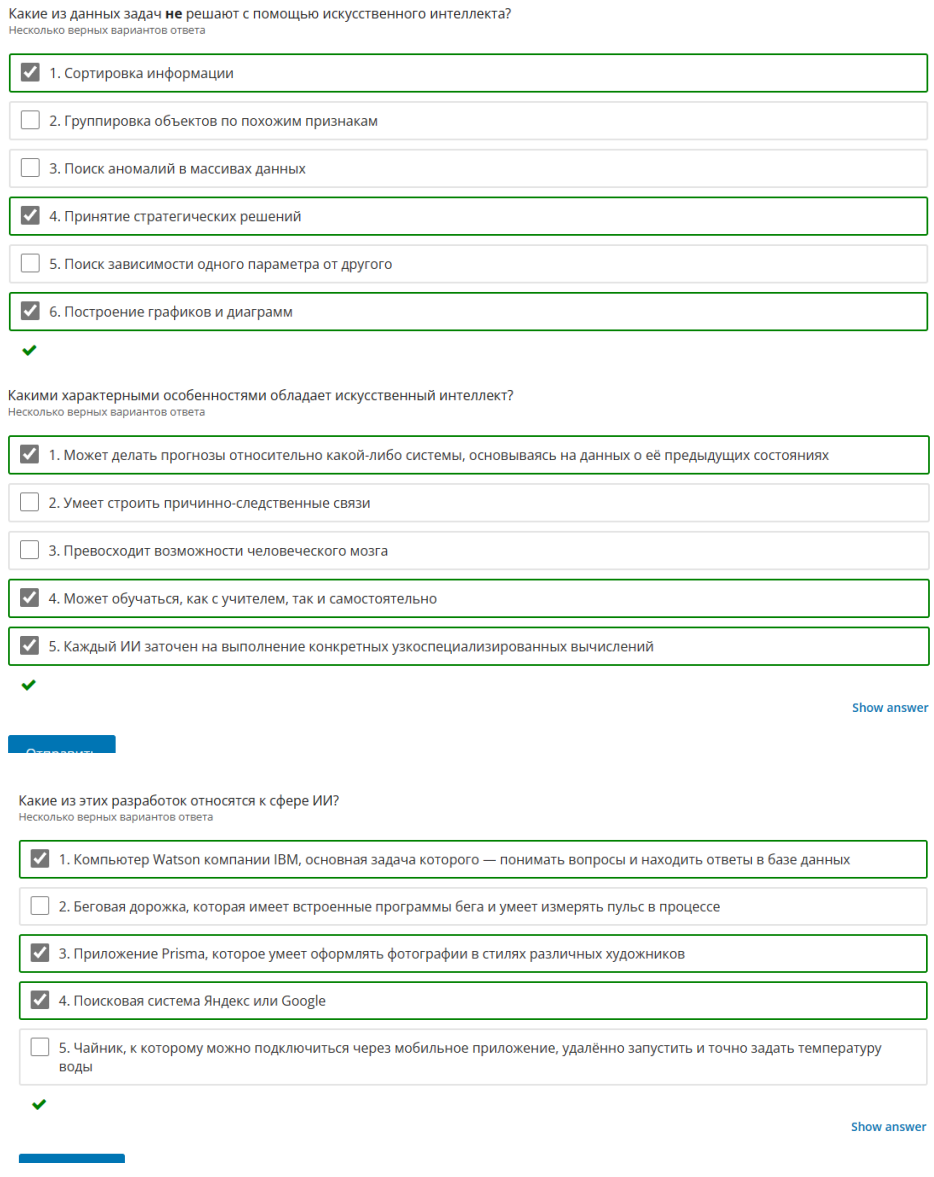
Отобразите на карте следующую информацию:

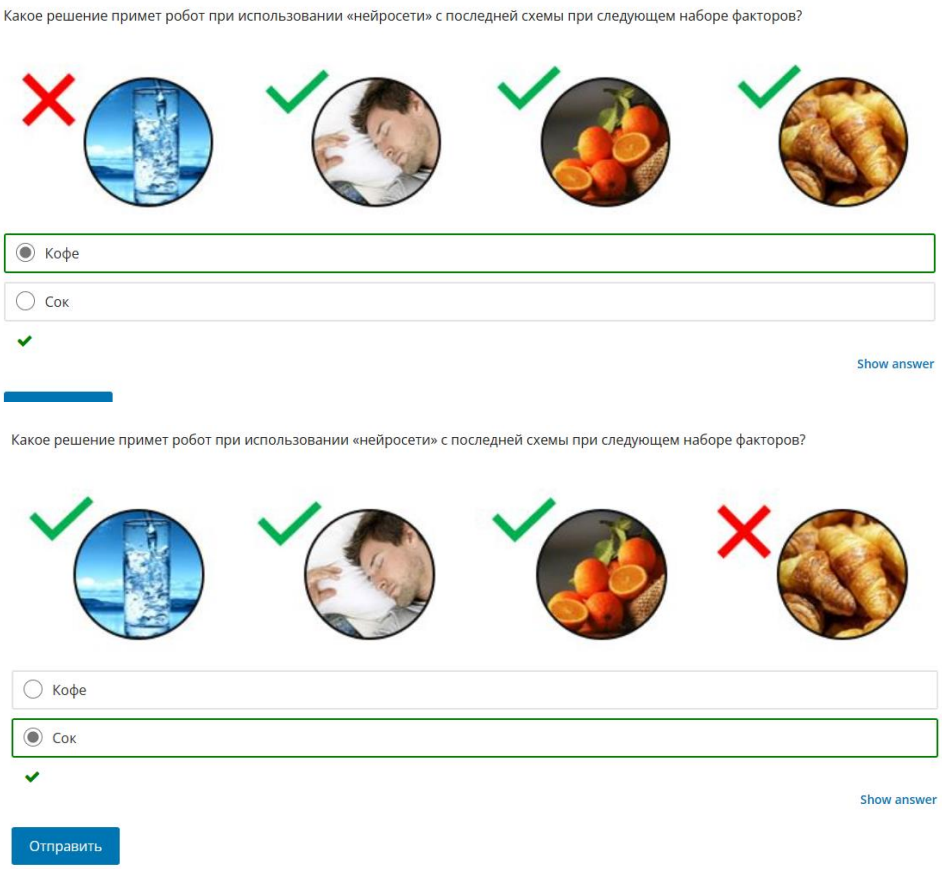
- какие задачи решает сфера BI;
- 5 этапов, из которых состоит работа бизнес-аналитика;
- какие технологии используются при анализе;

- какие специалисты могут быть задействованы в сфере.

Также вам нужно будет самостоятельно найти примеры 3–4 конкретных программных продуктов, которые используются в бизнес-анализе, и добавить их на карту.

**В материалах курса упоминается социальная сеть Фейсбук/Инстаграм, принадлежащая Meta Platforms Inc., деятельность которой запрещена на территории РФ в части реализации данных социальных сетей на основании осуществления ею экстремистской деятельности.*





В этом юните мы разберём тему искусственного интеллекта: что это такое, какие задачи решает, что уже «умеет» искусственный интеллект, а чему пока только учится. Также мы рассмотрим, какие бывают подходы к машинному обучению и немного заглянем «под капот» нейросетей и попробуем понять, как они работают.

Пример. Продолжим разбирать пример с приложением Яндекс.Такси. Цена поездки — это не просто количество километров, умноженное на тариф. При подсчёте цены учитывается большое количество факторов: пробки на дороге (не только в текущий момент, но и пробки, которые скоро возникнут или исчезнут), день недели и время суток, количество водителей и пассажиров в районе и даже погода. Такой сложный алгоритм расчёта нужен для того, чтобы цена была максимально сбалансированной и была привлекательной как для пассажира, так и для водителя. Проводить такие сложные расчёты приложению помогает искусственный интеллект.

В недалеком прошлом искусственный интеллект был скорее чем-то из разряда фантастики, но уже сегодня мы пользуемся разработками, основанными на ИИ, и часто даже не замечаем этого. «Умные» клавиатуры на телефонах с подсказками ввода, поисковики, беспилотные автомобили,

технологии распознавания лиц и речи, голосовые помощники и онлайн-переводчики — эти и многие другие привычные нам вещи основаны на искусственном интеллекте. Давайте разберёмся, что же это такое и как он работает.

Искусственный интеллект (Artificial Intelligence, AI) — это технология, которая обучает компьютеры и роботизированную технику решать задачи и мыслить так, как это делает человек. По сути, искусственный интеллект имитирует работу человеческого мозга, хотя и не в полной мере.

Сфера ИИ — невероятно перспективное направление. К 2025 году оценка рынка может достигнуть 190 млрд. долларов. Постоянно появляются всё новые исследовательские центры и компании, занимающиеся разработкой и исследованиями в области ИИ.

В сфере востребованы такие специалисты, как:

- **Разработчик ИИ** — программист, владеющий подходами и принципами разработки ИИ.
- **Архитектор ИИ** — определяет области, где имеет смысл внедрять искусственный интеллект, а также контролирует производительность и устойчивость системы.
- **Специалист по Big Data** — специалист, разбирающийся в том, как обрабатывать и анализировать большие объемы данных. Роль такого специалиста важна, потому что качество ИИ сильно зависит от качества данных, которые он получает на входе.
- **Специалист по машинному обучению** — реализует процесс обучения ИИ и разрабатывает меры по его развитию и улучшению показателей

Искусственный интеллект делят на *слабый* и *сильный.* Слабый ИИ ориентирован на решение определённого узкого круга задач. Сильный ИИ может решать разноплановые задачи, искать различные подходы к их решению и в целом вести себя подобно человеку. На самом деле, подавляющее большинство ИИ, существующих в настоящий момент, являются слабыми. Разработками сильного ИИ сейчас занимаются многие технологические гиганты.

Главная цель разработок в ИИ — создание аналитической системы, которая будет способна логически мыслить и принимать рациональные решения, подобно человеку, при этом работать быстрее и исключать ошибки так называемого «человеческого фактора».

Во многих сферах ИИ уже значительно превосходит человека. Например, программа для игры в го — *AlphaGo*, разработанная Google в 2015 году, уже в марте 2016 года смогла выиграть партию со счётом 4:1 у действующего чемпиона этой игры. При этом го — игра гораздо более сложная, чем шахматы, в ней существует более чем комбинаций, а для победы требуется особым образом подбирать стратегию игры.

Однако не стоит переживать насчёт того, что машины скоро заменят людей во всех сферах. Существует множество направлений, в которых ИИ не может сравниться с человеком (по крайней мере пока что):

- творческая деятельность (создание чего-то нового);
- мультизадачность (человек может легко сочетать разные виды деятельности, машины же заточены под конкретные задачи и не могут выходить за их рамки);
- построение причинно-следственных связей.

Вы можете попробовать сами поиграть с ИИ. Ниже есть несколько игр, которые полностью основаны на искусственном интеллекте. Кроме удовольствия от игры, это может дать некоторое понимание о возможностях ИИ и о том, как он работает.

A.I.Duet

A.I.Duet — этот ИИ умеет сочинять продолжения к мелодиям, которые вы наиграете на пианино.

Semi-Conductor

Почувствуйте себя дирижером оркестра.

Auto Draw

Auto Draw — по наброскам определяет, что вы пытаетесь изобразить, и предлагает варианты законченного рисунка.

Яндекс.Автопоэт

Яндекс.Автопоэт — составляет шуточные стихотворения, рифмуя поисковые запросы.

Можно выделить несколько типов задач, которые решают с помощью искусственного интеллекта.

Классификация
Задача заключается в распределении объектов по изначально заданным признакам. К примеру, фотографии собак ИИ отнесёт к группе «животные», фотографии цветов — к группе «растения» и так далее. Часто используется в спам-фильтрах, поиске похожих объектов, распознавании рукописного текста и др.

Регрессия
В таких типах задач нужно найти зависимость одного параметра от другого и предсказать его значение. Пример: предсказания биржевых котировок, погоды или пробок на дорогах.

Кластеризация
При решении задач кластеризации алгоритм группирует объекты по похожим признакам. Это используется при распознавании объектов на фото, в поисковых механизмах, социологических исследованиях.

Обнаружение аномалий
ИИ ищет данные, которые по тем или иным признакам выбиваются из общего массива. Наиболее

часто это используется в банковской сфере для обнаружения мошенников.

Ассоциация
Задача заключается в нахождении закономерностей между тем, как признаки одного объекта соотносятся с признаками других объектов. Пример: если вы заказываете в Интернет-магазине пряжу, сайт может порекомендовать вам наборы спиц и книги со схемами для вязания, потому что все эти предметы имеют общий признак: они относятся к вязанию.

С ИИ тесно связаны понятия нейронной сети и машинного обучения.

Машинное обучение

Машинное обучение — это направление в науке, которое занимается обучением аналитических систем решению конкретных задач.

Разделяют несколько видов машинного обучения:

- **С учителем**. При данном подходе к машинному обучению учитель (как правило, человек) даёт компьютеру вводную информацию и эталонные выходные данные. На основе этих данных компьютеру требуется найти зависимость между входными и выходными данными и научиться давать точный ответ для любого набора данных. Чем больше и обширнее обучающая выборка, тем точнее будет анализ.
- **Без учителя**. В данном случае ИИ также должен определить зависимость между входными и выходными данными, но на этот раз без обучающей выборки. Такой подход чаще всего используется для задач, где необходимо найти неочевидное решение.

- **Глубокое обучение**. Глубокое обучение может применяться как с участием учителя, так и без. Этот алгоритм использует многослойную схему вычислений. Каждый последующий слой получает результаты вычислений предыдущего слоя. Глубокое обучение часто используют для анализа *Big Data* (больших объемов данных), в компьютерном зрении, распознавании речи и многих других сферах.
- **Нейросети** — математическая модель, построенная по аналогии с нейронными связями в человеческом мозге. Разберём их немного подробнее.

Нейросети

В человеческом мозге нейроны взаимодействуют друг с другом посредством специальных каналов, по которым они обмениваются информацией. В **нейросети** в роли нейронов выступают отдельные вычислительные единицы, которые получают данные и проводят над ними определенные расчёты, чтобы определить, какое значение нужно передать дальше по цепочке.

Нейросеть можно проиллюстрировать так:

Желтым цветом обозначен **входной слой** — он принимает сигнал, но не обрабатывает его, а передает дальше.

Синим цветом выделены **скрытые слои**. Их может быть сколько угодно много. Именно на этом уровне происходит обработка информации.

Красным цветом обозначен **выходной слой**, который выводит результаты вычислений скрытого слоя.

Нейронные сети не программируются в привычном смысле этого слова, они **обучаются.** Обучение заключается в том, что нейросети дают решать задачу (например, найти среди набора картинок фотографии собак). После каждого полученного решения оценивают величину ошибки и запускают процесс заново до тех пор, пока величина ошибки не будет достаточно мала.

В случае успешного обучения сеть будет способна вернуть верный результат на основании данных, которые не использовались при обучении, а также неполных или частично искажённых данных.

Если вы когда-либо проходили проверку Google reCaptcha, в которой нужно найти на изображении автомобиль, светофор или витрину, знайте, что вы приняли участие в обучении нейросети, которую Google планирует использовать для управления беспилотными автомобилями. Решая такую несложную для человека задачку, пользователи, сами того не зная, помогают откалибровать и проверить работу искусственного интеллекта.

Пример с роботом-помощником

Давайте попробуем понять логику работы нейронной сети на практическом примере. Представим, что нам необходимо запрограммировать робота-помощника, который будет на основании образа жизни, состояния и предпочтений человека решать, что приготовить ему на завтрак. Для простоты объяснения ограничимся только напитками и будем делать выбор из 2-х вариантов: кофе или свежевыжатый апельсиновый сок.

За долгий период сбора данных роботу удалось установить следующие взаимосвязи:

- если человек с утра испытывает жажду, он выбирает сок;
- если человек не выспался, он предпочитает пить кофе;
- если дома есть любимый сорт апельсинов, часто выбор падает в сторону сока;
- если на завтрак будут круассаны, чаще всего человек выбирает кофе.

То есть на уровне входных нейронов у нас есть 4 фактора (жажда, не выспался, апельсины и круассан), каждый из которых влияет на принятие итогового решения. Каждый фактор должен иметь свой **вес**, то есть конкретное числовое значение, с которым можно производить вычисления.

Нахождение правильных весов для каждого фактора является одной из самых сложных задач при обучении нейросети. Веса нужно подобрать так, чтобы отобразить реальную зависимость итогового значения от входных данных.

Зададим следующие критерии принятия итогового решения: если сумма факторов **больше или равна 0.5**, то робот приготовит **кофе, если меньше 0.5**, то приготовит **сок**. Тогда распределим веса следующим образом:

Как мы видим, факторы имеют различные веса. Наиболее значительный фактор — то, что человек не выспался, далее идёт жажда и апельсины с круассанами.

Давайте попробуем подсчитать итоговый показатель на конкретном примере:

То есть, если человек не выспался и испытывает жажду, в холодильнике нет нужного сорта апельсинов и на завтрак будут круассаны, робот приготовит кофе.

Главная особенность нейросетей заключается в том, что, добавляя слои, мы можем усложнять процесс принятия решений и делать обобщения. В реальной жизни имеют значения не только сами факторы, но и их сочетания.

Добавим условия в задачу: предположим, что ИИ робота вывел в процессе наблюдения ещё одну взаимосвязь:

- если человек испытывает жажду и дома есть любимый сорт апельсинов, он в любом случае пьет сок, вне зависимости от остальных факторов.

Тогда нам понадобится добавить ещё один промежуточный слой нейронов (он называется скрытым). Нейроны на скрытом уровне функционируют похожим образом с нейроном на выходном уровне: если сумма сигналов с предыдущих нейронов больше или равна 0.5, то сигнал передаётся дальше, если сумма меньше 0.5 — сигнал на следующие нейроны не передаётся (или можно сказать, что передаётся сигнал с нулевым значением, так как число 0 не может влиять на итоговый результат).

Учитывая эти условия, перерисуем схему и перераспределим веса:

Первый нейрон на скрытом уровне отвечает за проверку сочетания факторов «жажда» + «апельсины». При их наличии сумма факторов будет равна 0.5 и сигнал будет передан дальше, остальные факторы не имеют значения, поэтому их веса на для данного нейрона равны нулю. Если хотя бы один из факторов будет отсутствовать (не испытывает жажду или нет

апельсинов), то сумма весов будет меньше 0.5 и сигнал с этого нейрона не будет влиять на результат.

Второй нейрон отвечает за принятие решение в случае, если сочетание факторов «жажда» + «апельсины» отсутствует. В таком случае веса факторов остаются такими же, как были до этого.

Давайте представим, что все факторы положительны и попробуем подсчитать итоговый результат:

На первом нейроне выполняется условие сочетания факторов, и на выходе передается значение -1, чтобы перекрыть все возможные положительные значения с других соседних нейронов (так как мы знаем, что сочетание «жажда» + «апельсины» должно вести к однозначному результату). На втором нейроне сумма факторов даёт значение 0.3, что меньше, чем 0.5, поэтому дальше передаётся нулевой сигнал, который не влияет на дальнейший результат.

Задание на mind map

Добавьте на ментальную карту сферу искусственного интеллекта.

Отобразите на карте следующую информацию:

- какие задачи решает ИИ;
- какие специалисты работают в этой сфере;
- добавьте на карту сферу машинного обучения (как часть сферы ИИ) и обозначьте, какие подходы к машинному обучению существуют;
- приведите примеры сервисов или приложений, которые основаны на ИИ;
- отобразите связь сферы ИИ со сферой бизнес-аналитики.

Дополнительно

- Ресурс, посвященный машинному обучению
- AI Experiments with Google
- Документальный фильм об AlphaGo (AlphaGo — The Movie)