**1 What is Software Engineering?**

Software engineering is the process of creating, designing, and building computer programs and applications. It involves using various techniques and tools to make software that performs tasks and functions on devices like computers and smartphones.

Разработка программного обеспечения — это процесс создания, проектирования и создания компьютерных программ и приложений. Он предполагает использование различных методов и инструментов для создания программного обеспечения, которое выполняет задачи и функции на таких устройствах, как компьютеры и смартфоны.

**2 What is the Waterfall method?**

The Waterfall method is a traditional approach to software development. It involves a linear and sequential process where each phase must be completed before the next one begins. The phases typically include requirements gathering, system design, implementation, testing, deployment, and maintenance. Once a phase is finished, it rarely goes back to make changes. This method is often used for projects with well-defined and unchanging requirements.

The Waterfall method is a way of doing things step by step, one after the other. Each step is completed before moving to the next, like a waterfall flowing down in stages.

Метод «Водопад» — это способ делать дела шаг за шагом, одно за другим. Каждый шаг завершается перед переходом к следующему, подобно водопаду, стекающему поэтапно.

Метод водопада — традиционный подход к разработке программного обеспечения. Он включает в себя линейный и последовательный процесс, в котором каждый этап должен быть завершен до начала следующего. Этапы обычно включают сбор требований, проектирование системы, внедрение, тестирование, развертывание и обслуживание. После завершения этапа редко кто возвращается для внесения изменений. Этот метод часто используется для проектов с четко определенными и неизменными требованиями.

**3 What is the difference between alpha and beta testing?**

**Alpha Testing** is the first phase of software testing where the software is tested by internal testers. It's not released to the public.

**Beta Testing** is the second phase where the software is released to a select group of external users for testing in a real environment.

Альфа-тестирование — это первый этап тестирования программного обеспечения, на котором программное обеспечение тестируется внутренними тестировщиками. Это не обнародовано.

Бета-тестирование — это второй этап, на котором программное обеспечение предоставляется избранной группе внешних пользователей для тестирования в реальной среде.

**4 What kind of processes are there in Software Testing?**

Common processes in software testing include Test Planning, Test Case Design, Test Execution, Defect Reporting, Regression Testing, and Test Closure.

Общие процессы тестирования программного обеспечения включают планирование тестирования, разработку тестового сценария, выполнение теста, отчетность о дефектах, регрессионное тестирование и закрытие теста.

**5 What is Agile software development?**

Изображение выглядит как текст, Шрифт, диаграмма, снимок экрана

Автоматически созданное описание

Agile is an iterative and flexible approach to software development, emphasizing collaboration, customer feedback, and small, incremental releases. It includes methodologies like Scrum, Kanban, and Extreme Programming (XP).

Agile software development is a flexible and collaborative way of creating computer programs or applications. It involves working in small steps, making changes as needed, and getting feedback from users to build better software.

Гибкая разработка программного обеспечения — это гибкий и совместный способ создания компьютерных программ или приложений. Он включает в себя работу небольшими шагами, внесение изменений по мере необходимости и получение отзывов от пользователей для создания более качественного программного обеспечения.

Agile — это итеративный и гибкий подход к разработке программного обеспечения, в котором особое внимание уделяется сотрудничеству, обратной связи с клиентами и небольшим, инкрементальным выпускам. Он включает в себя такие методологии, как Scrum, Kanban и Extreme Programming (XP).

**6 What are Product Requirements?**

Product requirements are detailed descriptions of what a software product should do. They include functional (features, capabilities) and non-functional (performance, security) requirements.

Требования к продукту — это подробное описание того, что должен делать программный продукт. Они включают функциональные (функции, возможности) и нефункциональные (производительность, безопасность) требования.

**7 What is the main difference between Requirement Elicitation and Requirements Specification?**

Requirement Elicitation is the process of gathering and discoveringrequirements from stakeholders and various sources.

Requirements Specification involves documenting and defining gathered requirements in a clear and structured manner, creating a formal document.

Requirement elicitation is about collecting and gathering information from users and stakeholders to understand what a software system needs to do. It's like asking people what they want from the software.

Requirements specification is about documenting and writing down the collected information in a detailed and organized way. It's like creating a clear list of what the software should do based on what people have said.

Выявление требований заключается в сборе информации от пользователей и заинтересованных сторон, чтобы понять, что должна делать программная система. Это все равно, что спрашивать людей, чего они хотят от программного обеспечения.

Спецификация требований заключается в подробном и организованном документировании и записи собранной информации. Это похоже на создание четкого списка того, что должно делать программное обеспечение, на основе того, что сказали люди.

Выявление требований — это процесс сбора и выявления требований от заинтересованных сторон и различных источников.

Спецификация требований включает в себя документирование и определение собранных требований в четкой и структурированной форме, создавая формальный документ.

**8 Why do some companies like Apple pay more attention to System Modeling for their products?**

Companies like Apple focus on system modeling for their products to ensure better quality, consistency in design, and efficient use of resources. It helps them make reliable and innovative products while reducing risks and improving collaboration.

Companies like Apple pay attention to system modeling because it helps them make great products. System modeling is like making a plan for building something, so they know exactly how it will work and look. This way, they can create products that are easy to use and work really well.

Такие компании, как Apple, уделяют внимание системному моделированию, потому что оно помогает им создавать отличные продукты. Системное моделирование похоже на составление плана создания чего-либо, чтобы они точно знали, как это будет работать и выглядеть. Таким образом, они могут создавать продукты, которые просты в использовании и действительно хорошо работают.

Такие компании, как Apple, уделяют особое внимание системному моделированию своих продуктов, чтобы обеспечить лучшее качество, согласованность дизайна и эффективное использование ресурсов. Это помогает им создавать надежные и инновационные продукты, одновременно снижая риски и улучшая сотрудничество.

**9 What are Software Design Patterns?**

Software design patterns are recurring solutions to common problems in software design. They represent best practices and proven approaches to designing and structuring software systems. These patterns provide templates for solving problems and improving code quality, making it easier to maintain and scale software applications. Common examples include the Singleton pattern, Factory pattern, Observer pattern, and many more. Design patterns help developers create efficient, maintainable, and scalable software by offering tested and reusable solutions to common design challenges.

Software design patterns are like helpful instructions for computer programmers. They show the best ways to solve common problems when writing computer programs. Design patterns make it easier for programmers to build software that works well and is easy to understand.

Шаблоны проектирования программного обеспечения подобны полезным инструкциям для программистов. Они показывают лучшие способы решения распространенных проблем при написании компьютерных программ. Шаблоны проектирования упрощают программистам создание программного обеспечения, которое хорошо работает и легко понять.

Шаблоны проектирования программного обеспечения — это повторяющиеся решения распространенных проблем проектирования программного обеспечения. Они представляют собой лучшие практики и проверенные подходы к проектированию и структурированию программных систем. Эти шаблоны предоставляют шаблоны для решения проблем и улучшения качества кода, упрощая обслуживание и масштабирование программных приложений. Общие примеры включают шаблон Singleton, шаблон Factory, шаблон Observer и многие другие. Шаблоны проектирования помогают разработчикам создавать эффективное, удобное в сопровождении и масштабируемое программное обеспечение, предлагая проверенные и повторно используемые решения для распространенных задач проектирования.

**10 What is Model-driven engineering?**

Model-driven engineering is a way to create software by using visual models that represent the system's structure and behavior. These models can be automatically transformed into code, which makes development faster and less error-prone.

Модельно-ориентированное проектирование — это способ создания программного обеспечения с использованием визуальных моделей, которые представляют структуру и поведение системы. Эти модели можно автоматически преобразовать в код, что ускоряет разработку и снижает вероятность ошибок.

**11 Why is Test-Driven Development convenient in the modern software engineering field?**

Test-Driven Development (TDD) is convenient in modern software engineering because it helps catch and fix issues early in the development process. Developers write tests before writing the code, ensuring that the software functions correctly and is maintainable.

Разработка через тестирование (TDD) удобна в современнойразработке программного обеспечения, поскольку помогает выявлять и устранять проблемы на ранних этапах процесса разработки. Разработчики пишут тесты перед написанием кода, гарантируя, что программное обеспечение работает правильно и его можно обслуживать.

**12 What is the difference between Release testing & User testing?**

Release testing checks if the software is ready for deployment, focusing on technical aspects.

User testing involves real users who check if the software meets their needs and is user-friendly.

Тестирование выпуска проверяет, готово ли программное обеспечение к развертыванию, уделяя особое внимание техническим аспектам.

В пользовательском тестировании участвуют реальные пользователи, которые проверяют, соответствует ли программное обеспечение их потребностям и является ли оно удобным для пользователя.

**13 Which is not included Requirements engineering processes**

Requirements Storytelling

**14 What is the Requirement Elicitation?**

Requirement Elicitation is the process of gathering and collecting information about the needs and requirements of a system from various stakeholders

Выявление требований — это процесс сбора информации о потребностях и требованиях системы от различных заинтересованных сторон.

**15 How do companies get the benefit and earn money for open-source software solutions?**

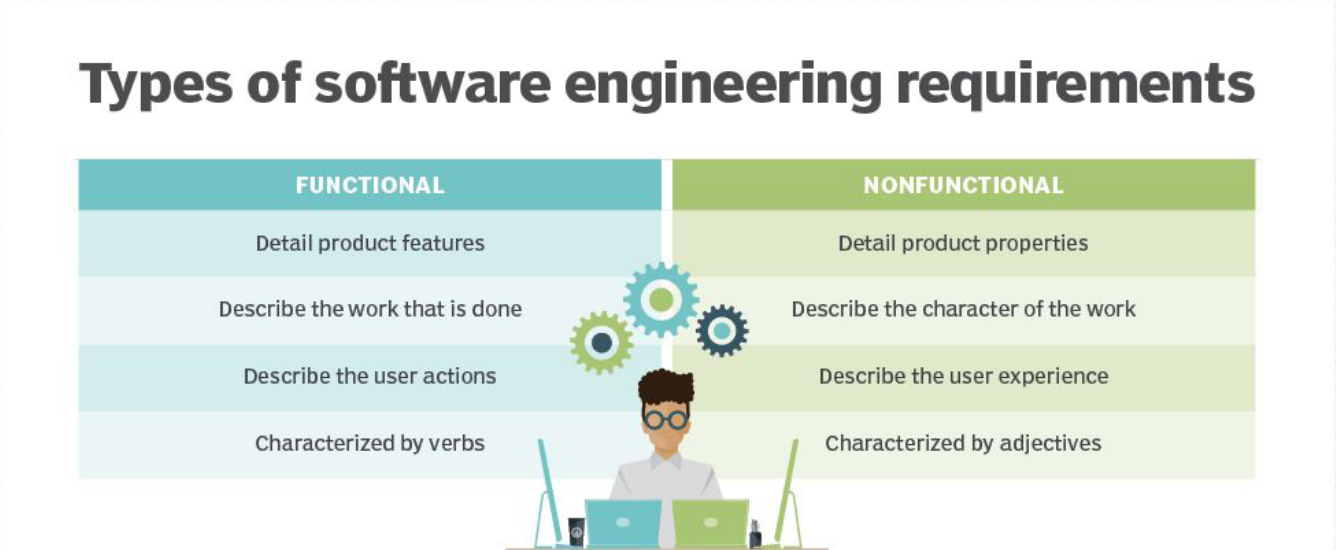
Companies benefit and earn money from open-source software solutions by offering services related to the software, such as consulting, customization, support, and training. They can also provide premium or enterprise versions with additional features and support.

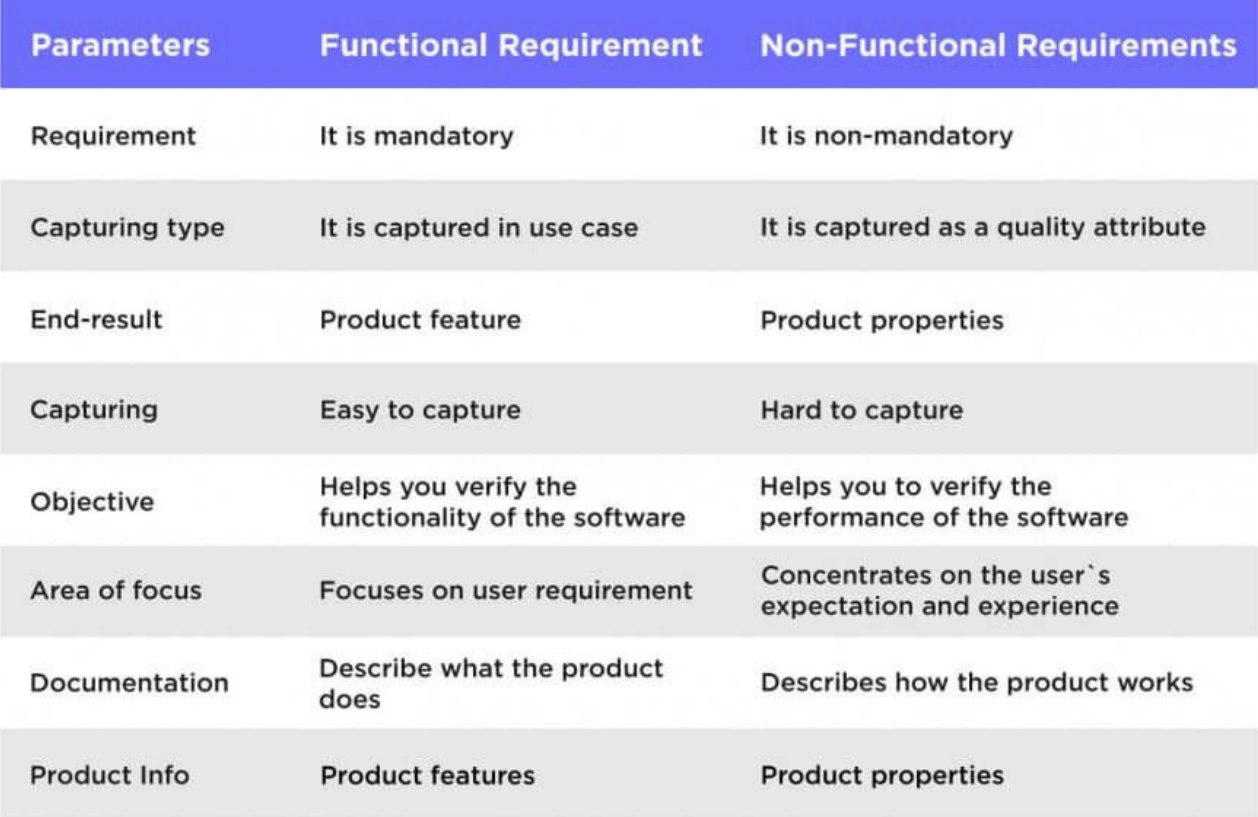
Компании получают выгоду и зарабатывают деньги на программных решениях с открытым исходным кодом, предлагая услуги, связанные с программным обеспечением, такие как консалтинг, настройка, поддержка и обучение. Они также могут предоставить премиум-версии или корпоративные версии с дополнительными функциями и поддержкой.

**16 What is not a widely used UML diagram?**

Class diagram

**17 What is the difference between functional and non-functional requirements?**





Functional requirements explain how the system must work, while non functional requirements explain how the system should perform.

Функциональные требования объясняют, как система должна работать, а нефункциональные требования объясняют, как система должна работать.

**18 What is development testing?**

Development testing (DevTest) is an approach in software development that aims to bring the development and testing phases closer together.

Тестирование разработки (DevTest) — это подход к разработке программного обеспечения, целью которого является сближение этапов разработки и тестирования.

**19 What is the Context Model?**

A context model defines how context data are structured and maintained; it plays a key role in supporting efficient context management. It aims to produce a formal or semi-formal description of the context information that is present in and to represent the reusable context information of the components.

A context model is like a simple map that shows the different parts of a system and how they connect with each other. It helps people understand how a system works without going into too many details. It's like looking at the big picture.

Контекстная модель похожа на простую карту, на которой показаны различные части системы и то, как они связаны друг с другом. Это помогает людям понять, как работает система, не вдаваясь в слишком много деталей. Это как смотреть на общую картину.

Контекстная модель определяет, как структурируются и поддерживаются контекстные данные; он играет ключевую роль в поддержке эффективного управления контекстом. Его цель — создать формальное или полуформальное описание контекстной информации, которая присутствует, и представить многократно используемую контекстную информацию компонентов.

**20 What are External Requirements?**External requirements refer to the status of other elements in the model that are not sub-components of the given element. For example, assume that we define a computer as a system, and place inside the system several sub-components (e.g., motherboard, CPU, power supply, DVD drive, monitor). Specifying that a motherboard can operate only if the power supply is operating is an external requirement with respect to the motherboard (since the power supply is external to the motherboard).

External requirements are things that come from the outside and affect how a system or product should be made. They can include laws, rules, or expectations from customers. Think of them as the rules that the system needs to follow.

Внешние требования — это вещи, которые приходят извне и влияют на то, как должна быть создана система или продукт. Они могут включать законы, правила или ожидания клиентов. Думайте о них как о правилах, которым должна следовать система.

Внешние требования относятся к статусу других элементов модели, которые не являются подкомпонентами данного элемента. Например, предположим, что мы определяем компьютер как систему и размещаем внутри системы несколько подкомпонентов (например, материнскую плату, процессор, блок питания, привод DVD, монитор). Указание того, что материнская плата может работать только при работающем блоке питания, является внешним требованием по отношению к материнской плате (поскольку блок питания является внешним по отношению к материнской плате).

**21 What is Requirements Validation?**Requirement validation is a phase of software development life cycle where requirements are validated to get rid of inconsistency, incompleteness. Stakeholders involved in the validation process to make requirements are suitable for the product.

Requirements validation is like double-checking to make sure that everything is correct and works as it should. It's like looking at the list of what needs to be done and making sure it all makes sense and fits together.

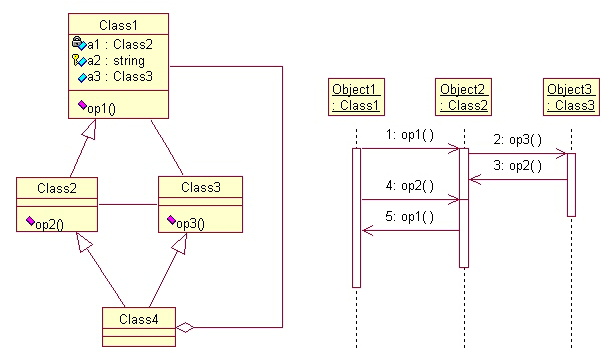
Проверка требований — это двойная проверка, позволяющая убедиться, что все правильно и работает так, как должно. Это все равно что просмотреть список того, что нужно сделать, и убедиться, что все это имеет смысл и сочетается друг с другом.

Валидация требований — это этап жизненного цикла разработки программного обеспечения, на котором проверяются требования, чтобы избавиться от несогласованности и неполноты. Заинтересованные стороны, участвующие в процессе проверки, чтобы составить требования, подходящие для продукта.

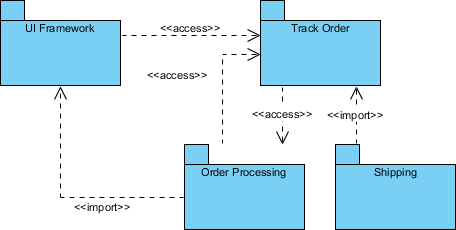
**Software engineering**

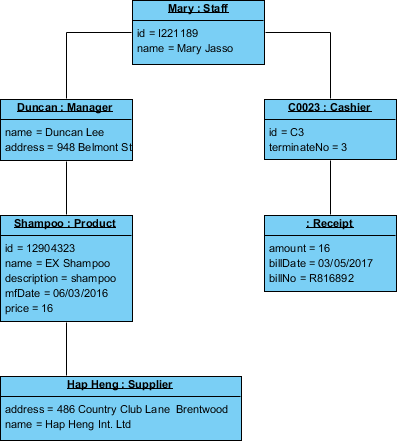
**Practice**

Class Diagrams & Sequence Diagrams

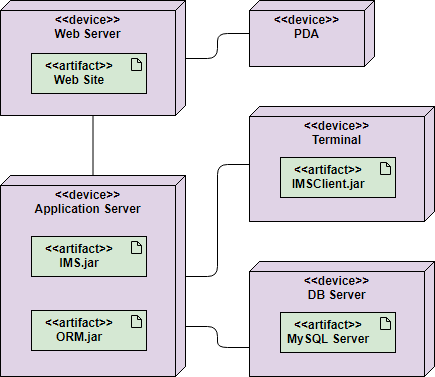


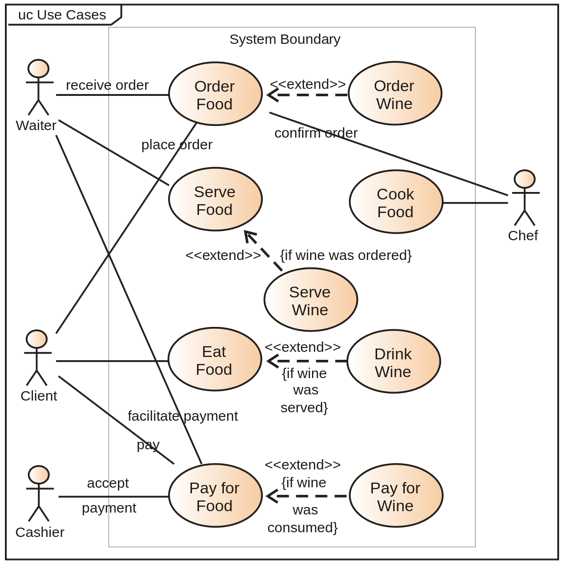
Object Diagrams & Package Diagrams

 Package diagram

 Object diagram

Deployment Diagrams & Use Cases

 Deployment Diagrams

 Use case diagram

State Machine Diagrams & Activity Diagrams

