1. **What is Software Engineering?**

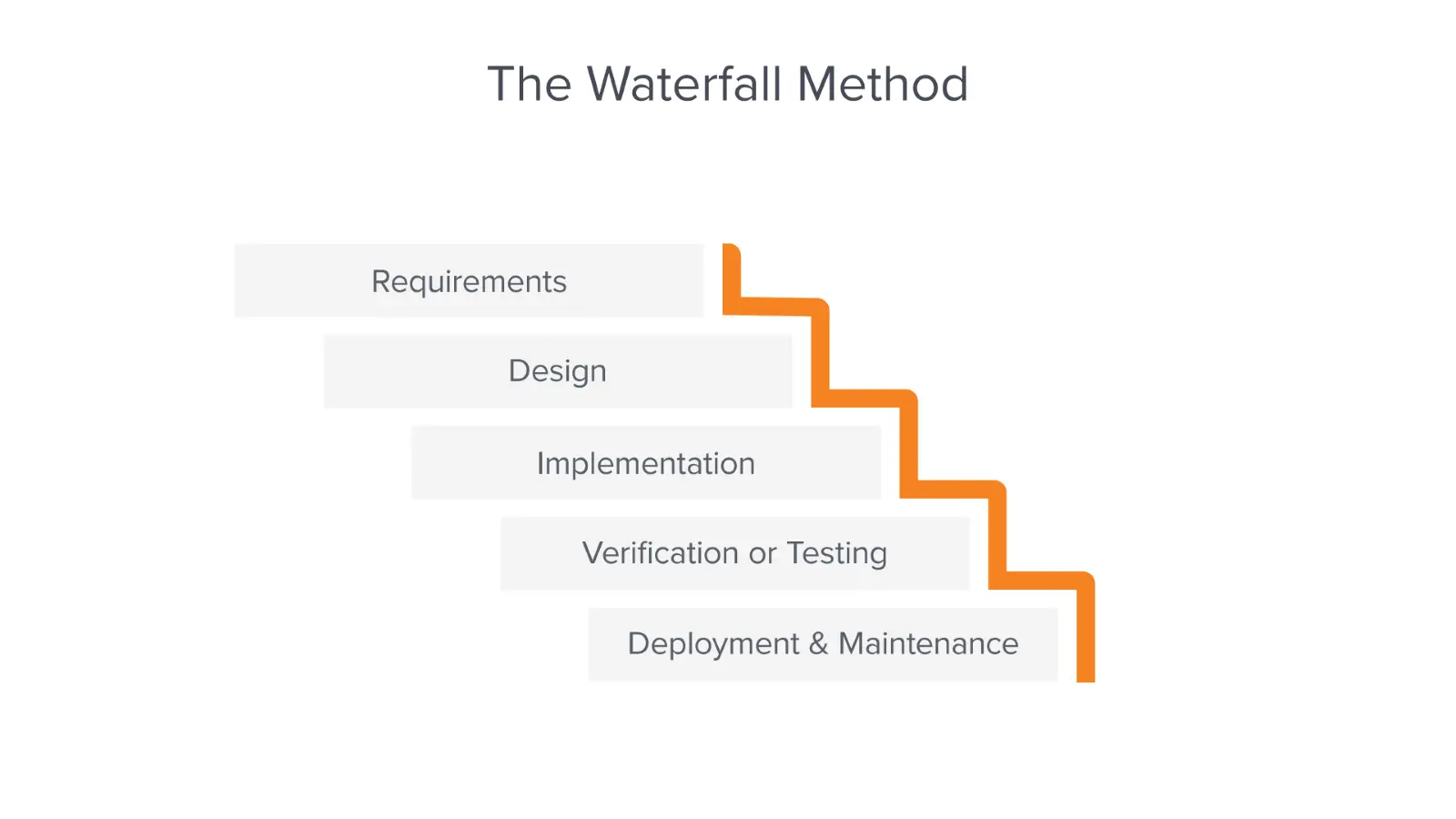
Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use. In this definition, there are two key phrases:

*1. Engineering discipline* Engineers make things work. They apply theories, methods, and tools where these are appropriate. However, they use them selectively and always try to discover solutions to problems even when there are no applicable theories and methods. Engineers also recognize that they must work within organizational and financial constraints, and they must look for solutions within these constraints.

*2. All aspects of software production* Software engineering is not just concerned with the technical processes of software development. It also includes activities such as software project management and the development of tools, methods, and theories to support software development.

1. **What is the Waterfall method?**

*The Waterfall methodology — also known as the Waterfall model* — is a widely used project management method with a linear approach. This method is a widely used project management method with a linear approach. And it is a sequential development process that flows like a waterfall through all phases of a project (analysis, design, development, and testing, for example), with each phase completely wrapping up before the next phase begins. While there are various types of project management methodologies, Waterfall is well suited for projects where the objectives are clearly outlined from the beginning.



1. **What is the difference between alpha and beta testing?***In alpha testing,* users and developers work together to test a system as it is being developed. This means that the users can identify problems and issues that are not readily apparent to the development testing team. *Alpha testing* is often used when developing software products or apps. It also reduces the risk that unanticipated changes to the software will have disruptive effects on their business. However, alpha testing may also be used when custom software is being developed.

*Beta testing* is mostly used for software products that are used in many different settings. This is important as, unlike custom product developers, there is no way for the product developer to limit the software’s operating environment. *Beta testing* is therefore used to discover interaction problems between the software and features of its operational environment.

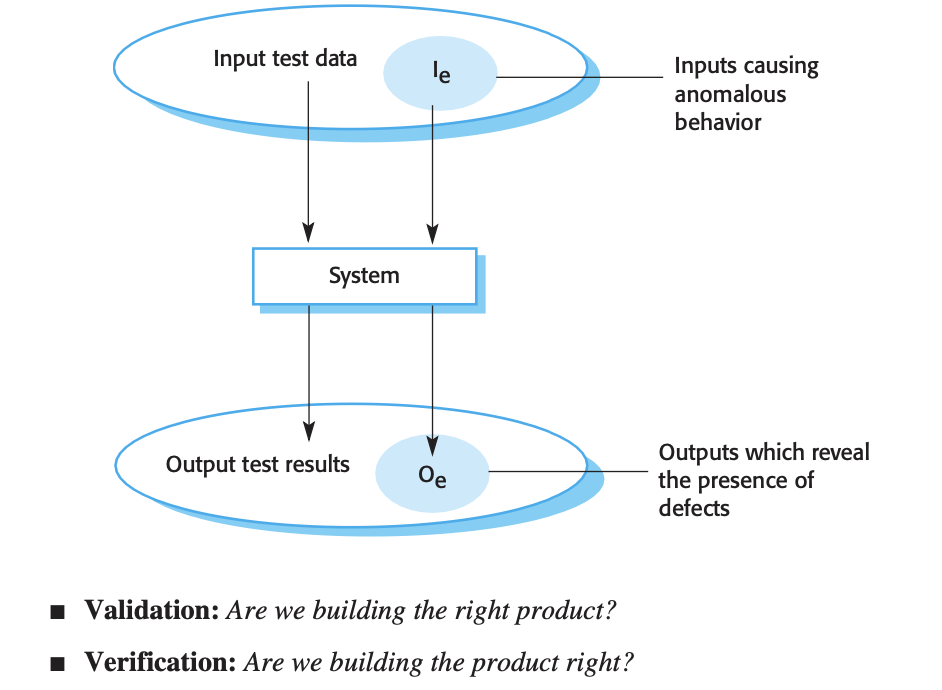
Alpha testing primarily takes place within the organization, whereas Beta testing takes place in the user's environment. Only functionality and usability are checked during Alpha Testing, whereas usability, functionality, security, and dependability are all thoroughly tested during Beta Testing.

1. **What is Agile software development?**

*Agile development* is a project management methodology that involves breaking the project into phases and emphasizes continuous collaboration and improvement. The Agile Manifesto, which was created in 2001, outlines the [four main values and twelve principles](https://agilemanifesto.org/principles.html) of Agile development.Agile development focuses on creating working software quickly, collaborating with customers frequently, and being able to adapt to changes easily. Teams follow a cycle of planning, executing, and evaluating. This methodology is especially beneficial for projects that are complex or have uncertain requirements.

1. **What kind of processes are there in Software Testing?**

Testing is part of a broader process of ***software verification and validation (V & V).*** Verification and validation are not the same thing, although they are often confused. Barry Boehm, a pioneer of software engineering, succinctly expressed the difference between them (Boehm 1979):



Verification and validation processes are concerned with checking that software being developed meets its specification and delivers the functionality expected by the people paying for the software. These checking processes start as soon as requirements become available and continue through all stages of the development process. *The goal of verification and validation processes* is to establish confidence that the software system is “fit for purpose.” This means that the system must be good enough for its intended use.

1. **What are Product Requirements?**

A product requirement is something that Product Management typically creates, providing the outline of acceptable criterial to the design and development team, so they can determine an appropriate solution to the need. Aside from the customer needs - the product’s requirements include all functions, features and behaviors that the product must possess, so that it will be efficient, ease to use, safe, low cost, etc. In other words - any function, constraint or other property that is required in order to satisfy user’s needs. User requirements are gathered from users and described from the analyst with a customer point of view.

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

The aims of the *requirements elicitation* process are to understand the work that stakeholders do and how they might use a new system to help support that work. During requirements elicitation, software engineers work with stakeholders to find out about the application domain, work activities, the services and system features that stakeholders want, the required performance of the system, hardware constraints, and so on. **Requirements elicitation focuses on describing the purpose of the system.** The client, the developers, and the users identify a problem area and define a system that addresses the problem. **Such a definition is called a requirements specification** and serves as a contract between the client and the developers. The specification also contains other related information necessary for the design, verification, and maintenance of the product.

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

System modeling helps the analyst to understand the functionality of the system and models are used to communicate with customers. Models of the existing system are used during requirements engineering.

Able to test a product or system works before building it. Can use it to find unexpected problems. Able to explore *'what if…'* questions. Can speed things up or slow them down to see changes over long or short periods of time.

Models are used during the requirements engineering process to help derive the detailed requirements for a system, during the design process to describe the system to engineers implementing the system, and after implementation to document the system’s structure and operation

1. **What are Software Design Patterns?**

Patterns are a way of reusing the knowledge and experience of other designers. Design patterns are usually associated with object-oriented design. Published patterns often rely on object characteristics such as inheritance and polymorphism to provide generality.

Patterns have made a huge impact on object-oriented software design. As well as being tested solutions to common problems, they have become a vocabulary for talking about a design. You can therefore explain your design by describing the patterns that you have used.

The Gang of Four defined the four essential elements of design patterns in their book on patterns:

1. A name that is a meaningful reference to the pattern.

2. A description of the problem area that explains when the pattern may be applied.

3. A solution description of the parts of the design solution, their relationships and their responsibilities. This is not a concrete design description. It is a template for a design solution that can be instantiated in different ways. This is often expressed graphically and shows the relationships between the objects and object classes in the solution.

4. A statement of the consequences—the results and trade-offs—of applying the pattern. This can help designers understand whether or not a pattern can be used in a particular situation.

1. **What is Model-driven engineering?**

Model-driven engineering (MDE) is an approach to software development whereby models rather than programs are the principal outputs of the development process. The programs that execute on a hardware/ software platform are generated automatically from the models. Proponents of MDE argue that this raises the level of abstraction in software engineering so that engineers no longer have to be concerned with programming language details or the specifics of execution platforms.

MDA focuses on the design and implementation stages of software development, whereas MDE is concerned with all aspects of the software engineering process. Therefore, topics such as model-based requirements engineering, software processes for model-based development, and model-based testing are part of MDE but are not considered in MDA.

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

Test-driven development (TDD) is an approach to program development in which you interleave testing and code development.You develop the code incrementally, along with a set of tests for that increment.

Test-driven development helps programmers clarify their ideas of what a code segment is actually supposed to do. To write a test, you need to understand what is intended, as this understanding makes it easier to write the required code.

Benefits of test-driven development are:

1. *Code coverage* In principle, every code segment that you write should have at least one associated test. Therefore, you can be confident that all of the code in the system has actually been executed. Code is tested as it is written, so defects are discovered early in the development process.

2. *Regression testing* A test suite is developed incrementally as a program is developed. You can always run regression tests to check that changes to the program have not introduced new bugs.

3. *Simplified debugging* When a test fails, it should be obvious where the problem lies. The newly written code needs to be checked and modified. You do not need to use debugging tools to locate the problem. Reports of the use of TDD suggest that it is hardly ever necessary to use an automated debugger in test-driven development (Martin 2007).

4. *System documentation* The tests themselves act as a form of documentation that describe what the code should be doing. Reading the tests can make it easier to understand the code.

1. **What is the difference between Release testing & User testing?**

*Release testing* is the process of testing a particular release of a system that is intended for use outside of the development team. Normally, the system release is for customers and users.The primary goal of the release testing process is to convince the supplier of the system that it is good enough for use. If so, it can be released as a product or delivered to the customer. Release testing, therefore, has to show that the system delivers its specified functionality, performance, and dependability, and that it does not fail during normal use.

*User or customer testing* is a stage in the testing process in which users or customers provide input and advice on system testing. User testing is essential, even when comprehensive system and release testing have been carried out. Influences from the user’s working environment can have a major effect on the reliability, performance, usability, and robustness of a system.

Release testing refers to the testing of software systems and applications by developers, before they are released to end-users. The focus is on ensuring the functionality and performance of the software meets the specifications and requirements. It includes activities such as unit testing, integration testing, and system testing.

User testing, on the other hand, refers to testing the software with real end-users to determine how well it meets their needs and expectations. The focus is on evaluating the user experience, user interface, and overall usability of the software. It typically involves tasks such as user acceptance testing, beta testing, and field testing.

In summary, release testing focuses on the technical aspects of software, while user testing focuses on the user's experience with the software.

1. **Which is not included Requirements engineering processes**a) Requirements Elicitation   
   b) Requirements Storytelling

c) Requirements Validation  
d) Requirements Specification

Answer: *B) Requirements Storytelling*

**14. What is the Requirement Elicitation?**

This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and procurers, task analysis, and so on. This may involve the development of one or more system models and prototypes. These help you understand the system to be specified.

The aims of the requirements elicitation process are to understand the work that stakeholders do and how they might use a new system to help support that work. During requirements elicitation, software engineers work with stakeholders to find out about the application domain, work activities, the services and system features that stakeholders want, the required performance of the system, hardware constraints, and so on.

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

Many OSS vendors make most of their revenue from support and service. Customers who use open-source software often need help setting it up, configuring it for their specific needs, and troubleshooting any problems. As a result, these companies charge for access to customer support and services.

Open source enables technology agility, typically offering multiple ways to solve problems. Open source helps keep your IT organization from getting blocked because a particular capability isn't available from a vendor. Instead of waiting for the vendor to deliver that capability, you can create it yourself.

**16. What is not a widely used UML diagram?**   
 a) Activity Diagram  
 b) Use Case Diagram

c) Class Diagram  
 d) Component Diagram

Answer: *D)Component Diagram*