What is Testing?

- Definition:

- The process of identifying the bugs/errors/defects.
- The process of checking whether the project/product is satisfying the requirements of the client(end user) or not.
- Testing is the process of examining or trying something to check its quality, performance, or functionality.
- Testing refers to evaluating a software application to ensure it behaves as expected, identifying bugs, and verifying its functionality, performance, and security.
- Testing is the process of verifying that a product, device, or machine meets design specifications and safety standards before it is used or sold.

What are the Objectives of Testing?

- 1. **Ensuring Quality**
- **Objective**: To ensure that the software or product meets the desired quality standards and functions as intended.
- **Goal**: To deliver a product that is reliable and performs well under various conditions.
- 2. **Identifying Bugs and Defects**
- **Objective**: To find and fix errors, bugs, or defects before the product is released to users.
- **Goal**: To detect issues early in development, reducing costly fixes later.
- 3. **Verifying Functionality**
- **Objective**: To check that the product performs the required tasks correctly, as outlined in the specifications.
- **Goal**: To ensure that each feature of the software works as expected and fulfills user requirements.

- 4. **Ensuring Compatibility**
- **Objective**: To verify that the product works across different devices, operating systems, browsers, or environments.
- **Goal**: To ensure users have a consistent experience, no matter how they access the product.
- 5. **Ensuring Security**
- **Objective**: To identify vulnerabilities or weaknesses that could be exploited by attackers.
- **Goal**: To protect the product and its users from security threats, data breaches, or unauthorized access.
- 6. **Performance Testing**
- **Objective**: To assess how the software behaves under various conditions, such as high user load or large amounts of data.
- **Goal**: To ensure that the system performs efficiently, even under stress or heavy traffic.

- 7. **Validating User Experience (UX)**
- **Objective**: To ensure the product provides a positive, intuitive, and user-friendly experience.
- **Goal**: To create a product that users enjoy interacting with and can easily navigate.
- 8. **Compliance with Standards and Regulations**
- **Objective**: To ensure the product complies with legal, regulatory, or industry standards.
- **Goal**: To avoid legal or financial penalties by meeting necessary requirements.

Why do we need Testing?

- **Ensures Software Quality**

Testing helps confirm that the software meets quality standards and functions correctly under all conditions.

Identifies Bugs and Defects

Bugs and defects can often go unnoticed during development.

Testing helps find these issues before the product reaches the user.

Verifies Requirements Are Met

Testing ensures that the software meets the requirements and specifications set by the stakeholders.

- **Improves Performance**
- Testing evaluates the performance of the software under different scenarios, including stress, load, and scalability testing.
- **Enhances Security**
- Security vulnerabilities are often overlooked in development, and testing can help identify potential security risks.
- **Minimizes Risks**

Testing allows you to identify potential risks in the software early, including security vulnerabilities, performance bottlenecks, or compatibility issues.

- **Reduces Maintenance Costs**
- Fixing issues early in the development cycle is much less expensive than fixing them after the product is deployed.
- **Ensures User Satisfaction**
- Testing helps identify usability issues, ensuring that the product is user-friendly and meets user expectations.
- **Facilitates Updates and Future Development**
- Regular testing ensures that new features or updates to the software don’t break existing functionality.

What is Software Testing?

- Software testing is a part of software development process.
- Software testing is an activity to detect and identify the defects in the software.
- The objective of testing is to release quality product to the client.
- Software testing is the process of checking a software application or system to make sure it works correctly and doesn't have any errors or bugs.
- It's like giving the software a "trial run" to see if it performs as expected, handles different situations properly, and provides a good user experience.

What is the need of software Testing?

- In order to deliver quality software/application/project/product.
- To satisfy the requirements of the user.
- To make the developed software more reliable.
- In order to avoid the negative feedback from the clients.
- 1.Find and Fix Bugs, 2.Ensure Software Quality, 3.Verify Requirements are Met, 4.Improve Security, 5.Ensure Compatibility, 6.Boost Performance, 7.Increase User Satisfaction, 8.Minimize Risks, 9.Cost-Effective.

What is software quality?

- **Quality:**
- It is defined as justification of all the requirements of a customer in an application.
- **Quality Software:**
- Quality software refers to a software product that meets the desired standards and requirements, providing a reliable, efficient, and positive user experience.
- Bug free software, satisfy the requirements of clients, reduce maintenance cost, if we develop and deliver the software within the budget, delivered on time

What is project and product?

- **Project:**
- If the software is developed based on the requirement of specific customer then it is called as "Project".
- A project is a temporary effort with a specific goal or outcome. It has a start and end date, and once the goal is achieved, the project is completed. It could be something like building a house, creating a new software application, or organizing an event.
- **Example**: Building a new website for a company.
- **Product:**
- If the software is developed based on the requirement of multiple customers in the market then it is called as "Product".

- **Product** (in simple terms):
- A **product** is something that is created and sold or used over a long period of time. It's a finished item or service that can be continually used, sold, or improved. Products are often developed, improved, and maintained over time.
- **Example**: A smartphone, a software app, or a car.
- Key Differences:

Aspect	project	product
Definition	A temporary effort with a specific goal.	A finished item or service for long-term use.
Duration	Has a defined start and end date.	Exists continuously, evolving over time.
Purpose	Aims to achieve a specific outcome or deliverable.	Created to be used, sold, or maintained long-term.
Focus	Focuses on completing tasks and meeting deadlines.	Focuses on providing value to users and maintaining quality.

What is an error, bug/defect, failure?

- 1. **Error**:
- **What it is**: A **mistake** made by a developer while writing the code.
- **Example**: Forgetting to add a condition or typing the wrong variable name.
- **Simple Explanation**: It's a human mistake in the code that causes problems.
- 2. **Bug/Defect**:
- **What it is**: A **problem** in the software caused by an error in the code that prevents the software from working as expected.
- **Example**: Clicking a button that doesn’t do anything, or a page not loading properly.
- **Simple Explanation**: It's the result of an error that creates a problem in the software.
- 3. **Failure**:
- **What it is**: When the software **doesn't work as intended** or stops working altogether.
- **Example**: The entire app crashes when you try to log in.
- **Simple Explanation**: It's when a bug causes the software to fail in performing its task or function.

Why the software has bugs normally?

- 1. **Human Mistakes**
- **Why?** Developers are people, and sometimes they make mistakes when writing code.
- **Example**: Typing an incorrect command or missing a small detail.
- 2. **Complexity of Software**
- **Why?** Software can be very complicated, with lots of features and connections. The more complex the software, the more opportunities there are for bugs.
- **Example**: A feature that depends on many different parts of the software might break if one part is changed.

- **Changing Requirements**
- **Why?** Sometimes, the requirements or goals of a project change while the software is being built. These changes can introduce bugs if not carefully managed.
- **Example**: A new feature is added, and it causes problems with existing features.
- 4. **Unpredictable Interactions**
- **Why?** Different parts of the software or different systems it interacts with may not work together as expected.
- **Example**: Two pieces of code designed by different developers might conflict, causing a bug.

5.Time Constraints

Why? Developers are often under pressure to finish software quickly, and rushing can lead to mistakes or overlooking issues.

Example: Missing small errors because of tight deadlines.

6. **Poor Testing**

Why? Sometimes, software isn't tested thoroughly enough, so bugs aren't caught before release.

Example: Not testing a feature on all devices, leading to bugs that appear only on certain ones.

1. Miscommunication, 8. s/w complexity, 9. lack of programming skills, 10. Lack of skilled testers, 11. Lack of collaboration.

Software development life cycle?

1. Requirement analysis
2 Design
3. development
4. testing
5 maintenance

1.Requirement Gathering and Analysis

- This is the first step where the project team talks to the clients or users to understand what they need from the software.
- Business analyst and project manager will collect the requirement from the clients.

2.Design

Once the requirements are clear, the next step is to plan how the software will work. This includes the layout, architecture, and structure of the system.

It's like drawing a blueprint of the software, deciding how the different parts will look and work together.

- **3.Development (Implementation)**
- This is the phase where the actual coding happens. Developers write the code based on the design to build the software.

- **4.Testing**
- After the software is built, it is tested to find and fix any bugs or errors. The goal is to make sure everything works as it should.
- It's like checking if everything in the product works correctly before you give it to the customer. You test it for problems and fix them.

- 5.**Maintenance**

. After the software is released, it needs regular updates, bug fixes, and improvements to keep it running smoothly.

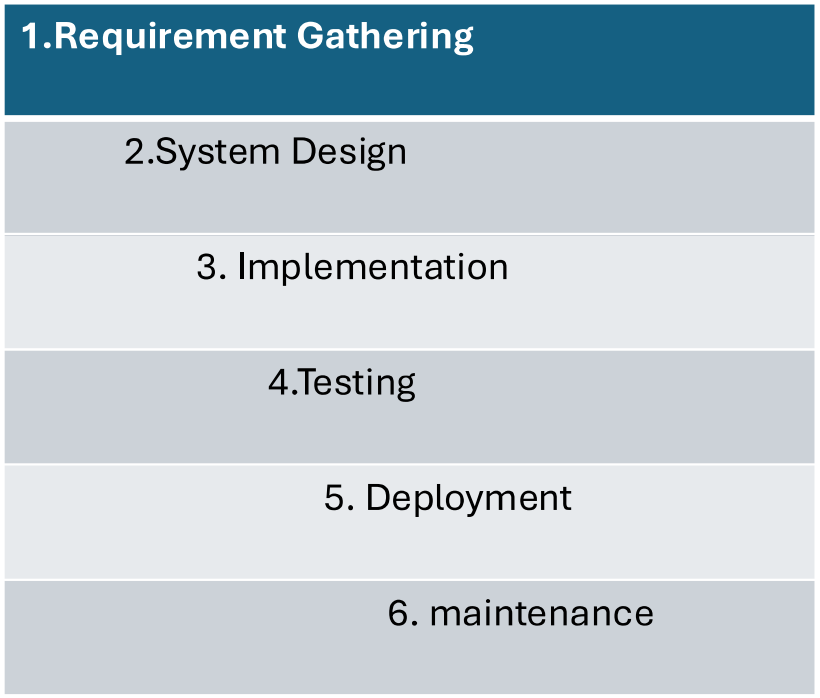
SDLC models?

1. Waterfall model
2. Agile model
3. V-model
4. Iterative model
5. Spiral model
6. Prototype model
7. Incremental model

Waterfall model?

- It is a linear, sequential approach where each phase must be completed before the next one begins.
- **Characteristics**: Simple, structured, and easy to understand, but not flexible.
- **Use case**: Best for projects with well-defined requirements that won’t change.

Flowchart of waterfall model?



Advantages and disadvantages of waterfall model

- Advantages of waterfall model:

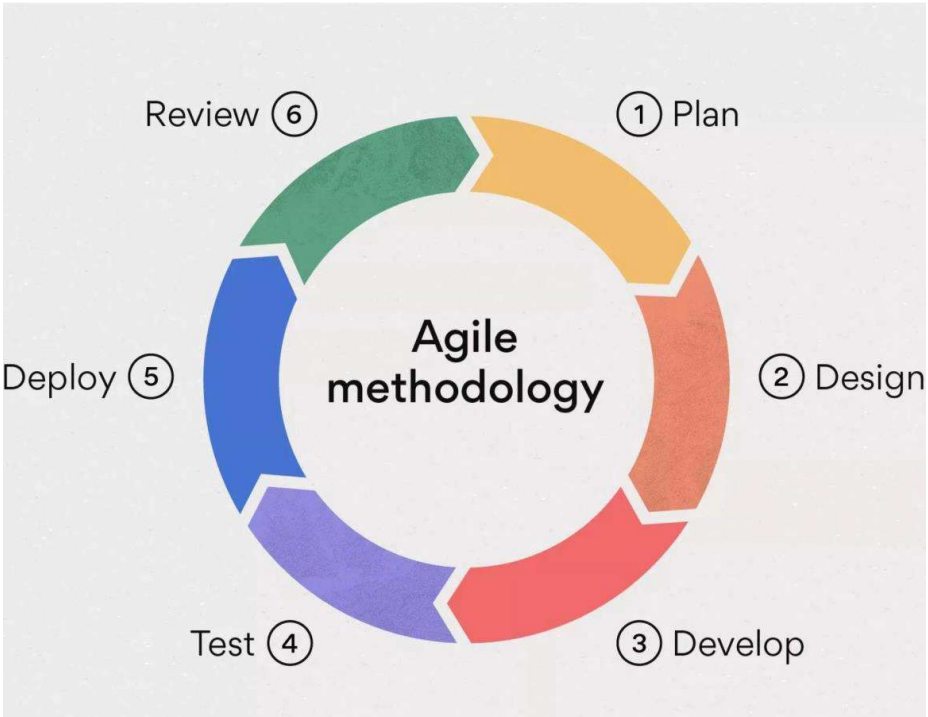
- Simple and easy to understand , easy to manage
- Well – structured documentation
- Suitable for small projects with clear requirements.
- Each phase has clear deliverables and milestones
- Well defined requirements

Disadvantages of waterfall model:

- **Inflexibility**:
- Once a phase is completed, it’s difficult to go back and make changes. If the requirements change, it can be costly and time-consuming.
- **Late Testing**:
- Testing happens after the development phase, meaning bugs may be discovered too late in the process.
- **Assumes All Requirements Are Known Upfront**:
- It doesn't handle changes well, so if new requirements come up during the project, they can’t be easily incorporated.
- **Not Ideal for Complex Projects**:
- Waterfall works well for smaller, well-defined projects, but for larger, more complex ones, it’s hard to adapt to changing needs.
- **Risk of Project Failure**:
- If errors or misunderstandings occur in the early stages (requirements or design), they can have a big impact on later stages, leading to project failure

Agile model?

- **What it is**: A flexible, iterative approach where software is developed in small, manageable chunks (called iterations or sprints).
- **Characteristics**: Focuses on customer collaboration, flexibility, and continuous improvement.
- **Use case**: Best for projects with changing requirements and a focus on delivering working software quickly.


Agile methodology
1 Plan
2 Design
3 Develop
4 Test
5 Deploy
6 Review

- Advantages and disadvantages of agile model:
- Advantages:
- **Flexibility to Changes**:
- Agile allows changes even after the project has started. If the requirements change or new ideas come up, they can be added in future iterations.
- **Frequent Delivery of Working Software**:
- You get a working product at the end of each sprint, which means you can see progress quickly and make adjustments as needed.
- **Customer Feedback**:
- Regular feedback from customers or stakeholders helps ensure the software meets their needs and expectations.
- **Faster Time to Market**:
- Because work is broken into small, manageable chunks (sprints), features can be delivered faster, making the product available sooner.
- **Improved Quality**:
- Regular testing and updates in each iteration help identify and fix problems early, improving overall software quality.

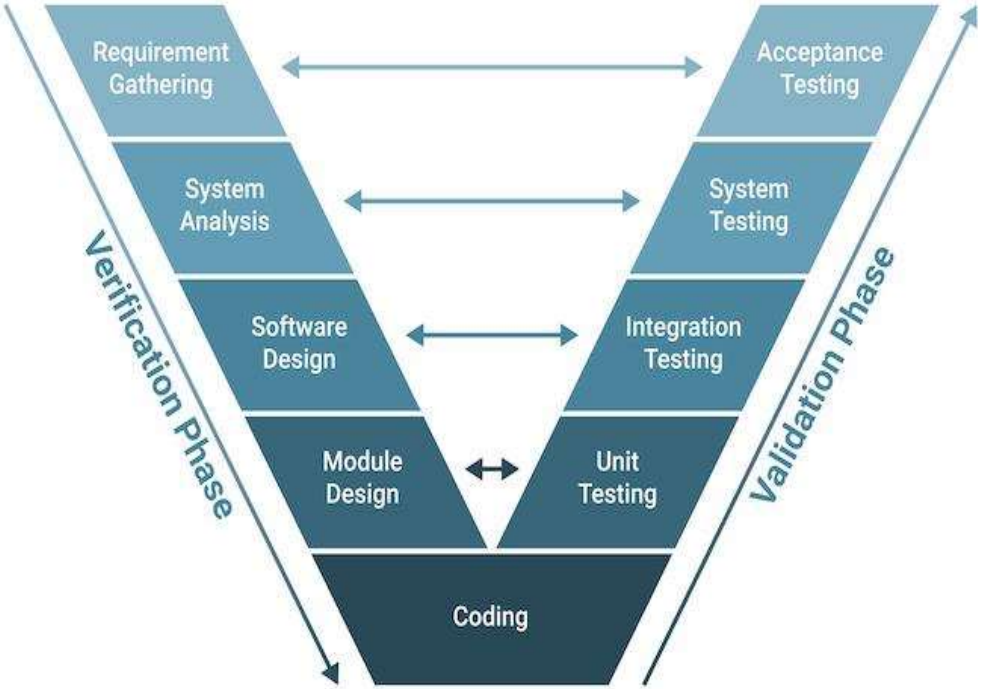
Disadvantages of agile model:

- **Can Be Unpredictable**:
- Because requirements can change and evolve, it can be hard to predict the exact outcome or timeline of the project.
- **Requires Continuous Involvement**:
- Agile needs frequent collaboration with stakeholders and a committed development team, which can be time-consuming.
- **Not Ideal for Large Projects**:
- While Agile works well for smaller projects, it can become difficult to manage large, complex projects with many teams.
- **Lack of Documentation**:
- Since Agile focuses more on working software and less on detailed documentation, it might lead to missing documentation in the long run.

V-Model (Verification and Validation):

- **What it is**: An extension of the Waterfall model that emphasizes testing at each phase. Each development stage has a corresponding testing phase.
- **Characteristics**: Like Waterfall but with parallel testing activities.
- **Use case**: Suitable for projects where high reliability is critical.

V-model diagram:



Advantages of v model:

- **Early Detection of Defects**:
- Since testing is planned and carried out alongside the development process, defects are caught early, making it easier to fix them before they become bigger problems.
- **Well-Defined Stages**:
- The V-Model has clear and structured phases, making it easy to understand and manage the development and testing process.
- **Easy to Manage**:
- Due to its structured approach, it is easy to track progress, manage tasks, and monitor quality at each stage of the project.
- **Testing at Each Stage**:
- Each development phase has a corresponding testing phase (like unit testing, integration testing, system testing), ensuring that quality is maintained throughout the process.
- **Better Documentation**:
- The V-Model ensures detailed documentation for both development and testing phases, which is helpful for future maintenance and audits.

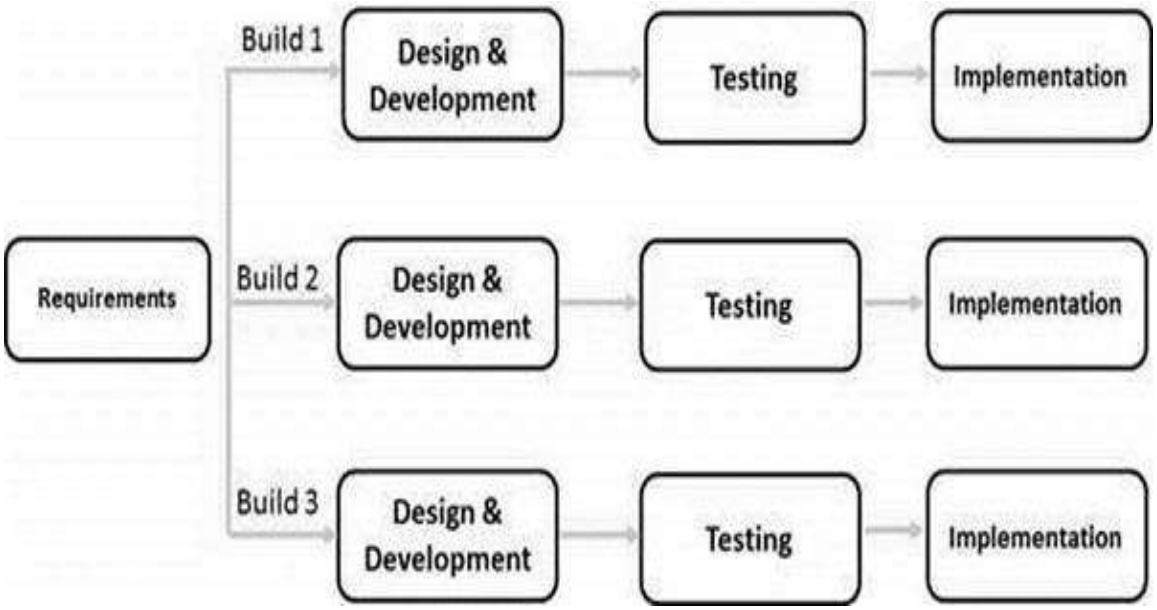
Disadvantages of v- model:

- **Inflexibility**:
- Once the development phase begins, it's difficult to go back and make changes. The V-Model doesn't handle changes easily, which can be a problem if requirements evolve during development.
- **High Cost for Changes**:
- Since each stage is completed before moving to the next, making changes late in the process (like after testing) can be expensive and time-consuming.
- **Not Suitable for Complex Projects**:
- The V-Model works best for projects with well-defined requirements. It may not be the best choice for large or complex projects that require frequent adjustments.

Iterative Model?

- **What it is**: Similar to Agile, but the focus is on developing software through repeated cycles (iterations) where each cycle builds on the previous one.
- **Characteristics**: Allows for changes based on feedback after each iteration.
- **Use case**: Best for projects that can be developed and improved over time with feedback.

Iterative model diagram:



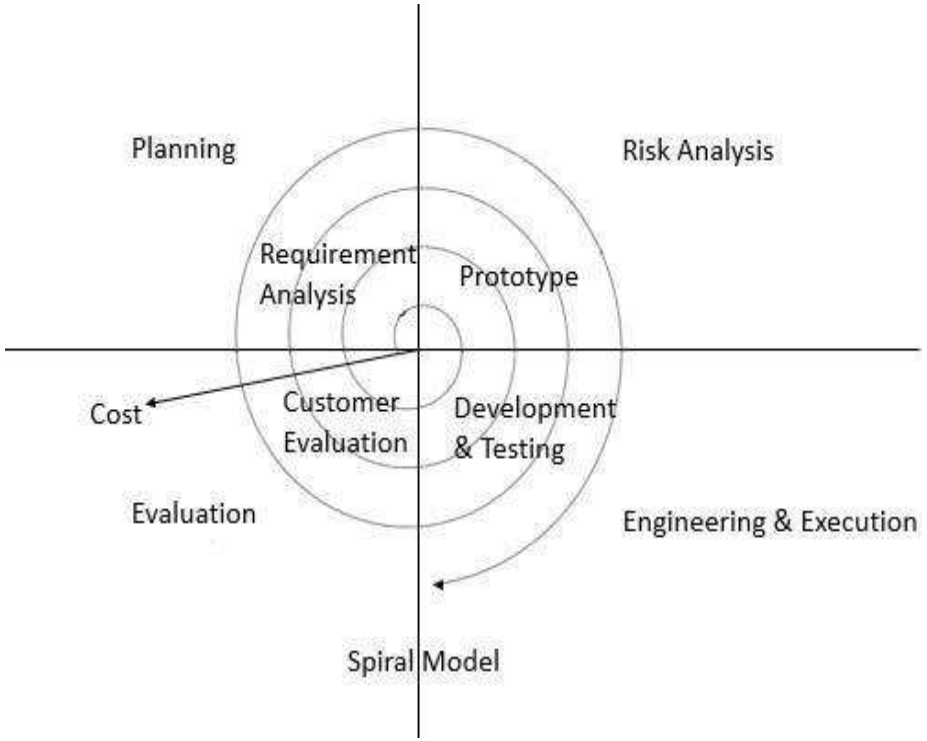
Advantages and disadvantages of iterative model

- **Advantages:**
- Flexibility and Adaptability
- Early Delivery of Working Software
- Frequent Feedback
- Risk Management
- Improved Quality
- **Disadvantages:**
- Can Be Time-Consuming
- Requires Constant Feedback
- Complex to Manage
- Unclear End Product

Spiral model?

- **What it is**: Combines elements of both iterative development and the Waterfall model. It focuses on risk analysis and allows for repeated refinement.
- **Characteristics**: Risk-driven and allows for frequent revisiting of previous stages.
- **Use case**: Ideal for large, complex, and high-risk projects.
- or
- The **Spiral Model** is a software development process that combines elements of both the **Waterfall Model** and the **Iterative Model**. It is designed to manage risks and handle large, complex projects effectively.

Spiral model:



Advantages and disadvantages of spiral model:

- **Advantages:**
- **Effective Risk Management**:
  - The Spiral Model focuses on identifying and managing risks in each cycle, reducing the chances of serious problems later on.
- **Flexible to Changes**:
  - Since the model works in repeated cycles, it's easier to make changes based on feedback or new requirements that come up during development.
- **Frequent Feedback**:
  - Regular feedback from stakeholders (like customers or users) helps ensure that the project is on the right track and meets the needs of everyone involved.
- **Good for Large Projects**:
  - The Spiral Model is ideal for complex and large projects because it breaks the work into smaller, manageable parts and allows for detailed planning.
- **Better Quality**:
  - As the project is developed in stages and reviewed frequently, the quality of the software is likely to improve with each cycle.

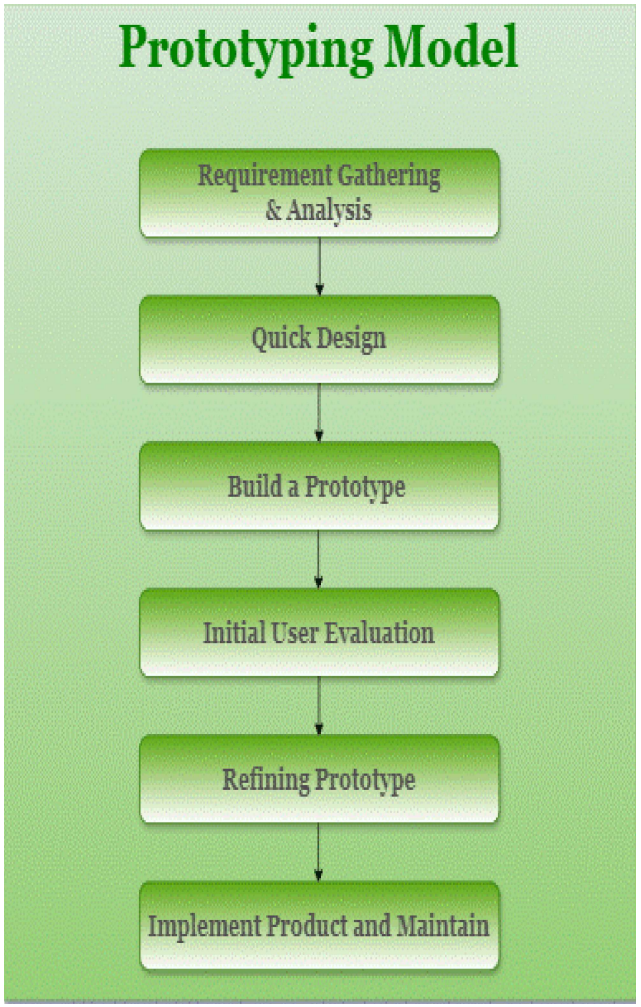
Disadvantages of spiral model:

- **Can Be Expensive**:
  - The repeated cycles, detailed planning, and constant risk analysis make the Spiral Model more costly and time-consuming than other models.
- **Complex to Manage**:
  - The process of planning, developing, and evaluating in multiple cycles can be hard to manage, especially for large teams or projects.
- **Requires Expertise**:
  - The model requires experienced project managers and developers because of its complexity, especially when identifying and managing risks.
- **Time-Consuming**:
  - Since the project goes through several cycles, it may take more time compared to other development models.
- **Not Ideal for Small Projects**:
  - The Spiral Model is more suited for large, complex projects. For small projects, it can be too heavy and unnecessary.

Prototype model:

- **What it is**: Involves building a prototype (an early version of the software) and improving it through feedback from users until the final product is ready.
- **Characteristics**: Quick iterations and early feedback from users.
- **Use case**: Best for projects where user feedback is needed before the final product is developed.
- Or
- The **Prototype Model** is a software development approach where a **working prototype** (an early version of the software) is built quickly to help understand and clarify the requirements of the system.

Prototype model diagram:



Advantages and disadvantages of prototyping model:

- **Advantages:**
- **Better Understanding of Requirements**:
  - By seeing a working version of the software early, users can give clear feedback on what they need, helping to define better requirements.
- **User Involvement**:
  - Users are actively involved throughout the development process, which ensures the final product meets their expectations.
- **Reduces Risk of Misunderstanding**:
  - The prototype helps clarify unclear or vague requirements early, reducing the chances of misunderstandings.
- **Faster Feedback**:
  - The quick development of a prototype means you can get feedback sooner and make adjustments early in the process.
- **Improved Quality**:
  - Continuous testing and feedback help improve the quality of the software as it evolves.

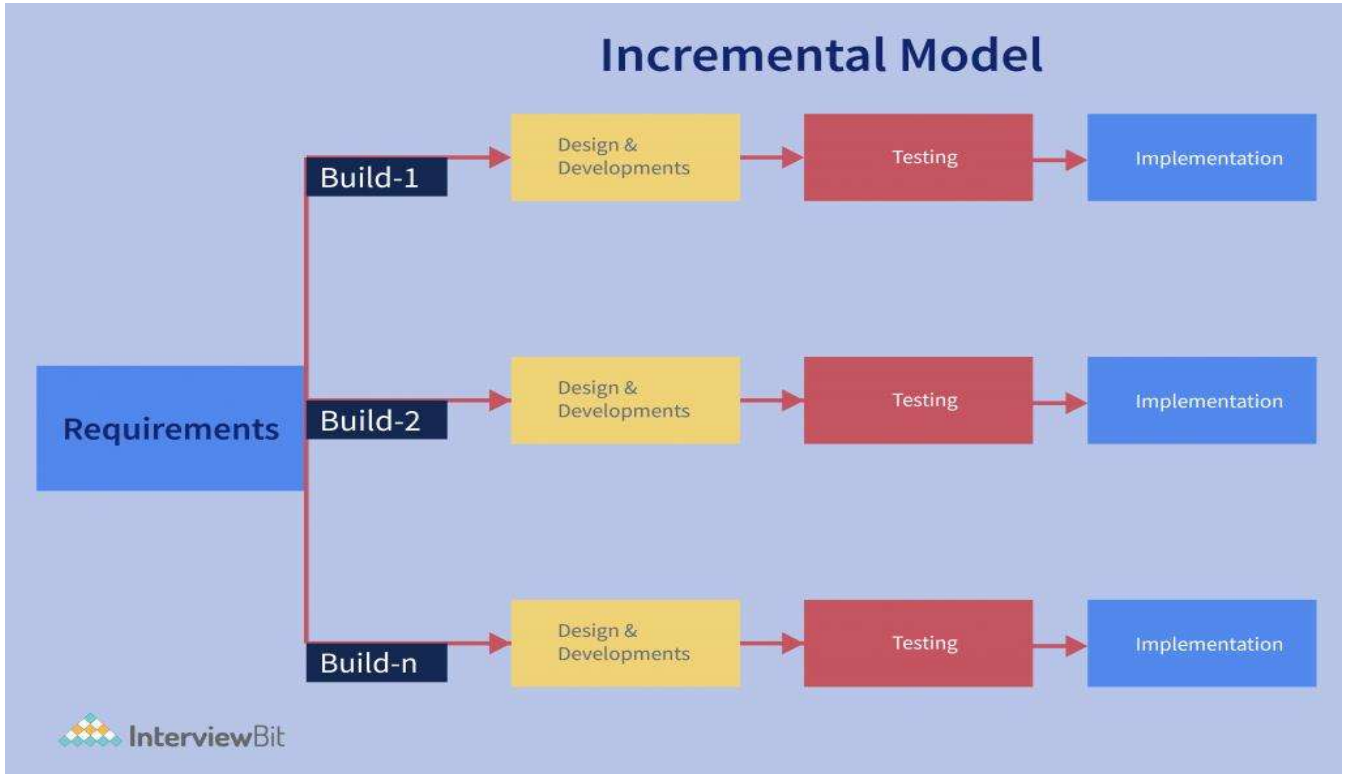
Disadvantages of prototype model:

- **Can Be Time-Consuming**:
  - Building and refining multiple prototypes can take extra time, especially when the software is complex.
- **Incomplete Final Product**:
  - Prototypes are often not fully developed, and there may be features missing that could affect the final product's usability or performance.
- **Misleading Expectations**:
  - Since prototypes are not the final product, users might have unrealistic expectations of what the final software will look like or be able to do.
- **Can Be Expensive**:
  - Developing several prototypes and refining them based on feedback can lead to higher development costs.
- **Not Ideal for Large Projects**:
  - For large and complex systems, the prototype model can become difficult to manage and might not be the best choice.

Incremental model?

- **What it is**: The software is developed in small, manageable pieces (increments), each delivering part of the functionality.
- **Characteristics**: New functionality is added in increments, and the software grows gradually.
- **Use case**: Good for projects that can be broken down into smaller parts and delivered progressively.
- or
- The **Incremental Model** is a software development approach where the product is developed in small parts or **increments**. Each increment adds a piece of functionality to the software, and after each increment, a working version of the software is released.

Incremental model diagram:



Incremental model advantages and disadvantages:

- **Advantages of Incremental Model**:
- **Faster Initial Delivery**:
  - The first version of the software is delivered quickly, even if it's just a basic version with limited features, allowing users to start using it early.
- **Flexibility to Change**:
  - As development progresses in increments, it’s easier to make changes or add new features based on user feedback or evolving needs.
- **Easier to Manage**:
  - Smaller, incremental development is easier to manage and track progress, making it more efficient compared to building the whole software at once.
- **Improved Risk Management**:
  - Since you’re releasing parts of the software early, you can identify risks and fix issues in the earlier increments before they become bigger problems.
- **User Feedback Early**:
  - Users get to see and test the product early on, giving valuable feedback that helps shape the future increments and overall product.

Disadvantages of incremental model:

- **Disadvantages of Incremental Model**:
- **Requires Good Planning**:
  - The software must be carefully planned, so that each increment builds upon the last in a way that doesn’t create confusion or conflicts.
- **Might Have Incomplete Functionality**:
  - Since each increment only delivers part of the functionality, users may get an incomplete product that doesn’t work fully until later increments are released.
- **Difficult to Manage Large Projects**:
  - For very large or complex projects, managing multiple increments can become complicated, and coordination across increments can be challenging.

Major differences between waterfall model and agile model:

Aspect	Waterfall model	Agile model
Development Process	Linear and sequential (one phase after another).	Iterative and incremental (work in small cycles).
Flexibility	Difficult to make changes once the process starts.	Highly flexible, allows changes during development.
Feedback	Limited feedback, usually at the end of the project.	Continuous feedback from customers throughout
Requirements	All requirements are defined at the start.	Requirements evolve and can change during the process.
Time to Deliver	Longer time to get a working product (delivered at the end).	Faster delivery with frequent working versions (increments).

Difference between waterfall and agile model:

Documentation	Heavy documentation at every stage	Less focus on documentation, more on working software.
Risk	Risk is identified and addressed late in the process.	Risks are continuously identified and managed early on.
Customer Involvement	Low customer involvement after initial requirements.	High customer involvement during each iteration.
Best for	Projects with clear, unchanging requirements.	Projects where requirements are likely to change or evolve.

What is cloud and cloud computing?

- **Cloud:**
- It is a huge space with all the services to use and rent.
- **Cloud computing:**
- It is an ability or place to store your data
- We can process it and access it from anywhere

What are the top cloud providers?

1.Amazon Web Services (AWS)

Market Share: Approximately **32%** to **34%**.

Overview: AWS is the **largest** cloud service provider, widely adopted across industries for infrastructure services, computing power, and storage solutions.

2. Microsoft Azure

Market Share: Approximately **20%** to **22%**.

Overview: Azure is particularly strong in **enterprise** and **hybrid cloud** solutions, heavily integrated with Microsoft's ecosystem, including Windows Server, SQL Server, and Office.

3.Google Cloud Platform (GCP)

Market Share: Approximately **9%** to **10%**.

Overview: Known for its strong **AI/ML** capabilities and big data tools, GCP is popular among businesses focused on **data analytics** and **machine learning**.

4.Alibaba Cloud

- **Market Share**: Approximately **6%** to **8%**.
- **Overview**: The **leading cloud provider in China** and growing globally, Alibaba Cloud is known for cost-effective solutions, particularly in the **Asia-Pacific** region.

5.IBM Cloud

- **Market Share**: Approximately **4%** to **5%**.
- **Overview**: IBM Cloud is well-regarded for its **enterprise solutions**, particularly in **hybrid cloud** and AI technologies like **Watson**.

6.Oracle Cloud

- **Market Share**: Approximately **2%** to **3%**.
- **Overview**: Oracle Cloud excels in **enterprise database** solutions and provides comprehensive **cloud-based ERP** services.

7.Salesforce

- **Market Share**: Approximately **2%**.
- **Overview**: Focused primarily on **CRM (Customer Relationship Management)**, Salesforce is a major player in **cloud-based customer service and marketing solutions**.

8.DigitalOcean

- **Market Share**: Less than **2%**.
- **Overview**: Known for its **simplicity and developer-friendly cloud services**, Digital Ocean is popular with small businesses and startups.

9.VMware Cloud

- **Market Share**: Less than **2%**.
- **Overview**: VMware Cloud is highly popular in **hybrid cloud** environments and used by organizations with existing VMware-based infrastructure.

10.Tencent Cloud

- **Market Share**: Approximately **2%**.
- **Overview**: Growing rapidly, especially in **China** and **Asia**, Tencent Cloud is focusing on **gaming** and **AI** technologies.

Cloud computing types:(2types)

- 1.service models------ ----------
- 1.Saas(software as a service)
- 2.Iaas(infrastructure as a service)
- 3.paas(platform as a service)
- 2.deployment models---------
- 1.public cloud
- 2.private cloud
- 3.hybrid cloud
- 4.community cloud

Difference between service models(Saas,

Aspect	SaaS(software as a service)	IaaS(infrastructure as a service)	PaaS(Platform as a service)
Definition	Provides ready-to-use software applications over the internet.	Provides virtualized computing resources (servers, storage, etc.) on demand.	Provides a platform and environment for developers to build, deploy, and manage applications.
User Responsibility	End-users only use the software, no management of infrastructure needed.	Users manage applications and data, while the provider manages infrastructure.	Users manage applications, while the provider handles platform maintenance (servers, OS).
Example	Google Workspace (Gmail, Docs), Dropbox, Microsoft 365.	Amazon EC2, Google Compute Engine, Microsoft Azure VMs.	Google App Engine, Microsoft Azure App Services, Heroku

Aspect	Saas	IaaS	PaaS
Control	Minimal control over the software (only configuration and usage).	Full control over the infrastructure (e.g., VMs, storage).	Control over applications, but the platform (OS, infrastructure) is managed by the provider.
Target Users	End users and businesses who need a ready-to-use application.	Businesses needing scalable infrastructure without owning hardware.	Developers who want to focus on creating apps without worrying about underlying infrastructure.
Use Case	Accessing web-based software without worrying about maintenance or installation.	Hosting websites, running virtual machines, or storing large amounts of data.	Developing, testing, and deploying web applications quickly.
example	**Zomato App** is a **SaaS** product for customers. Users simply download and use the app to search for restaurants, read	Zomato could use **IaaS** services like **Amazon EC2** or **Google Compute Engine** to	Zomato might use **PaaS** to develop and host its backend services for restaurant management,

Deployment models?

Aspect	Public cloud	Private cloud	Hybrid cloud	Community cloud
Ownership	Owned and operated by third-party cloud providers (e.g., AWS, Azure).	Owned and operated by the organization or a third-party just for that organization.	A mix of both public and private clouds, managed separately but connected.	Shared by multiple organizations with common concerns.
Access	Available to the public and multiple organizations.	Accessible only to a single organization.	Both public and private clouds are accessible to the organization.	Available to a specific group of organizations (e.g., healthcare, government).
Security	Security managed by the cloud provider, may be less secure compared to private clouds.	High security as it's dedicated to a single organization	Combines the security of private cloud with the scalability of public cloud.	Security tailored to the group but less than private cloud.

Aspect	Public cloud	Private cloud	Hybrid cloud	Community cloud
Cost	Pay-as-you-go, generally cheaper because the resources are shared.	Expensive to set up and maintain, as the infrastructure is dedicated.	Cost varies depending on the usage of both public and private clouds.	Moderate cost, shared resources, and shared maintenance.
Examples	Amazon Web Services (AWS), Microsoft Azure, Google Cloud.	Private cloud solutions like VMware, OpenStack, or hosted private clouds.	A combination of AWS and on-premises infrastructure, or Azure + private data center.	A government cloud shared by several federal agencies.