**MODULE 1**

1. What is a software?

Software is a set of items or objects that form a “configuration” that includes

• programs • data • documents

1. What r diff. Types of process models?

A process model in software engineering is an abstraction of the software development process, which presents a description of the process from a specific perspective

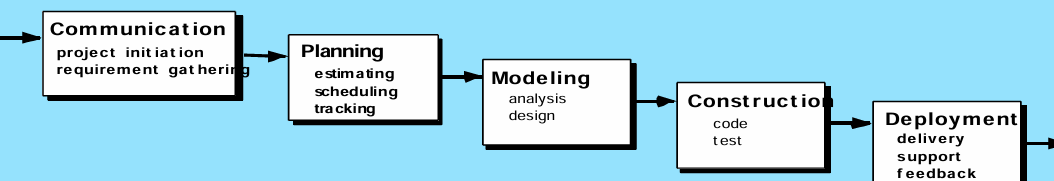
Prescriptive Process Models: The Waterfall Model, V Model.

Incremental Process Model: Incremental Model

Evolutionary Process Models: Prototyping Paradigm Spiral Model

Concurrent Process Models: Concurrent Process Model

The Waterfall Model



Advantages

1. simple n easy to understand
2. Small proj k liye acha hai
3. Complete in 1 go

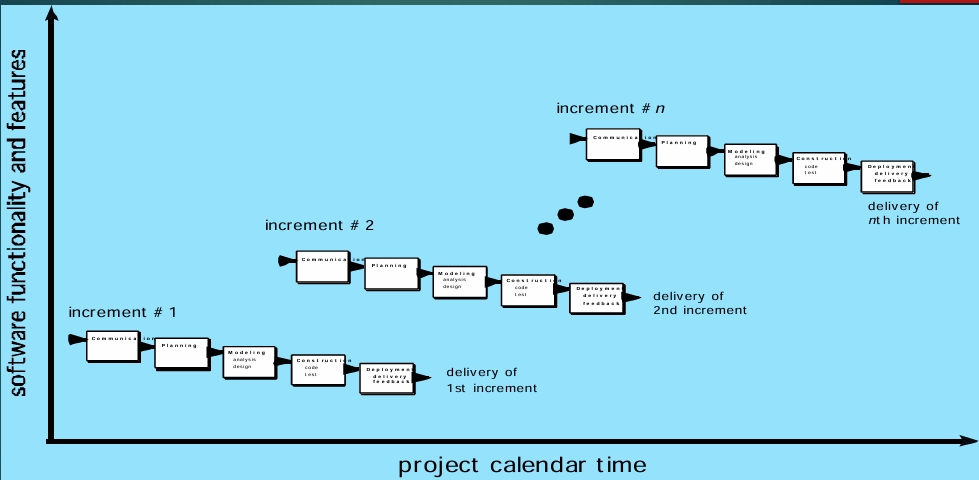
Disadv

1. Not for complex obj o proj
2. Diff. To go back n change

When to use

1. Requirements stable n known ho

The Incremental Model



- custom waterfall model bol saktey

- breaking the prod. Into increments -> delivering each increment to the customer

- a software development project with a large and complex system might use the Incremental model.

Advantages

1. Release a prod. Early in the market
2. Ez to resolve bugs n test
3. Customers can repond to each build -> give feedback

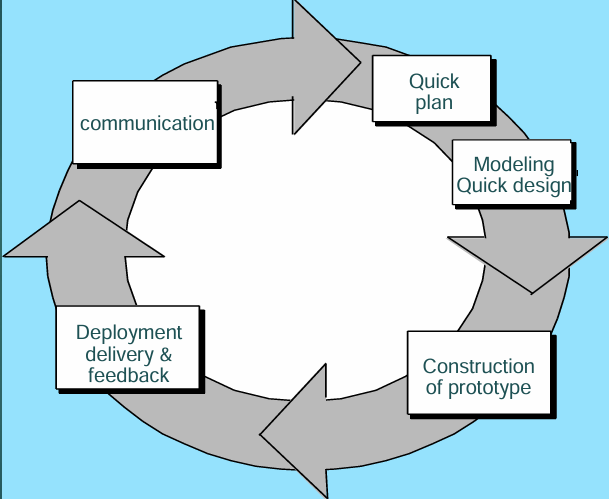
Disadv

1. Cost > than waterfall
2. Needs good planning

When to use

1. When major req. Are known (small can be added later)
2. prod. is req. In market asap
3. Using new tech/have high risk feats/resource not avail

Prototyping Paradigm



- software development project with a new and innovative product might use the Prototyping paradigm.

Advantages

1. Quick user feeback
2. Missing functionalities identified asap
3. Error detected asap
4. Minimal working product is dev

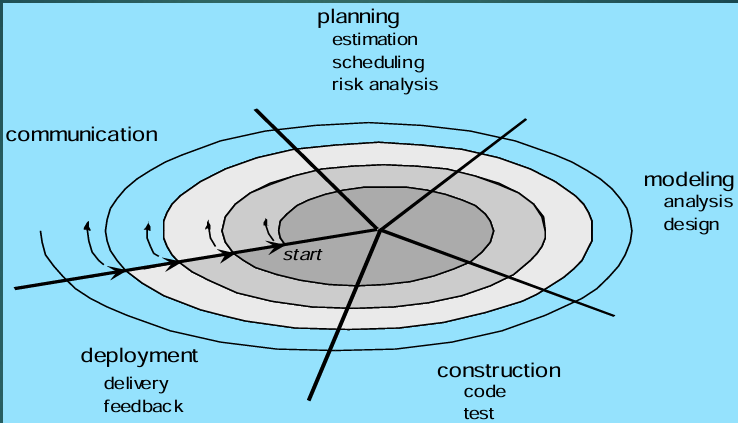
Disadv

1. Scope may inc unexpectedly
2. High research n analysis is reqd
3. Incomplete application -> no use

When to use

1. when the requirements are unclear or are likely to change.
2. Lot of interaction w end users

Spiral Model



- It involves a series of iterations, each with its own set of requirements and testing.

- a software development project with a large and complex system might use the Spiral model.

Advantages

1. Complex n large proj.
2. High risk analysis
3. Released in the market early

Disadv

1. Cost high
2. Risk analyis -> expert chahiye
3. Not for small proj
4. Diff. To convince stakeholders that spiral model is controllable

When to use

1. For high risk projects
2. When the req. Is not known fully at the beginning
3. New product line
4. User not sure of their needs

3. What is agility?

- refers to the ability to respond and adapt to change quickly and effectively. It is characterized by a flexible and iterative approach to development, where requirements and solutions evolve through collaborative efforts and feedback loops.

- Agile methodologies, such as Scrum, Kanban, and Extreme Programming (XP), promote agility by emphasizing certain principles and practices

4. What is SCRUM?

- framework - used in SE - for managing n completing complex proj

- emphasizes teamwork, accountability, and flexibility

Key Components :

- Product Backlog -> feat to be completed for the projecct

- Sprint -> short cycles (1-4 weeks) to complete a set of tasks

- Daily Scrum -> daily meets to discuss progress, obstacles; plan the day

- Sprint review -> At the end of the sprint - work demonstration - receive feedback from stakeholders

- Sprint retrospective -> reflect on the sprint - identify areas of improv.

Example: Suppose a software development team is building a new mobile app. The Product Backlog includes features like user registration, login, and a search function. The team decides to complete the user registration and login features in the first Sprint.

* During the Daily Scrum, team members discuss their tasks and progress, and the Scrum Master ensures everyone is on track.
* At the end of the Sprint, the team demonstrates the completed user registration and login features to stakeholders.
* The stakeholders provide feedback, and the team identifies areas for improvement, such as improving the search function.

5. Explain Agile Modeling

Advantages

- Agile vs. Waterfall:

- Agile focuses on quick iterations and frequent feedback rather than linear phases.

- crucial in a fast-moving digital landscape where products can become obsolete quickly.

- Frequent Delivery

- Face-to-Face Communication: Agile encourages regular, face-to-face communication with clients or stakeholders.

- Flat Organizational Structure: Agile teams operate with a flat hierarchy, promoting collaboration and empowerment among team members.

Disadvantages

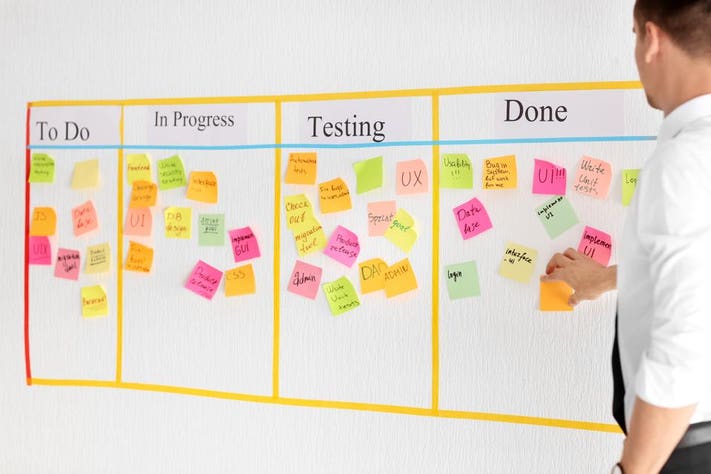
- reduced emphasis on documentation

- maintenance difficulties.

why to use -

- adopted by major companies like Facebook, Amazon, Google, and Adobe.

6. Explain Kanban Model



- Agile proj. Management framework - to VISUALIZE n optimize the workflow

- visualized using a kanban board

- board consists of columns that rep. Diff. Stages of work such as "Backlog," "Ready for Development," "In Progress," "Testing," and "Done."

- work to be done is broken down into smaller, manageable tasks.

- Each task is represented by a card or sticky note which includes task description, priority, assignee, and due date.

- each col. Has a limit to prevent overload & smooth workflow

- encourage continuous delivery as task move across to “Done” column

- stakeholders can see the progress in real-time

**MODULE 2**

1. What is SRS (software requirement specification) ?

- detailed doc comprising of purpose and env for software under dev

- comprises of functional n non-functional requirements, attributes, stc

- blueprint/ roadmap for the software

Functional requirements - defines what a system/component must be qualified to perform

Examples

- email on user registration / auth on user login

- system shut down in cyber attack

Non-fuctional requirements - defines how the system should perform

Examples

- website load time < 3s if users > 10000

- emails received latency < 12 hrs

- request processing < 10 sec

1. What are different requirement models?

* Scenario-Based Models:

- These models describe the system from the user's perspective, focusing on the user's interactions and the system's responses.

- Example: A traveler uses an airline's mobile application to generate a boarding pass. The scenario-based model would describe the steps the traveler takes and the application's responses, such as displaying the boarding pass after submitting the form.

* Class-Based Models:

- These models identify classes, attributes, and relationships within the system, focusing on the structure and organization of the system.

- Example: In the airline application, the traveler and boarding pass are classes. The traveler's first and last name, travel document type, and other details are attributes. The relationship between the traveler and boarding pass classes is that the traveler must enter these details into the application to get the boarding pass.

* Behavioral Models:

- These models describe the behavior of the system, focusing on how it responds to different inputs and events.

- Example: In the airline application, a behavioral model might describe how the system generates a boarding pass after the traveler submits the form. It would detail the steps the system takes, such as validating the traveler's details and generating the boarding pass.

* Flow Models:

- These models depict how data flows through the system, focusing on the transformation of data from input to output.

- Example: In the airline application, a flow model might describe how the system processes the traveler's details, validates them, and generates the boarding pass. It would detail the steps the system takes to transform the input data into the output boarding pass.

1. Why to use FP and not cocomo ?

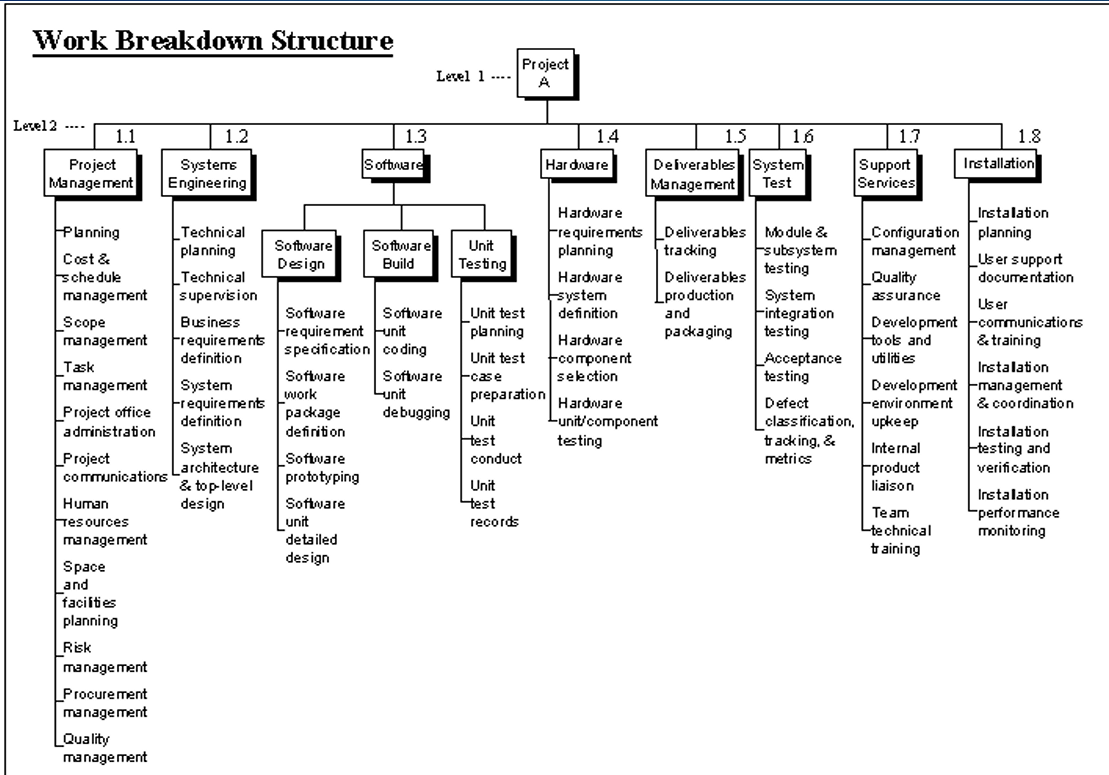
- FP is a method of measuring the functionality of software based on the inputs, outputs, and user interactions. It quantifies the size and complexity of a software system, which can then be used to estimate effort, cost, and duration of a project.

- COCOMO is a model that estimates the effort and cost required to develop software based on different project parameters, such as size, complexity, development team experience, and more. It provides estimation models for different development stages, such as COCOMO I, COCOMO II, and COCOMO III.

- Function Points are a commonly used technique for estimating software size and complexity, and they provide a more abstract and high-level approach to estimation. COCOMO, on the other hand, is a more detailed and comprehensive model that considers various factors and parameters to estimate effort and cost.

**MODULE 4**

1. Explain WBS



Steps Work Breakdown Structure:

Step 1: Identify the major activities of the project.

Step 2: Identify the sub-activities of the major activities.

Step 3: Repeat till undividable, simple, and independent activities are created.

Firstly, the project managers and top level management identifies the main deliverables of the project.

After this important step, these main deliverables are broke down into smaller higher-level tasks and this complete process is done recursively to produce much smaller independent tasks.

It depends on the project manager and team that upto which level of detail they want to break down their project.

Generally the lowest level tasks are the most simplest and independent tasks and takes less than two weeks worth of work.

**MODULE 5**

1. Explain Version Control.

Definition: Version control is a system that manages and tracks changes made to files, code, and project assets over time. It provides a structured way to collaborate, manage code, and maintain a history of changes.

Practical Importance:

1. Branching and Merging: Developers can create branches to work on new features or experiments independently without affecting the main codebase. Merging allows changes from branches to be integrated back into the main codebase.

1. Collaboration: Version control systems enable developers to collaborate efficiently by providing a central repository for sharing code, tracking changes, and resolving conflicts. Multiple developers can work on the same project simultaneously without interfering with each other's work.

3. Quality Assurance: Version control enables developers to track and review changes, ensuring that all modifications are well-documented and tested before integration. - maintain code quality - reduce bugs.

4 Code Management: Version control systems maintain a history of changes, allowing developers to roll back or restore code as needed.

5. Code Reusability: Version control systems facilitate code reuse by enabling developers to track and manage code snippets, libraries, and components that can be reused across multiple projects.

6. Continuous Integration and Deployment

7. Project Management: Version control systems provide a structured way to manage projects by tracking changes, assigning tasks, and monitoring progress. They offer a clear view of the project's history and status.

**MODULE 6**

1. Explain Test-Driven Development (TDD)

Explanation:

Test-Driven Development (TDD) is a software development approach where tests are written before the actual code. It involves a cyclical process of writing a failing test, writing the minimum code to make the test pass, and then refactoring the code. TDD ensures that the code meets the specified requirements and functions correctly, leading to more robust and reliable software.

Use Case: Building a Social Media Sharing Feature

Scenario:

Imagine you are developing a social media platform and need to implement a new feature that allows users to share posts with their friends. Let's apply TDD to this scenario:

1. Write the Test First:

- Start by writing a test case that specifies the expected behavior of the sharing feature. For example, the test could verify that a user can successfully share a post with another user.

2. Run the Test (and Fail):

- Run the test, which should fail initially since the sharing feature has not been implemented yet.

3. Write the Minimum Code to Pass the Test:

- Implement the sharing feature by writing the simplest code that allows the test to pass. This could involve creating a function that sends the post to the specified user.

4. Run the Test (and Pass):

- Re-run the test to ensure it passes, indicating that the sharing feature is working correctly.

5. Refactor the Code:

- Refactor the code to **improve its design, readability, or performance** without changing its behavior. Ensure that the test still passes after refactoring.

6. Repeat the Process:

- Write additional test cases for different sharing scenarios, such as sharing multiple posts, sharing with multiple users, or sharing across different platforms.

- Follow the TDD cycle for each new test case, ensuring that the sharing feature is thoroughly tested and functions as expected.

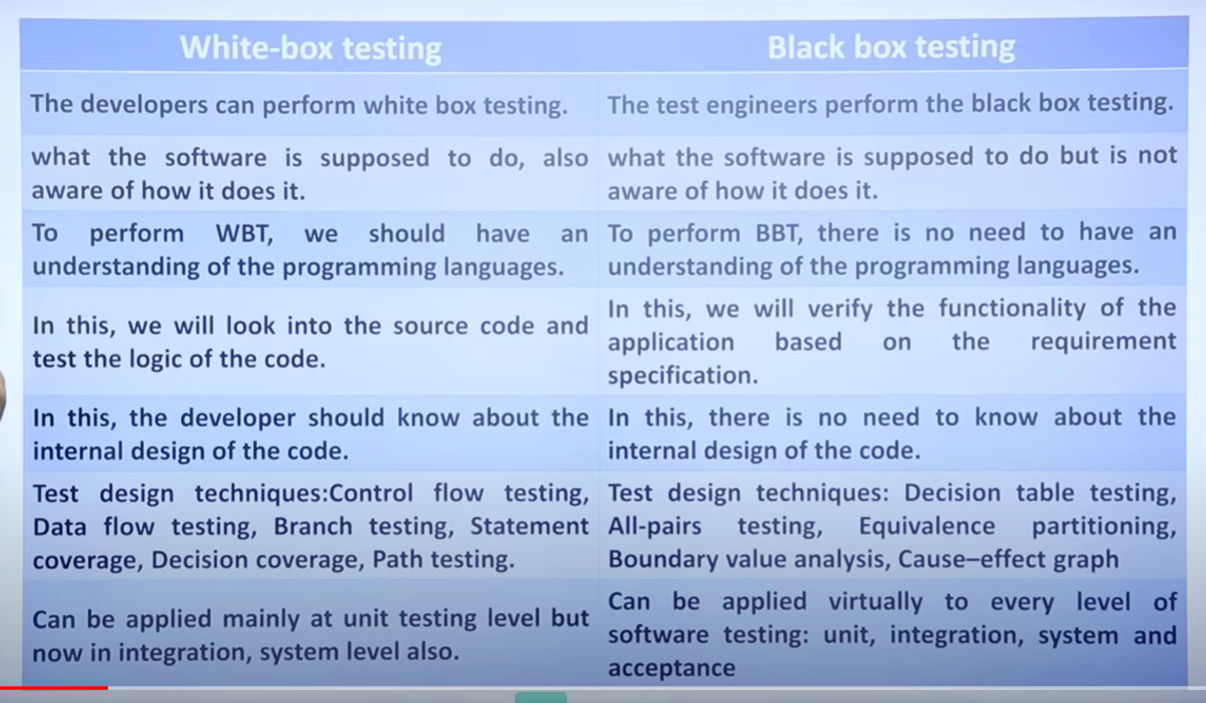
Benefits of TDD in this Use Case:

- Early Detection of Issues: TDD helps catch bugs early in the development process, ensuring that the sharing feature works correctly.

- Improved Design: By writing tests first, TDD encourages a clear understanding of the feature's requirements and leads to better design decisions.

- Confidence in Code: Developers gain confidence in the reliability and functionality of the sharing feature through comprehensive testing.

- Maintainable Code: TDD results in well-documented and maintainable code, making it easier to extend or modify the feature in the future.

1. White box and black box testing  
     
   
2. What are the different types of security testing ?

* Vulnerability Scanning: Involves automated tools to identify security vulnerabilities in a software application or network. It focuses on known risks and provides recommendations for remediation.
* Penetration Testing: A subset of ethical hacking that simulates real-world attacks to locate vulnerabilities in a software application. The goal is to identify potential security threats and remediation measures.
* Application Security Testing (AST): Evaluates the security of a software application by analyzing its source code, configuration, and design. It involves a combination of automated and manual testing techniques to detect and mitigate security risks.
* Web App Security Testing: Focuses on identifying vulnerabilities in web-based applications through a combination of manual and automated testing methods like SQL injection testing, cross-site scripting (XSS) testing, and authentication testing.
* API Testing: Evaluates the security of an application's APIs and the systems they interact with by sending malicious requests to identify potential vulnerabilities. It ensures APIs are secure from attacks and sensitive data is protected.
* Security Auditing: Evaluates the security of a software application or network to identify vulnerabilities and ensure compliance with security standards and best practices. It includes manual methods like code review, vulnerability scanning, and penetration tests.

3. Which has minimum overhead stubs or drivers and  why ?

stubs bcz drivers have to perform complex operations as well as they :

- simulate the behavior of higher-level modules

- handle data flows