SOFTWARE ENGINEERING

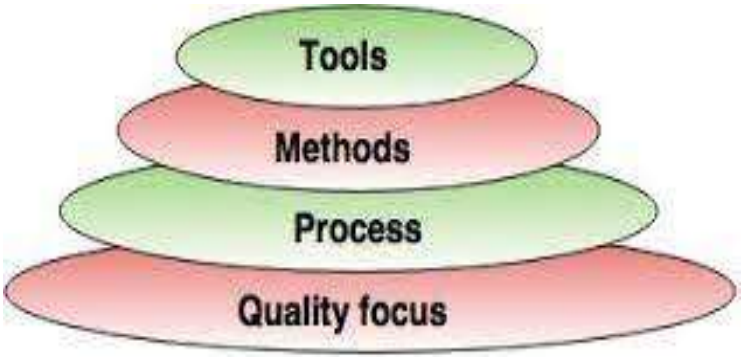
UNIT – 1

TOPIC – 4

SOFTWARE ENGINEERING – A LAYERED TECHNOLOGY, PROCESS FRAMEWORK, GENERIC PROCESS MODEL

I. SOFTWARE ENGINEERING - A LAYERED TECHNOLOGY

Software engineering isn't just about writing code; it involves several layers that work together to produce reliable, high-quality software. These layers provide a structured approach to software development.



Software Engineering – A Layered Technology

Layers in Software Engineering

- **Process Layer:**
  - This layer defines the series of steps that need to be followed to develop software. It acts as a guide for what needs to be done and when.
  - **Example:** A company developing a mobile app may have a process that starts with gathering requirements, followed by design, coding, testing, and finally deployment.
- **Methods Layer:**
  - Methods are the techniques or procedures used during software development. They guide how specific tasks, such as designing the software architecture or testing the code, should be performed.

- **Example:** During the design phase, methods like UML diagrams (Unified Modelling Language) might be used to visually represent the system's components.

- **Tools Layer:**
  - Tools are software applications that help automate tasks or manage the process of software development. They make the work faster and more efficient.
  - **Example:** Version control tools like Git help developers keep track of changes in code, allowing multiple developers to work on the same project without conflicts.
- **Quality Focus (Bedrock):**
  - Quality is the foundation that supports all other layers. Every step, method, and tool used in the software engineering process must focus on ensuring the software meets the required quality standards.
  - **Example:** Before releasing a product, rigorous testing is conducted to ensure it performs well under various conditions, just like ensuring a cake is baked properly before serving.

II. THE SOFTWARE PROCESS FRAMEWORK

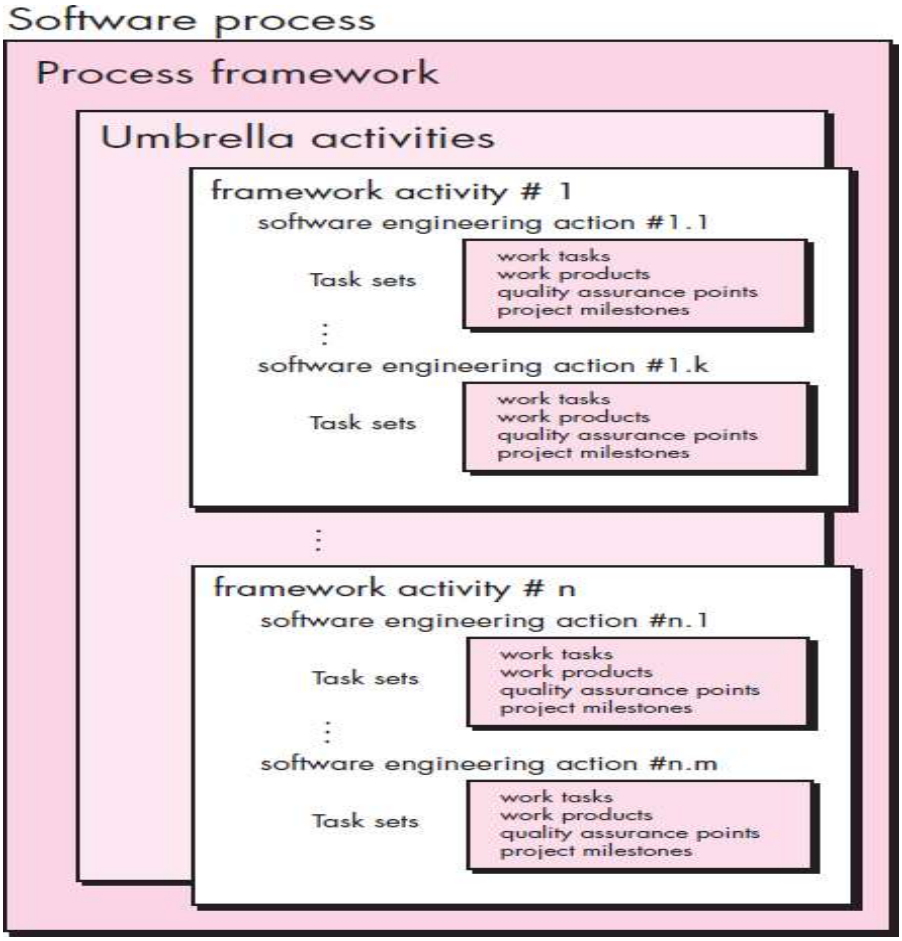
A software process framework is an organized approach that combines processes, methods, and tools to guide software development. It serves as the backbone of any software engineering project.

Definition and Structure:

- A software process framework outlines the essential steps and tasks that must be followed to develop software. These steps are general enough to apply to any software project.
- **Example Steps:**
  - **Gathering Requirements:** Understanding what the software needs to do.
  - **Communication:** Constant interaction with the team and stakeholders.
  - **Delivery:** Final product handover.

3. Detailed Breakdown of Software Process Framework:

The software process framework can be thought of as a structured plan or recipe, ensuring that all necessary activities are covered to produce high-quality software.



Step-by-Step Workflow:

1. **Umbrella Activities:** These are ongoing tasks that run throughout the software development process.
    - **Examples:** Risk management, quality assurance, and configuration management.
2. **Framework Activities:** Major phases that define the main tasks in a software project.
    - **Examples:** Communication, planning, modelling, construction, and deployment.
3. **Task Sets:** Each framework activity is further broken down into smaller, manageable tasks.

- **Example:** During the construction phase, tasks might include coding individual modules and running unit tests.

4. **Engineering Actions:** Specific tasks within the framework, focusing on creating work products, conducting quality checks, and achieving milestones.
    - **Example:** Writing code (engineering action) produces code files (work products), which are then reviewed for quality.
5. **Iteration of Framework Activities:** Each step is repeated as needed, refining the product until it meets the required standards.

III. A GENERIC PROCESS MODEL

The Generic Process Framework is a simplified, universal approach that applies to nearly every software development project. It is often referred to as a "universal recipe" that guides teams through the creation of software.

Key Activities in the Generic Process Framework:

1. **Communication:**
    - Purpose: To interact with customers and stakeholders to gather clear requirements and understand the project scope.
    - **Example:** A team developing an e-commerce website meets with the client to understand which features (like payment options or product filters) are needed.
2. **Planning:**
    - Purpose: To create a roadmap that outlines tasks, schedules, resources, and risks.
    - **Example:** The project manager creates a Gantt chart that details when each task (like coding the login system) should be completed and assigns it to team members.
3. **Modeling:**
    - Purpose: To create models or diagrams that represent how the software will function and be structured.
    - **Example:** Developers create flowcharts to map out user interactions within a mobile app.

4. **Construction:**
    - Purpose: To build the software by writing and testing code.
    - **Example:** The development team codes the main functionalities of the app, like user authentication, and runs unit tests to ensure each part works correctly.
5. **Deployment:**
    - Purpose: To deliver the finished software to the customer, who then uses it and provides feedback.
    - **Example:** After testing, the app is released on the App Store, and users can download and provide feedback on their experience.

5. Umbrella Activities in Detail

These activities support the main framework activities by addressing cross-cutting concerns that affect the entire development process.

Examples of Umbrella Activities:

1. **Risk Management:**
    - Involves identifying potential risks and creating strategies to mitigate them.
    - **Example:** A backup plan is created in case a critical team member is unavailable, or contingency resources are allocated if project delays occur.
2. **Software Quality Assurance (SQA):**
    - Ensures that the software meets predefined quality standards through planned reviews and testing.
    - **Example:** Before releasing a new app version, it undergoes thorough testing on different devices to ensure it functions smoothly across various platforms.
3. **Software Configuration Management (SCM):**
    - Manages changes in the software's configuration to keep track of all modifications and versions.
    - **Example:** Developers use tools like Git to manage code changes, enabling them to revert to previous versions if new changes cause issues.

4. **Measurement:**
    - Involves tracking project metrics like cost, time, and resources to ensure the project remains within budget and schedule.
    - **Example:** Regularly reviewing the time spent on coding and comparing it against the initial estimates to adjust schedules if needed.
5. **Formal Technical Review (FTR):**
    - Regular meetings that assess the project's progress and review technical aspects of the software.
    - **Example:** Weekly team reviews to check progress, identify challenges, and plan the next steps in development.