SPM ENDSEM

UNIT 3

Objectives of Activity Planning

Activity planning is a crucial aspect of project management that ensures the systematic execution of tasks within a project.

The main objectives of activity planning are:

1. Determining Project Feasibility
    - To assess whether the project objectives can be achieved within the given time and resource constraints.
    - Example: In a software development project, activity planning helps evaluate if a new feature can be delivered within a two-month deadline.
2. Resource Allocation
    - To identify and assign the necessary resources, including personnel, equipment, and materials, for project tasks.
    - Example: Allocating skilled developers, designers, and testers for specific modules in a mobile application project.
3. Defining Milestones and Deliverables
    - To establish clear checkpoints and deliverables to measure progress.
    - Example: Setting milestones for completing the UI design, backend development, and system testing phases.
4. Ensuring Task Dependencies are Identified
    - To understand the sequence of activities and manage dependencies to avoid delays.
    - Example: Ensuring that database schema design is completed before starting the API integration task.
5. Minimizing Risks and Uncertainties
    - To identify potential risks early and plan for contingencies to minimize disruptions.
    - Example: Adding buffer time in the schedule to account for unforeseen technical challenges during integration.
6. Optimizing Resource Utilization
    - To prevent under-utilization or overburdening of resources by balancing workload distribution.

- Example: Distributing tasks evenly among team members to maintain productivity and avoid burnout.

7. Establishing a Realistic Schedule
    - To create a timeline that aligns with stakeholder expectations while being achievable.
    - Example: Estimating and scheduling 3 weeks for front-end development based on team capacity and complexity.
8. Facilitating Communication
    - To provide a shared understanding of the project timeline and responsibilities among stakeholders.
    - Example: Using Gantt charts to visually communicate the project schedule and dependencies.
9. Tracking and Monitoring Progress
    - To enable regular monitoring and control of the project's progress to ensure alignment with the plan.
    - Example: Weekly review meetings to compare actual progress against the planned schedule.
10. Breakdown of Work into Manageable Activities
    - Complex projects are divided into smaller, manageable tasks to ensure clarity and focus.
    - Example: In an online shopping system, activities can include user interface design, payment gateway integration, and database setup.

Example of Activity Planning

In an online shopping system development project, activity planning might involve:

- Identifying activities such as database setup, front-end development, backend APIs, and payment gateway integration.
- Sequencing tasks, e.g., completing backend APIs before integrating them with the front end.
- Estimating 1 month for development and allocating 2 developers for front-end and 2 for backend tasks.

By addressing these objectives, activity planning ensures that projects are executed efficiently, on time, and within budget.

Project Schedules

In the case of large projects, detailed planning for subsequent stages will be postponed till the details of the necessary task is available at an earlier point in time.

Before starting the work on a project of perhaps in a bigger project phase, a project plan should be created to show the date when each activity should start, and end and the amount of each resource needed.

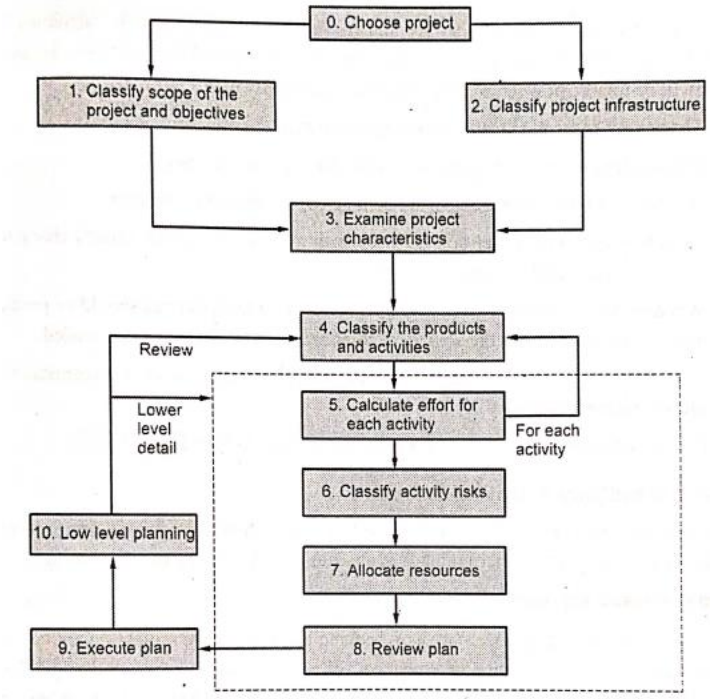
When a plan is improved to a level of detail it is called a project schedule.

It is process of organizing tasks, resources, and timeframes to achieve project objectives within a defined timeline.

It involves creating a roadmap for completing tasks and ensuring efficient resource utilization.

There are four phases to developing a project schedule: -

- The initial stage in creating a plan is deciding which activities must be completed and in what sequence they must be completed. On this basis, we can create an ideal activity plan, that is, if resources are not limited, ideally a time plan for each action can be taken.
- The second phase, ideal activity plan now becomes the subject of activity risk analysis, the purpose of which is to identify potential problems that can come up with an activity plan that will almost certainly affect the allocation of resources.
- The third phase is to allocate resources. Expected resources availability may limit how some activities can be taken, and our ideal plan may need to take this into account.
- The last step is scheduling. Once resources are allocated for each activity, we can create and produce a project schedule, which contains the start and end dates of the plan and an overview of the resource requirements for each activity.



Project Activities

Project Activities are the specific tasks or steps required to achieve the project objectives.

They form the building blocks of a project and are defined during the project planning phase.

These are very specific, practical, and well-defined measures.

By reading the description of the project activity, we can immediately describe the project in our mind effortlessly.

Activities need to de defined to fulfil these standards. Any activity that does not fulfil these standards should be reviewed.

Characteristics of Project Activities

1. Clearly Defined: Activities must have a clear objective and deliverable.
2. Sequenced: Each activity should be placed in a logical order based on dependencies.
3. Time-Bound: Activities should have a defined start and end time.
4. Resource-Dependent: Require specific resources (e.g., personnel, tools).
5. Measurable: Progress and outcomes should be measurable.

The activity-based approach, product-based approach, and the hybrid approach are the three basic techniques to determining the activities or tasks that make up a project.

syllabus:

1. Activity-Based Approach

- Definition: This approach focuses on identifying activities based on the work required to deliver the project. Activities are determined by breaking down the project into tasks and subtasks.
- Process:
    1. Start with the project objectives.
    2. Divide the project into phases or stages (e.g., design, development, testing).
    3. Identify specific tasks required in each phase.
    4. Assign resources and timelines to each task.
- Example: In an online shopping system project:
    - Phase: Frontend Development
    - Activities: Designing user interface, coding pages, testing UI components.

2. Product-Based Approach

- Definition: This approach derives activities from the end product or deliverables. The focus is on what needs to be delivered rather than how it is done.
- Process:
    1. Define the deliverables or outputs of the project.
    2. Break each deliverable into smaller components.
    3. Determine the tasks required to produce each component.
    4. Map out dependencies between tasks.
- Example: In an online shopping system project:
    - Deliverable: User Authentication System
    - Activities:
        - Designing the login/logout mechanism.
        - Creating a password recovery feature.
        - Testing authentication and authorization.

3. Hybrid Approach

- Definition: Combines elements of both activity-based and product-based approaches to determine activities. This approach allows for flexibility by leveraging the strengths of both methods.
- Process:
    1. Identify project phases and deliverables simultaneously.
    2. Break down each deliverable into tasks while considering the overall workflow.
    3. Balance between task-specific and deliverable-specific focus.
    4. Adjust activities based on dependencies and constraints.
- Example: In an online shopping system project:
    - Use the activity-based approach for overall project phases (e.g., development, testing).
    - Use the product-based approach for specific deliverables (e.g., shopping cart feature, payment gateway integration).

Sequencing and Scheduling

Sequencing of Activities: -

Sequencing involves determining the order in which project activities will be performed.

It considers dependencies between tasks and ensures logical progression toward project objectives.

Steps in Sequencing

1. List Activities: Identify all the tasks to be performed in the project.
2. Identify Dependencies: Determine relationships between tasks (e.g., Task B cannot start until Task A is completed).
3. Classify Dependencies:
    - Finish-to-Start (FS): Task B starts only after Task A finishes.
    - Start-to-Start (SS): Task B starts simultaneously or shortly after Task A begins.
    - Finish-to-Finish (FF): Task B finishes only after Task A finishes.
    - Start-to-Finish (SF): Rare case where Task B finishes only after Task A starts.
4. Use Dependency Tools: Employ tools like Dependency Matrix or Precedence Diagramming Method (PDM) to map dependencies.
5. Document Activity Sequence: Represent the sequence using charts or lists for reference.

Example

For an online shopping system:

- Task A: Develop Login Page → Task B: Test Login Page (FS)
- Task C: Design Payment Gateway → Task D: Integrate Payment Gateway (FS)

Scheduling of activities

Scheduling involves assigning specific timelines to the sequenced activities, defining when and how tasks will be completed.

Key Elements of Scheduling

1. Milestones: Specific points in time marking important achievements (e.g., "Login module completed by Day 5").
2. Duration Estimation: Estimating the time required for each activity.
3. Resource Allocation: Assigning resources (personnel, tools, budget) to tasks.

4. Critical Path Analysis (CPA): Identifying the longest sequence of dependent tasks (critical path) to determine the project's minimum duration.
5. Slack/Float: Measuring the time a task can be delayed without affecting the overall project timeline.

Scheduling Techniques

1. Gantt Charts: A visual representation of the schedule showing tasks, durations, and dependencies.
2. PERT (Program Evaluation and Review Technique): Uses optimistic, pessimistic, and most likely estimates to calculate expected task durations.
3. CPM (Critical Path Method): Focuses on identifying the critical path and ensuring critical tasks are not delayed.

Example

Using a Gantt chart for an online shopping system:

- Task A: Day 1–3: Develop Login Page
- Task B: Day 4–5: Test Login Page
- Task C: Day 6–8: Design Payment Gateway
- Task D: Day 9–11: Integrate Payment Gateway

Sequencing and scheduling are vital to efficient project management.

Proper sequencing ensures logical progression, while effective scheduling ensures timely completion of tasks.

By using appropriate tools and techniques, project managers can streamline workflows and avoid delays.

Project Scheduling Techniques

1. Critical Path Method (CPM):

It identifies the sequence of critical tasks that determines the shortest project duration.

It helps you to determine both longest and shortest time taken to complete a project.

There are three essential elements which are considered: -

- The tasks required to complete the project.
- Which tasks depend on the completion of others.
- A time estimates for each activity.

Steps:

1. List all tasks and their durations.
2. Identify dependencies.
3. Calculate early start, early finish, late start, and late finish for each task.
4. Identify critical path (tasks with zero slack).

It is mostly used in projects with predictable durations.

Example:

- There are four tasks A, B, C, and D.
- A task B and D can only begin after task A completes, where as task C has no restrictions.
- Task A will be time sensitive as any delay in its completion can delay in task B and D. Thus, A is called as critical task.
- This helps in identifying and separating the independent variables.
- Finally, it adds milestones to the project.

2. PERT

PERT stands for Program Evaluation and Review technique.

It is a way to schedule flow of tasks in a project and estimate total time taken to complete it.

It uses probabilistic time estimates to handle uncertainty in task durations.

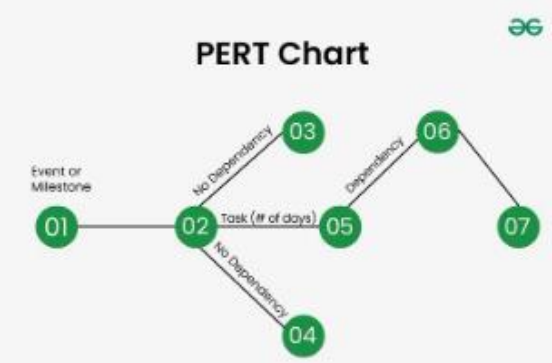
Time estimates: -

- Optimistic Time (O): Quickest time you can complete the project.
- Pessimistic Time (P): Longest time it will take to complete the project.
- Most likely Time (M): How long it will take to finish your project if there are no problems.
- Expected time is calculated as: (O + 4M + P) / 6.

It is mostly suitable for R & D or projects with high uncertainty.

In this technique, a PERT chart is made which represent a schedule for all the specified tasks in the project.

A PERT chart is a project management tool used to plan and schedule tasks, illustrating the sequence and timing of project activities.



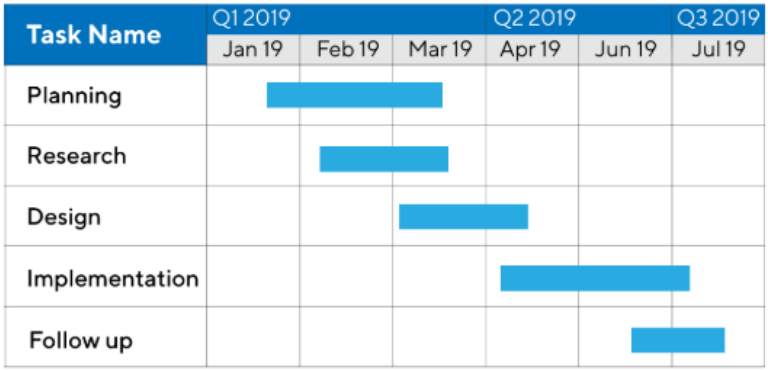
3. Gantt Chart

It is a visual representation of tasks along a timeline.

A Gantt Chart is a type of bar graph that project managers use for planning and scheduling in complex project.

It represents each task horizontally on a bar chart, which shows the start and end dates, and they frequently include deadlines and dependencies of tasks.

It is easier to visualize the progress of the project and see how different tasks interact with one another.



4. Task list

- One of the simplest project scheduling techniques is the creation of task list.
- Create the task list using word processor or spreadsheet software.
- It creates a list of tasks and include important information like the task manager, start date, deadline and completion status.

5. Milestone Charts

- Focuses on key project deliverables rather than individual tasks.
- It is simple and concise and helps in tracking significant project achievements.

6. Time Boxing

- Fixed timeframes are assigned for completing tasks regardless of scope.
- It prevents over-engineering and ensures focus on priority tasks.
- It is commonly used in Agile projects.

Difference between PERT and CPM

Aspect	PERT	CPM
Abbreviation	PERT stands for Project Evaluation and Review Technique.	CPM stands for Critical Path Method
Definition	PERT is a technique of project management which is used to manage uncertain (i.e., time is not known) activities of any project.	CPM is a technique of project management which is used to manage only certain (i.e., time is known) activities of any project.
Orientation	It is event oriented technique which means that network is constructed on the basis of event.	It is activity oriented technique which means that network is constructed on the basis of activities.
Model Type	It is a probability model.	It is a deterministic model.
Focus	It majorly focuses on time as meeting time target or estimation of percent completion is more important.	It majorly focuses on Time-cost trade off as minimizing cost is more important.
Precision	It is appropriate for high precision time estimation.	It is appropriate for reasonable time estimation.
Nature of Job	It has Non-repetitive nature of job.	It has repetitive nature of job.
Crashing	There is no chance of crashing as there is no certainty of time.	There may be crashing because of certain time bound.
Dummy Activities	It doesn't use any dummy activities.	It uses dummy activities for representing sequence of activities.
Sustainability	It is suitable for projects which required research and development.	It is suitable for construction projects.

Network Model

A Network Model is a graphical representation of the sequence and interrelationships of project activities.

It visually illustrates the dependencies, sequence, and critical paths for tasks in a project, helping in planning, scheduling, and managing projects effectively.

The network is typically represented using nodes and arrows, where:

- Nodes represent activities or milestones.
- Arrows represent dependencies or precedence relationships between activities.

Features of Network Model are: -

- It shows the order in which activities must be completed.
- Facilitates calculation of start and finish times for each activity.
- Helps identify the sequence of critical tasks that directly impact the project timeline.
- Determines the amount of time non-critical tasks can be delayed without affecting the project completion. It is referred as Slack time.
- Assists in resource allocation and leveling.

Components of Network model are: -

1. Nodes (Boxes): Nodes represent project activities or tasks. Each node contains information such as the activity name, duration, resources, and any other relevant details.
2. Arrows (Lines): Arrows represent dependencies between activities. They show the logical sequence in which activities must be performed. The direction of the arrow indicates the flow of work.
3. Relationships: There are four types of relationships between activities:
    - Finish-to-Start (FS): The successor activity cannot start until the predecessor activity finishes.
    - Start-to-Start (SS): The successor activity cannot start until the predecessor activity starts.
    - Finish-to-Finish (FF): The successor activity cannot finish until the predecessor activity finishes.
    - Start-to-Finish (SF): The successor activity cannot finish until the predecessor activity starts.
4. Critical Path: The critical path is the longest sequence of dependent activities that determine the shortest possible duration for completing the project. Activities on the critical path have zero slack or float time, meaning any delay in these activities will directly impact the project's overall timeline.

Types of Network Diagram are: -

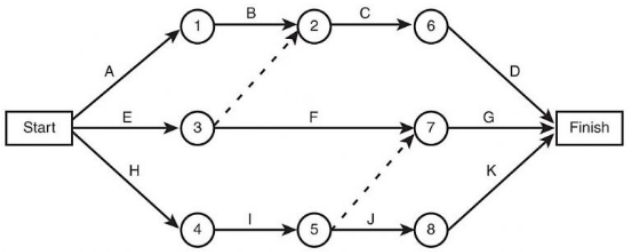
1. Arrow Diagram Method (ADM):

The ADM, or activity network diagram, depicts project-related activities using arrows.

In ADM:

- The tail of the arrow signifies the beginning of the action, while the head represents the end.
- The length of the arrow often represents the duration of the action.
- Each arrow connects two boxes, called "nodes." Nodes are used to indicate the beginning or end of an activity in a sequence. The initial node of an activity is sometimes referred to as the "i-node," whereas the last node of a sequence is sometimes referred to as the "j-node."
- The only relationship between nodes and activity that an ADM chart may indicate is "finish to start" (FS).

In some cases, ADM network diagrams require the inclusion of "dummy activities" to represent indirect relationships. For instance, if activity C can only start after both activities A and B are finished, but A and B are not directly related, a dummy activity is added between B and C to show this dependency.



An Arrow Network Diagram

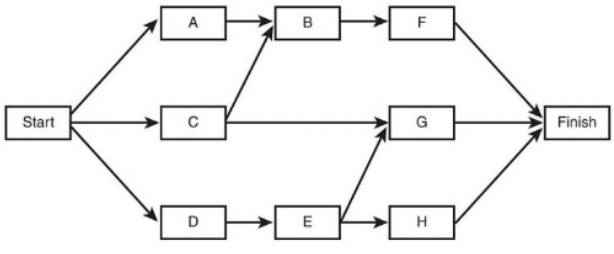
2. Precedence Diagram Method (PDM):

In the Precedence Diagramming Method, each node signifies an activity, while arrows depict relationships between them.

These arrows symbolize all four potential connections: "Finish to Start" (FS), "Start to Start" (SS), "Finish to Finish" (FF), and "Start to Finish" (SF).

- “Finish to Start” (FS): When an activity cannot start before another activity finishes
- “Start to Start” (SS): When two activities can start simultaneously
- “Finish to Finish” (FF): When two tasks need to finish together
- “Start to Finish” (SF): This is an uncommon dependency and is only used when one activity cannot finish until another activity starts

PDM allows you to write lead and lag times beside the arrows.



A Precedence Network Diagram

Network models help in Project planning in following way: -

- Visualizing the Project Flow: Clearly show the sequence of activities and their interdependencies.
- Identifying the Critical Path: Determine the longest path of dependent activities to find the minimum project completion time.
- Resource Allocation: Highlight resource bottlenecks and assist in effective allocation.
- Risk Mitigation: Identify potential delays and critical activities that require close monitoring.
- Tracking Progress: Enable tracking of project progress and adherence to schedules.
- Decision Making: Provide a clear basis for rescheduling, resource adjustment, or risk management.

Key Terms used are: -

1. Predecessor Activity: An activity that must be completed before another activity starts. Example: In a house construction project, laying the foundation is a predecessor to building the walls.
2. Successor Activity: An activity that starts only after its predecessor(s) is completed. Example: Painting the walls is a successor to plastering the walls.
3. Parallel Activity: Activities that can be executed simultaneously without waiting for each other. Example: Wiring and plumbing in a house can be done in parallel.

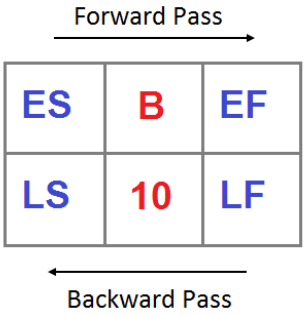
Forward pass and Backward pass in project scheduling

Forward pass is a technique to move forward through network diagram to determining project duration and finding the critical path or Free Float of the project.

Whereas backward pass represents moving backward to the end result to calculate late start or to find if there is any slack in the activity.

Following are some important terms which are taken in consideration: -

- Early Start (ES): The earliest time an activity can begin, assuming all predecessor activities have been completed. It is plotted on top left corner box.
- Early Finish (EF): The earliest time an activity can be completed, assuming it starts as early as possible. It is plotted on top right corner box.
- Late Start (LS): The latest time an activity can start without delaying the project's completion. It is plotted on bottom left corner box.
- Late Finish (LF): The latest time an activity can be completed without delaying the project's completion. It is plotted on bottom right corner box.



The forward pass determines the earliest possible start (ES) and earliest possible finish (EF) times for each activity in a project.

The backward pass determines the latest allowable start (LS) and latest allowable finish (LF) times for each activity without delaying the project.

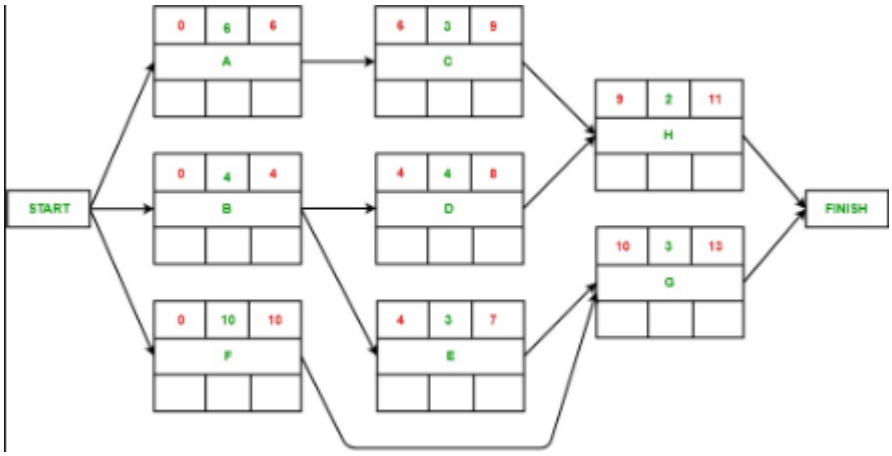
For e.g.

Activity	Duration (in weeks)	Precedents
A	6	–
B	4	–
C	3	A
D	4	B
E	3	B
F	10	–
G	3	E,F
H	2	C,D

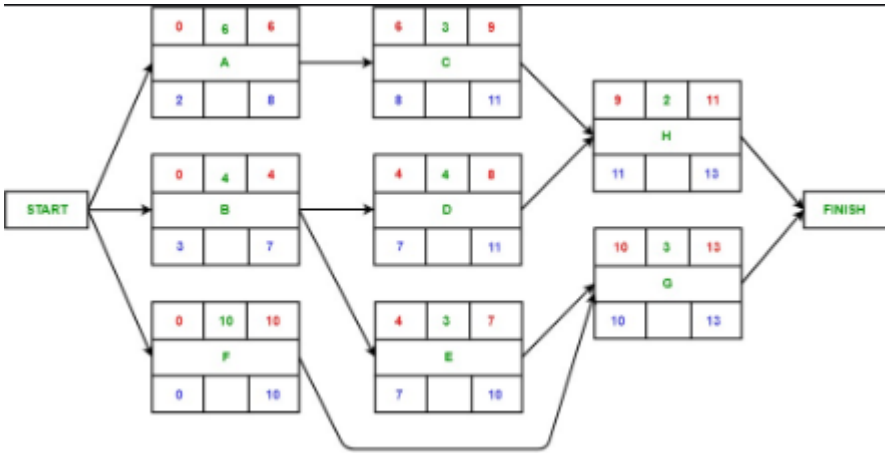
Node representation: -

Earliest Start	Duration	Earliest Finish
Activity Label		
Latest Start	Float	Latest Finish

By forward Pass: -



By backward Pass: -



Risk Management

The process of minimizing any potential issues that can adversely affect a project's schedule is known as risk management.

An unforeseen circumstance that could have an impact on people, procedures, technology and resources used in a project is considered as risk.

Risk management is an important part of project planning activities.

It involves identifying and estimating the probability of risks with their order of impact on the project.

It helps to minimize the negative effects of risks and maximize opportunities during the project lifecycle.

Purpose:

- Ensures project goals are met despite uncertainties.
- Helps allocate resources effectively to mitigate risks.
- Promotes proactive planning instead of reactive problem-solving.

The processes involved in Risk Management are:

1. Risk Identification: -

- It involves brainstorming activities and the preparation of risk list.
- Brainstorming is a group discussion technique where all the stakeholders meet together.
- Preparation of a risk list involves the identification of risks that are occurring continuously in previous software projects.
- It uses tools like SWOT Analysis and Risk Checklists.
- It output a risk register listing all identified risks with description.

2. Risk Analysis and Prioritization: -

It is a process that consists of the following steps:

- Identifying the problems causing risk in projects.
- Identifying the probability of occurrence of the problem.
- Identifying the impact of the problem.
- Assigning values to step 2 and step 3 in the range of 1 to 10.
- Calculate the risk exposure factor which is the product of values of Step 2 and Step 3.
- Prepare a table consisting of all the values and order risk based on risk exposure factor.

3. Risk Planning: -

In these strategies are developed to address the identified risks.

Strategies:

- Avoid: Change project plans to eliminate risk.
- Mitigate: Take steps to reduce the likelihood or impact of the risk.
- Transfer: Shift the risk to a third party (e.g., insurance).
- Accept: Acknowledge the risk and monitor it without active response.

It outputs the risk response strategies for each significant risk.

4. Risk Monitoring: -

In this technique, the risk is monitored continuously by reevaluating the risks, the impact of risk, and the probability of occurrence of the risk.
This ensures that:

- Risk has been reduced.
- New risks are discovered.
- The impact and magnitude of risk are measured.

It output the updated risk register with progress report

5. Risk Communication: -

- Ensures all the stakeholders are informed about risks and responses.
- Clear communication ensures alignment among project teams.

Risk Identification

It involves brainstorming activities and the preparation of risk list.

Brainstorming is a group discussion technique where all the stakeholders meet together.

Preparation of a risk list involves the identification of risks that are occurring continuously in previous software projects.

It uses tools like SWOT Analysis and Risk Checklists.

It output a risk register listing all identified risks with description.

IT Project Risk Identification Framework: -

The IT Project Risk Identification Framework provides a structured approach to identifying risks specific to IT projects.

It ensures that all potential risks are considered and systematically documented during the early stages of the project lifecycle.

The framework focuses on understanding the nature of IT-specific risks and their impact on project outcomes.

Components of the IT Project Risk Identification Framework

1. Stakeholder Involvement: Engage key stakeholders to identify risks from various perspectives (business, technical, operational).
2. Project Scope Review: Analyze the project scope to identify areas of uncertainty or ambiguity.
3. Historical Data Analysis: Review risks encountered in similar past projects to identify patterns or common issues.
4. Checklist and Frameworks: Use standardized checklists tailored to IT projects to identify potential risks.
5. Environmental Scanning: Identify external factors (legal, economic, technological) that could affect the project.
6. Risk Categorization: Classify risks into categories (technical, organizational, external) for better tracking and management.

Types of Risks are: -

- Technical risks: Risks associated with technology, tools, and infrastructure. For e.g. Software bugs or integration issues.
- Management risks: Risks arising from poor project planning, execution, or resource management. For e.g. Inaccurate time or cost estimation.
- Operational risks: Risks affecting the day-to-day operations during project execution. For e.g. Key team members leaving mid-project.
- Organizational risks: Risks related to organizational policies, culture, and priorities. For e.g. Lack of management support.

- External Risks: Risks originating outside the project's control. For e.g. Economic downturn or natural disasters.
- Security and Privacy Risks: Risks related to data breaches, cyberattacks, or failure to meet privacy regulations. For e.g. Unauthorized access to sensitive information.
- Financial Risks: Risks impacting the project budget or financial resources. For e.g. unexpected cost overruns.

By using the IT Project Risk Identification Framework and understanding the types of risks, project teams can better anticipate, plan for, and mitigate risks, ensuring project success.

Risk Prioritization

The process of determining which project risks are the most important so they may be dealt with first is known as Risk Prioritization.

The likelihood of the risk and the possible harm it causes to the organization should be used to determine priority.

It helps project managers focus on the most critical risks that could impact the project's objectives most, ensuring efficient use of time and resources in managing them.

Steps in Risk Prioritization: -

- Assigns scores to the risk: Assign numerical value to the impact that risk could make on the project and probability of risk occurrence in scale 1 to 10.
- Risk Assessment Matrix: Calculate Risk exposure factor for each risk based on the impact and probability of risk occurrence. Risk exposure factor is the product of impact and the probability of risk occurrence.
- Categorize Risks: Categorize risks into tiers based on scores such as:
    - High Priority: Requires immediate attention.
    - Medium Priority: Needs monitoring and planned action.
    - Low Priority: May be monitored periodically or handled later.
- Evaluate Time Sensitivity: Consider when the risk is likely to occur. Address urgent risks first. Example: A risk expected in the current project phase should take precedence.
- Analyze Interdependencies: Assess how one risk could trigger or amplify others. Example: Delayed hardware delivery (operational risk) could lead to project timeline overruns (project risk).
- Use Decision Making Tools: Tools like Failure Mode and Effects Analysis (FMEA) or Pareto Analysis help prioritize risks by their criticality.

Risk prioritization is a vital step in **Risk Management** to ensure that the most pressing threats to a project are identified and addressed systematically.

Risk Planning

The Risk Planning phase involves creating strategies and action plans to manage, mitigate, or eliminate risks identified during the risk identification process.

The objective is to ensure that potential risks do not derail the project's objectives by proactively addressing them.

Steps in Risk Planning are: -

- Prioritize Risks: Use techniques like the Risk Matrix or Risk Scoring to determine which risks need immediate attention. Focus on high-impact and high-probability risks.
- Define Risk Response Strategies: Select appropriate actions to manage each risk. Common strategies include:
    - Avoidance: Change plans to eliminate the risk.
    - Mitigation: Reduce the impact or likelihood of the risk.
    - Transfer: Shift the risk to another party (e.g., insurance).
    - Acceptance: Acknowledge the risk and prepare to deal with its impact.
- Develop Contingency Plans: Outline specific actions to take if a risk materializes. Example: For a risk of server downtime, a contingency plan might involve switching to a backup server.
- Allocate Resources: Assign resources such as budget, personnel, and tools to manage and mitigate risks effectively.
- Establish Risk Monitoring Procedures: Define how risks will be tracked and monitored throughout the project lifecycle. Example: Use risk management tools or periodic review meetings.
- Create a Risk Management Plan: Document all risk-related strategies, plans, and processes. Include: Risk description, Response strategies, Contingency plans, Resource allocation.
- Communicate the Plan: Share the risk management plan with stakeholders to ensure everyone understands their roles in managing risks.

The Risk Planning phase is a cornerstone of effective risk management, enabling teams to anticipate challenges and establish a roadmap for addressing them systematically.

Risk Control

Risk control involves the continuous monitoring, evaluation, and management of risks throughout the project lifecycle.

It ensures that risks remain within acceptable levels and that mitigation strategies are implemented effectively.

Steps in Risk Control: -

- Monitor Risks: Regularly assess known risks to ensure they remain under control. Use tools like risk dashboards or progress reports.
- Detect New Risks: Stay vigilant to identify emerging risks. Example: Unexpected market changes or new regulations.
- Implement Risk Responses: Execute planned mitigation or contingency actions for identified risks. Example: Deploying additional resources to meet deadlines.
- Review and Update Risk Plans: Adjust the risk management plan based on new information or project changes.
- Evaluate Effectiveness: Measure the success of risk responses and refine strategies as necessary.

Importance of Risk Control

- Proactive Management: Ensures timely identification and mitigation of risks.
- Minimized Impact: Reduces the likelihood of risks disrupting the project.
- Stakeholder Confidence: Keeps stakeholders informed and assured about risk management efforts.
- Adaptability: Helps the project team respond to unexpected changes dynamically.

Risk Management Tools

SpiraPlan by Inflectra

- SpiraPlan is a comprehensive solution that integrates project management, risk management, and quality assurance in one platform.
- It includes a robust risk management feature to identify, analyze, prioritize, and mitigate risks throughout the project lifecycle.
- Offers highly customizable dashboards to visualize project progress, risks, and other critical metrics.
- Compatible with Agile, Scrum, Kanban, and Waterfall project management approaches.
- Provides tools for tracking requirements, tasks, and deliverables to ensure project alignment.
- SpiraPlan seamlessly integrates with popular tools like JIRA, Git, and Jenkins for better project coordination.
- Facilitates team collaboration through shared documents, discussion boards, and notifications.
- Generates detailed reports on risks, including their likelihood, impact, and mitigation strategies.
- Designed to handle small projects as well as large-scale, enterprise-level initiatives.

- Cloud-based and accessible from multiple devices, ensuring that team members can stay connected anywhere.

Risk Management Studio

- Risk Management Studio (RMS) is designed for identifying, assessing, and managing risks across various industries.
- Supports frameworks like ISO 31000 and ISO 27001 to ensure compliance with international risk management standards.
- Offers tools for risk identification, qualitative and quantitative risk assessments, and likelihood-impact analysis.
- Provides a centralized, customizable risk register to document and monitor risks effectively.
- Enables users to simulate different risk scenarios and their potential impact on project objectives.
- The installation is simple, and the annual price includes free upgrades and customer support.
- Since RM Studio is simple to learn, anyone can start using it right away like a pro.
- It offers importing capability for switching from Excel to RM Studio.
- Tracks and evaluates risk controls to ensure effective mitigation strategies are implemented.
- Features an intuitive interface that simplifies navigation and risk management workflows.
- Generates detailed reports and visual dashboards for better insights into risk trends and status.
- Includes tools for auditing and documenting risk management processes for accountability and transparency.
- Suitable for small businesses, medium enterprises, and large organizations managing complex risks.

GRC Cloud

- GRC stands for Governance, Risk and Compliance platform.
- GRC Cloud is a comprehensive tool for managing governance, risk, and compliance processes within organizations.
- Operates entirely on the cloud, enabling easy accessibility and integration across multiple locations and devices.
- Provides real-time insights into risk exposure and compliance status through advanced dashboards.
- Helps in creating, distributing, and tracking compliance with organizational policies and regulations.
- Features qualitative and quantitative risk assessment capabilities to evaluate and prioritize risks.
- Supports customizable workflows for risk mitigation, incident management, and compliance tracking.

- Includes tools to conduct audits and generate compliance reports to meet regulatory requirements.
- Ensures high levels of data security with encryption and compliance with global data protection standards.
- There is a system in place for alerts and it operates automatically.
- The system has the ability to activate emails depending on the danger and timing of the occurrence.
- The risk score is used to prioritize the hazards in this risk assessment, which is based on the score. This provides a way to show the applications risk areas using a heat-map as well.
- Assists in evaluating and managing risks associated with vendors, suppliers, and other external entities.
- Designed to cater to the unique GRC needs of industries like finance, healthcare, and technology.

UNIT 4

Project Tracking and Control

Project Tracking and Control activities occurs concurrently with project execution process group activities, so that the project is monitored and controlled while the project work is being completed by implementing the appropriate level of oversight and corrective action.

Project Tracking and Control are processes that are required to track, review, and regulate the project's progress and performance.

It also identifies the areas where the project management method needs to be changed and initiates the necessary changes.

The project is frequently inspected and measured against the project plan to ensure that cost, schedule and scope deliverables are within acceptable limits and that risks and issues are continuously monitored and remedied as needed.

Monitoring and Control Processes

After the work plan has been published and the project has started, progress should be monitored.

This requires monitoring what is happening, comparing actual performance to the schedule and adjusting as needed.

Revise plans and schedules as needed to get the project back on track as much as possible.

The project management plan serves as a project's foundation. This document serves as a guide for tracking and controlling the project.

The inputs to the monitoring and controlling project are: -

- Project management plan: It is a primary source of information for the project's execution, monitoring and control. It is a plan as well as any other subsidiary plans that may be required.
- Work performance information: Information about project activities is known as Work Performance information. This information contains progress, deliverables, costs and quality assurance validations.
- Rejected change requests: When examined in the context of determining how the project is progressing, reject change requests might be instructive.

Following are some tools and methods used for monitoring and regulating the project operations: -

- Expert judgement: Project managers and team members can use expert judgement to make judgements, such as whether to take corrective or preventive actions, based on current project information and previous experience with similar project.

- Earned Value Technique (EVT): It is a method of calculating the current project schedule and cost performance for project managers. This data can then be used by project managers to predict future schedule and cost performance.
- Organizational Project Management: This approach offers project managers with detailed instructions and processes to enable effective monitoring and control during each stage of project.
- Project Management Information System (PMIS): A PMIS helps us to keep track of and regulate variables like cost and resource utilization. It can also help project managers to calculate, and manage earned value data, as well as request and update project data automatically.
- Kanban Boards: Visualize project workflows by tracking tasks using columns for "To Do," "In Progress," and "Done."
- Gantt Charts: Represent project schedules with bars showing task durations and dependencies.
- Project Calendars: Track timelines and deadlines by visualizing key milestones and deliverables.
- Status reports: Provide regular updates on progress, roadblocks, and completed tasks.
- Phone, Online, or In-Person Interviews: Gather real-time updates on progress and challenges directly from team members.
- Cost Monitoring Tools: Track expenditures against the budget using tools like Microsoft Excel or Primavera.
- Software Configuration Management (SCM) Tools: Tools like Git, Team Foundation Server, and Ansible track software versions and ensure proper configuration management.

Each method/tool is designed to provide transparency, control, and a clear view of progress, ensuring that the project stays on track while addressing risks and challenges effectively.

Following are the outputs of project work monitoring and control: -

- Recommend corrective actions: The project manager or team employs professional judgement to provide solutions to problems that have occur by comparing this information to the project plan.
- Preventive actions recommendation: The project manager or team utilizes professional judgement to identify strategies to prevent project risks by comparing this information with the project plan.
- Predictions: Forecasts allow for the projection of successful project outcomes based on work performance data collected during the monitor and control project work process.
- Recommended defect repairs: When a product fails to satisfy quality standards, this output suggests what has to be done to fix it.
- Requested changes: These are new actions that are required to accomplish the project’s goals.

What is project control? Explain the different types of control mechanism in details.

Project control is the process of tracking, reviewing, and regulating project progress to ensure that it stays on track to meet its objectives, timelines, and budgets.

It involves identifying deviations from the plan and implementing corrective actions.

Types of Project Control Mechanisms:

1. Schedule Control: Ensures that the project is progressing according to the planned timeline. Example: Using Gantt charts or earned value analysis to identify delays and reschedule tasks.
2. Cost Control: Monitors project expenditures to keep them within the budget. Methods: Cost variance analysis, tracking actual vs. planned costs using tools like Primavera or Excel.
3. Scope Control: Ensures that only approved changes to the project scope are implemented. Key Tools: Change control boards and documented approval processes.
4. Quality Control: Verifies that the project's deliverables meet the required standards and specifications. Techniques: Inspections, peer reviews, and testing.
5. Risk Control: Identifies and mitigates risks that could derail the project. Approach: Regular risk assessments and contingency plans.
6. Resource Control: Ensures that resources (human, technical, and material) are allocated and utilized effectively. Example: Monitoring workloads and reallocating team members if needed.
7. Communication Control: Maintains clear and consistent communication among stakeholders to avoid misunderstandings. Tools: Status reports, regular meetings, and dashboards.
8. Change Control: Manages and monitors changes to the project scope, schedule, or resources to ensure they are justified and documented. Example: Using a change control process with defined steps and approvals.

By employing these mechanisms, project control helps achieve the desired outcomes efficiently while addressing risks, maintaining quality, and staying within budget and timeline constraints.

Explain plan monitor control cycle used in the project in detail with example

The Plan-Monitor-Control Cycle is a systematic approach used in project management to ensure that projects are executed efficiently and effectively.

This cycle consists of three key stages—Planning, Monitoring, and Controlling—and is iterative, enabling continuous improvement throughout the project lifecycle.

The three phases are as follows: -

1. Planning

- It defines project task, timelines, resources, and deliverables.
- Activities includes,
    - Establishing objectives and deliverables.
    - Creating Schedules.
    - Allocating resources and estimating budgets.
    - Identifying risks and preparing mitigation plans.
- Example: For an online shopping system, the planning phase would involve defining features (e.g., login, cart, payment gateway), determining a timeline, and allocating developers to different tasks.

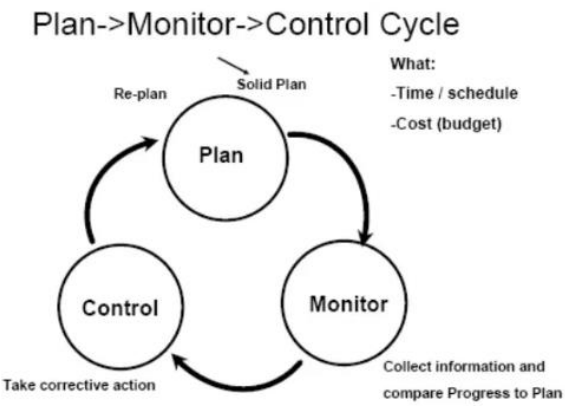
2. Monitoring

- Track project progress and performance to identify deviations from the plan.
- Activities includes,
    - Collecting data on progress (e.g., task completion, budget spent).
    - Measuring performance using Key Performance Indicators (KPIs).
    - Analyzing variances in scope, cost, or schedule.
    - Tools: Dashboards, status reports, and earned value analysis.
- Example: For the online shopping system, the project manager uses project management software (e.g., Jira) to track whether tasks like building the user interface or integrating the payment gateway are on schedule.

3. Controlling

- Take corrective actions to bring the project back on track.
- Activities includes,
    - Addressing delays by reallocating resources or extending timelines.
    - Adjusting budgets if costs exceed estimates.
    - Managing changes in scope through change control processes.
    - Communicating progress and issues with stakeholders.
- Example: If the online shopping system's payment integration is delayed, the project manager might assign additional developers to speed up completion or adjust subsequent deadlines

After controlling, the plan is updated based on new insights, and the monitoring-control cycle repeats.



Benefits of Plan-Monitor-Control Cycle

1. Proactive Management: Helps identify issues early.
2. Continuous Improvement: Iterative process ensures adaptability.
3. Stakeholder Communication: Provides transparency through regular updates.
4. Better Resource Utilization: Aligns resources with project priorities.

This cycle ensures that projects meet objectives, timelines, and budgets while adapting to challenges and changes effectively.

Collection of Project Data

Collecting project data is a critical aspect of project tracking, monitoring, and control.

It involves gathering accurate, timely, and relevant information about various project activities to ensure that the project stays on track and deviations are addressed promptly.

Objectives of Data Collection: -

- To monitor project progress and performance.
- To identify deviations from the project plan.
- To ensure effective communication among stakeholders.
- To support decision-making and risk management.

Types of Project Data Collected: -

- Progress Data: Information on task completion, milestones achieved, and percentage of work completed.
- Schedule Data: Start and end dates of tasks, delays, and timeline adjustments.
- Cost Data: Budget allocation, expenditures, and cost variances.
- Quality Data: Defects identified, testing results, and customer feedback.
- Risk Data: Identified risks, mitigations, and impacts on the project.

Data Collection Techniques

- Status Reports: Regular updates on tasks, milestones, and risks.
- Meetings and Interviews: Direct communication with team members for qualitative insights.
- Dashboards: Real-time visualization of project metrics through tools like Tableau or Power BI.
- Surveys and Feedback: Gathering opinions from stakeholders for project improvement.

Partial Completion Reporting

Partial completion reporting refers to the practice of documenting and communicating the progress of a project or task before it is fully completed.

This is crucial for keeping stakeholders informed, ensuring alignment with project goals, and identifying potential issues early.

Objectives of Partial Completion Reporting

- Progress Monitoring: Track ongoing activities and measure progress against the project schedule.
- Stakeholder Communication: Keep all stakeholders updated on the current status and achievements.
- Issue Identification: Detect problems or delays early to take corrective actions.
- Resource Optimization: Reallocate resources based on the current progress and future requirements.

Methods for Partial Completion Reporting

- Progress Reports: Written reports summarizing the status of tasks or phases. Example: "Module X development is 75% complete, with testing scheduled to start next week."
- Dashboards and Visual Reports: Real-time visualization tools, such as Gantt charts, burndown charts, or Kanban boards.
- Standup Meetings: Daily or weekly team meetings to discuss ongoing tasks and blockers.
- Milestone Reviews: Formal reviews at specific checkpoints to evaluate progress and readiness for the next phase.

Data Collection Methods

Phone vs. Online vs. In-Person Interviews

- Phone Interviews: Data collection conducted via phone calls.
- Online Interviews: Virtual interviews using tools like Zoom, Microsoft Teams, or survey forms.
- In-Person Interviews: Face-to-face interactions conducted at a physical location.

Aspect	Phone Interviews	Online Interviews	In-Person Interviews
Convenience	Easy to schedule and conduct remotely.	Highly flexible; accessible from anywhere.	Requires physical presence; less convenient.
Cost	Affordable; no travel costs.	Very low cost, especially for large groups.	High cost due to travel, venue, and logistics.
Time Efficiency	Saves travel time; suitable for quick data collection.	Allows simultaneous interviews (group settings).	Time-consuming due to setup and travel.
Data Quality	Limited by audio-only interaction.	Can include visuals (screenshare, forms).	Rich data through verbal and non-verbal cues.
Participant Comfort	Suitable for shy participants.	Comfortable for tech-savvy individuals.	Best for building trust and rapport.
Suitability	Small to medium-scale projects.	Large-scale or geographically dispersed teams.	Small, critical discussions or sensitive topics.

Visualizing Progress

Visualizing progress in project management involves using graphical and visual tools to track and communicate the current state of the project.

These methods enhance clarity, simplify complex data, and support informed decision-making.

Benefits of Visualizing Progress: -

- Provides an **overview of project performance** and milestones.
- Helps **identify bottlenecks** and areas of improvement.
- Ensures all stakeholders have a **common understanding** of the project's status.
- Supports **real-time decision-making** by presenting data in an intuitive format.
- It is simple to understand, as the information is presented in a single, consistent style.
- The project team is required to review project on a regular basis.
- To create a report, only small amount of effort is required.

Methods used for visualizing project progress are: -

1. Gantt chart

- A Gantt chart is one of the earliest and most straightforward methods for monitoring project progress.
- Essentially, this is an activity bar chart with activity labels that shows the duration and dates of planned activities.
- The graphic shows reported progress, and a today cursor shows which tasks are ahead of or behind schedule immediately.
- It is used for visualizing task schedules, dependencies and milestones.
- A Gantt Chart is a type of bar graph that project managers use for planning and scheduling in complex project.
- It represents each task horizontally on a bar chart, which shows the start and end dates, and they frequently include deadlines and dependencies of tasks.
- It is easier to visualize the progress of the project and see how different tasks interact with one another.

2. Slip Chart

- A slip chart compares planned progress against actual progress, highlighting deviations.
- Clearly shows tasks that are ahead, on time, or behind schedule.
- Reflects ongoing changes in the project timeline.
- Tracks individual tasks' performance against the plan.
- Highlights areas requiring immediate attention to avoid delays.
- Aids in making informed decisions to adjust timelines or resources.
- Offers a straightforward view of schedule slippage.
- The more the slip line slants, the more deviation there is from the original plan.
- At regular intervals, new slip lines are inserted and as they accumulate, the project manager may see whether or not the project is progressing.
- Rearranging the schedule is indicated as highly jagged slip line.

3. Timeline

- The timeline chart is a tool for documenting and visualizing how targets have changed during the course of the project.
- A timeline is a linear visual representation of key events, tasks, or milestones in a project, arranged chronologically.
- It shows tasks or milestones in a chronological order.
- It provides a quick overview of the entire project without excessive detail.
- It emphasizes significant events and deadlines.
- It is simple and intuitive for stakeholders to understand.
- It can be used for short-term or long-term projects.
- It easily integrates with other tools like presentations or dashboards.
- It is helpful during both project execution and for post-implementation analysis.
- Analyzing the time chart and the causes of the changes can reveal estimation errors or other mistakes that, with this knowledge, could be avoided in the future.

Visual Project Management

The ability to effectively communicate with our team on both the small details and the big picture of our projects is one of the biggest and most crucial problems in project management.

After all its doubtful that individuals will work on the proper tasks in the proper order if they are unaware of the importance of each task and its timely completion to the project's success.

Visual Project Management is one of the finest ways to communicate with our team.

By using tools to help everyone involved visualize the project's needs and progress, visual project management is a technique for organizing and visualizing processes over traditional projects.

When it comes to information delivery, visual project management brings up information given in a way that anyone can consume it at a time, place and manner that is convenient to them.

Kanban Boards

Definition: A Kanban board is a visual representation of tasks and their statuses in a project, using columns to indicate different stages of work (e.g., To Do, In Progress, Done).

Purpose: It helps teams visualize work, identify bottlenecks, and optimize the flow of tasks across the project lifecycle.

Components:

- Columns: Represent stages of the workflow (e.g., Backlog, Development, Testing, Done).
- Cards: Represent individual tasks or work items.
- Work in Progress Limits: The number of cards that can ever be in column at once is the WIP limit. There can never be more then three cards in a column with WIP limit.
- Commitment Point: Board backlogs are common for Kanban teams. Ideas for projects that the team can pick up when ready are posted here by clients and coworkers.
- Delivery point: The workflow of the kanban team ends at the delivery point. When the consumer has the product or service in their hands, it is the delivery point for majority of teams.

Workflow Visualization: Enables teams to see all tasks in one place and understand their current status at a glance.

Task Prioritization: Tasks are arranged in order of priority, ensuring the most critical work gets attention.

Flexibility: The board can be customized to suit the needs of any team, project, or organization.

Real-Time Updates: Allows for immediate reflection of changes in the workflow, keeping the team aligned.

Limit Work in Progress (WIP): Encourages setting WIP limits to prevent overloading team members and maintain efficiency.

Collaboration: Facilitates team communication by providing a shared platform to track tasks and progress.

Focus on Continuous Delivery: Helps teams focus on delivering small, incremental improvements rather than waiting for a big release.

Transparency: Offers complete visibility of who is working on what, improving accountability and understanding among team members.

Bottleneck Identification: Highlights areas where tasks are stuck or delayed, enabling quick resolution.

Metrics and Reporting: Kanban tools often provide metrics like cycle time (time to complete a task) and lead time (time from task creation to completion).

Digital Tools: Kanban boards can be physical (whiteboards with sticky notes) or digital, using tools like Trello, Jira, or Asana for enhanced functionality.

Scalability: Useful for small teams, large enterprises, or cross-functional teams to manage workflows across various domains, such as IT, marketing, or HR.

A Kanban Board is a versatile tool that fosters visual management, continuous improvement, and effective collaboration, making it an essential component of modern project management practices.

Project Calendars

A Project Calendar is a schedule management tool used in project management to organize and track project activities, resources, and deadlines over a defined timeline.

It provides a structured framework to ensure project tasks are completed on time and within scope.

Definition: A project calendar is a visual representation of the time allocated for project activities, milestones, and deadlines.

Purpose: It ensures efficient time management by clearly defining when tasks should begin and end.

Components:

- Workdays: Indicates the days available for work (e.g., Monday to Friday).
- Holidays: Lists non-working days like weekends and public holidays.
- Task Durations: Specifies the time allocated for each activity.

Customizable: Can be tailored to fit the unique requirements of a project, such as including overlapping schedules or varying work hours.

Resource Allocation: Helps identify resource availability and plan resource utilization effectively.

Dependencies: Shows task dependencies, ensuring that activities requiring inputs from others are scheduled accordingly.

Milestones: Marks significant events or achievements in the project timeline for better tracking.

Improved Coordination: Enhances team collaboration by providing a unified schedule that all stakeholders can refer to.

Conflict Resolution: Identifies potential scheduling conflicts, such as overlapping tasks or resource overuse, and enables proactive adjustments.

Risk Mitigation: Anticipates delays by visualizing the impact of schedule changes and preparing contingency plans.

Cost Monitoring

Cost monitoring is the process of tracking, analyzing, and controlling the expenditures of a project to ensure it stays within the approved budget.

It involves real-time observation of spending patterns and aligning them with the financial plan to identify variances and take corrective actions.

Four steps in Project Cost Management are: -

1. Plan Cost Management

- The process of setting up the rules, guidelines and documentation for budgeting, managing and controlling project expenses is known as plan cost management.
- This step involves defining how the project costs will be estimated, budgeted, and controlled.
- It offers the direction and guidance on how the project costs will be handled throughout the project.
- It creates a framework for consistent and transparent financial management throughout the project lifecycle.
- It develops a cost management plan outlining tools, techniques, and procedures.
- It establishes guidelines for cost reporting and escalation processes.

2. Estimate Costs

- Estimating costs involves forecasting the amount of financial resources required for project activities.
- Its purpose is to determine the cost of labor, materials, equipment, and other resources.

- Various techniques such as Analogous estimation, bottom-up estimation, and parametric-estimation can be used for estimating the project cost.
- In analogues estimation, we use costs from past projects with similar scopes.
- In bottom-up estimation, we sum up costs from detailed task-level estimates.
- In parametric estimation, we use mathematical models based on project metrics.
- Cost estimates are the prediction based on the facts available at the time.
- Estimated costs involves the selection and evaluation of costing options for starting and finishing the project.
- To get the best prices for the project, cost trade-offs and hazards like make vs. purchase, buy vs. lease and resource sharing should be taken into account.
- Cost estimates are often represented in units of some currency, but in some cases, other units of measure are used to assist comparisons by removing the impacts of currency fluctuations, such as staff hours or staff days.
- As the project moves through the project life cycle, its estimation accuracy will rise.

3. Determine budget

- It aggregates the estimated costs to create a project budget baseline.
- To develop an authorized cost baseline, the process of determining budget involves adding up the expected costs of several work packages or individual activities.
- It establishes a cost baseline by which project performance can be tracked and managed.
- All the funds authorized to carry out the project are included in the project budget.
- The time-phased project budget is the version of the cost baseline that has been approved, but management reserves are not included.
- It provides a financial framework against which actual costs are tracked.
- It Allocate funds for each task and milestone.
- It includes contingency reserves for unexpected expenses.

4. Control cost

- This step involves managing changes to the project budget and ensuring expenditures align with the financial plan.
- Its purpose is to maintain financial discipline and mitigate overspending.
- Activities includes,
    - Monitor actual costs versus the budget using tools like Earned Value Management (EVM).
    - Identify and address cost variances.
    - Implement corrective actions to bring costs back on track.
    - Tracking work performance in relation to budgetary expenditures.
    - Limiting anticipated cost overruns to acceptable levels.
    - Making sure all change requests are handled quickly.
    - Controlling the real changes as they happen.
    - Notifying the proper parties of any approved changes and their costs.

Earned Value Analysis

It is a project management approach that is use for monitoring and controlling purpose.

It evaluates project performance and progress by comparing planned work with completed work and its cost.

It integrates scope, time, and cost metrics to provide insights into a project's health.

EVA is a quantitative method to measure a project's performance against its baseline plan by analyzing three core metrics: Planned Value (PV), Earned Value (EV), and Actual Cost (AC).

The planned value represents the budgeted cost of work scheduled to be done by a specific time. It is used as a benchmark for tracking progress.

The Earned value reflects the budgeted value of work that has actually been completed by a given time. It indicates progress made in terms of cist and scope.

The Actual cost represents the actual expenditure incurred to complete the work at a given point in the project. It helps to identify cost variances.

Key Performance Indicators

- Cost Variance (CV): Difference between EV and AC (CV = EV - AC).
- Schedule Variance (SV): Difference between EV and PV (SV = EV - PV).
- Cost Performance Index (CPI): Efficiency of resource usage (CPI = EV / AC).
- Schedule Performance Index (SPI): Efficiency of project scheduling (SPI = EV / PV).

EVA provides estimates for future project performance and costs, such as:

- Estimate at Completion (EAC): Predicted total cost of the project.
- Estimate to Complete (ETC): Expected cost to finish remaining work.

EVA combines scope, schedule, and cost, making it a holistic tool for tracking project performance.

It identifies deviations from the plan early, enabling corrective actions to avoid delays or budget overruns

EVA is widely used in industries such as construction, software development, and manufacturing to manage complex projects.

It follows standardized frameworks such as those outlined in PMBOK (Project Management Body of Knowledge).

Results of EVA are often displayed in graphical formats like S-curves for better comprehension of performance trends.

EVA focuses on quantitative metrics and may not consider qualitative factors like team morale or resource availability.

Project Tracking

Project tracking is the process of continuously monitoring the progress of a project to ensure it stays aligned with the planned scope, schedule, and budget.

It involves assessing project performance, identifying risks or deviations, and making necessary adjustments to achieve project objectives.

Effective Approach to Track Projects

1. Create Project Outline
    - Define the project's scope, deliverables, and resources in a detailed project plan.
    - Use tools like Gantt charts, Kanban boards, or Work Breakdown Structures (WBS) to organize tasks.
    - Ensure stakeholders have a clear understanding of the project framework.
2. Establish Goals and Milestones
    - Break the project into smaller, manageable phases with specific objectives.
    - Set measurable milestones to track progress at key stages.
    - Ensure goals are SMART (Specific, Measurable, Achievable, Relevant, and Time-bound).
3. Check in Regularly
    - Schedule regular team meetings or stand-ups to assess progress and resolve bottlenecks.
    - Use performance indicators like Earned Value Analysis (EVA) or Key Performance Indicators (KPIs) to measure success.
    - Collect updates on task completion, risks, and resource utilization.
4. Establish Clear Deadlines
    - Assign realistic deadlines to tasks and sub-tasks, accounting for dependencies and resource constraints.
    - Use scheduling tools to visualize timelines and avoid overlaps.
    - Communicate deadlines effectively to all stakeholders to maintain accountability.

Project tracking and an effective approach to it, such as creating outlines, setting milestones, checking in, and maintaining clear deadlines, ensure project objectives are achieved efficiently while minimizing risks and deviations.

Status Report

A status report is a document or communication that provides an update on the progress of a project.

It outlines completed tasks, ongoing work, upcoming activities, and any issues or risks affecting the project.

It is a vital tool for keeping stakeholders informed, ensuring accountability, and facilitating decision-making.

Purpose of a Status Report

- Communicate Progress: Highlights what has been accomplished so far.
- Identify Risks: Brings attention to potential issues and challenges.
- Facilitate Decision-Making: Provides the necessary information to make informed adjustments.
- Track Milestones: Helps measure the project's progress against planned goals.
- Ensure Accountability: Maintains transparency across teams and stakeholders.

Four Features of a Good Status Report

1. Clarity and Conciseness: The report should present information in a clear and straightforward manner, avoiding technical jargon where possible. Use bullet points, summaries, and visuals to ensure readability and easy understanding.
2. Focus on Key Metrics: Include critical data like completed milestones, budget status, resource utilization, and upcoming deadlines. Highlight deviations from the plan and corrective measures being taken.
3. Timeliness: The report should be delivered regularly and on time to ensure that stakeholders have up-to-date information. Delayed reports may lead to outdated decisions and missed opportunities for intervention.
4. Action-Oriented Content: Outline actions to be taken to address risks, delays, or resource constraints. Provide clear recommendations or requests for stakeholder input when necessary.

Benefits of Good Status Reporting

- Keeps all stakeholders aligned with the project's progress.
- Facilitates proactive management of risks and challenges.
- Enhances transparency and trust among team members.
- Helps in evaluating performance and adjusting plans effectively.

Change Control

Change control focuses on systematically managing changes to a project's baseline to ensure that they are evaluated, approved, implemented, and documented effectively.

Steps in the Change Control Process:

1. Identify Change: A change request is raised when a modification is required. The change can be related to functionality, design, resources, or scope.
2. Analyze Impact: Assess the potential impact of the change on the project's scope, timeline, budget, and quality. Identify risks and dependencies.
3. Evaluate and Approve: The change is reviewed by the Change Control Board (CCB). If approved, it is scheduled for implementation.
4. Implement Change: Make the approved change in the software system. Update the documentation and version history.
5. Test and Validate: Verify that the change works as intended and does not introduce new issues.
6. Close Change Request: Mark the change as complete and update the configuration baseline.

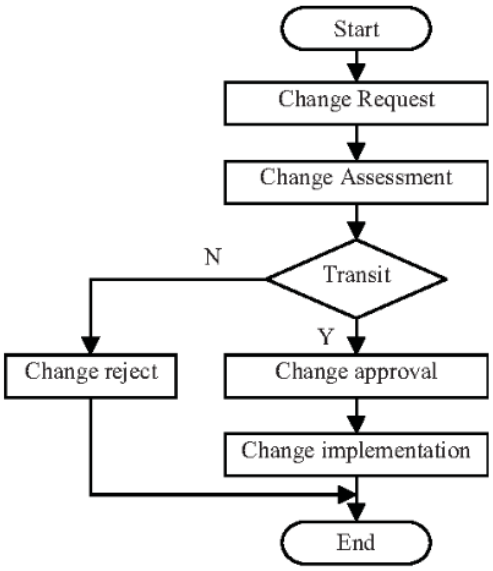


Fig. 3. Change control process

Version Control

Version control involves managing changes to software artifacts, such as source code, documents, and configurations.

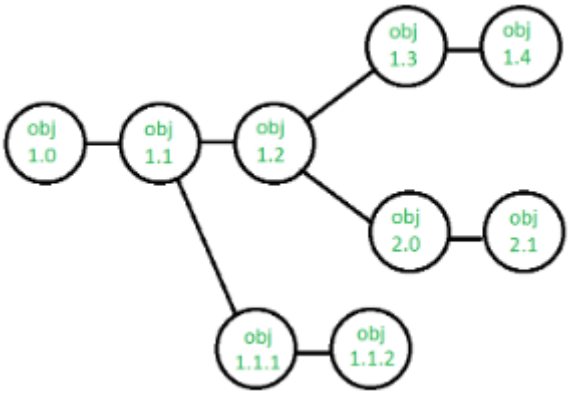
It enables teams to maintain a history of changes, work collaboratively, and revert to previous versions if needed.

Key Features of Version Control:

- Track Changes: Maintains a complete history of changes made to files.

- Branching and Merging: Allows parallel development by creating branches and integrating them.
- Collaboration: Multiple team members can work on the same codebase without conflict.
- Conflict Resolution: Identifies and resolves conflicts when merging changes.

Example Tools: Git, Apache Subversion (SVN), Mercurial.



Software Configuration Management (SCM)

Software Configuration Management (SCM) is the process of tracking, controlling, and managing changes in software throughout its lifecycle.

It ensures the integrity and traceability of the software system and supports collaborative development.

Goals of SCM:

1. Ensure software quality and reliability.
2. Facilitate version control and change management.
3. Prevent unauthorized changes.
4. Maintain consistency across software builds and releases.

Tasks in the SCM Process

The SCM process encompasses several tasks essential for effective management of software configuration items (SCIs):

1. Configuration Identification: Identify software artifacts (code, documents, libraries) to be managed as configuration items (CIs). Define naming conventions, versioning rules, and metadata.
2. Version Control: Track multiple versions of configuration items. Maintain a history of changes and manage branches for parallel development.
3. Change Control: Evaluate and approve change requests systematically. Update baselines after implementing and validating changes.

4. Configuration Auditing: Ensure that configuration items adhere to defined standards. Verify that the configuration baseline is accurate and consistent.
5. Reporting: Provide updates on the status of configuration items, changes, and progress. Facilitate traceability and accountability.

Participants in the SCM Process

SCM involves various stakeholders who perform specific roles and responsibilities.

1. SCM Manager: Oversees the SCM process, ensures compliance with policies, and coordinates with the project team.
2. Developers: Commit changes to the repository. Use version control tools to manage their code and resolve conflicts.
3. Quality Assurance (QA) Team: Validate changes and ensure they meet quality standards before integration.
4. Configuration Auditor: Conducts audits to verify compliance with configuration baselines.
5. Project Manager: Ensures that SCM processes align with project timelines and objectives.
6. Change Control Board (CCB): Reviews and approves change requests. Determines the impact of changes on the project's scope, budget, and schedule.
7. Release Manager: Handles the deployment of software builds. Ensures that all configuration items are included and documented.
8. End Users/Clients: Provide feedback on issues and suggest changes.

Software Configuration Management Tools

1. Git

- Git is a distributed version control system widely used for tracking changes in source code and supporting collaborative development.
- Each developer has a complete copy of the repository, allowing offline work.
- Developers can create isolated branches for features or bug fixes and merge them seamlessly.
- Git maintains a detailed log of changes, making it easy to track and revert updates.
- Changes can be reviewed before committing them to the repository.
- Git is optimized for performance and handles large projects efficiently.
- Git integrates with tools like Jenkins, GitHub Actions, and GitLab CI/CD for automation.
- Supports teamwork through pull requests, issue tracking, and code reviews.
- Git is free and open-source, making it accessible to developers worldwide.
- Compatible with Windows, macOS, and Linux.

- Git repositories are hosted on platforms like GitHub, GitLab, and Bitbucket for easier collaboration.

2. Team Foundation Server

- Team Foundation Server (TFS) by Microsoft is an enterprise-grade tool for managing the entire software development lifecycle.
- Provides a centralized system for managing code changes, in contrast to Git's distributed model.
- Seamlessly integrates with Visual Studio, enhancing productivity for developers.
- Tracks tasks, bugs, and enhancements within projects.
- Automates the build process, ensuring consistency and quality.
- Supports test planning, execution, and reporting.
- Facilitates Scrum and Kanban methodologies for project management.
- Enables automated pipelines for code integration and deployment.
- Role-based access control ensures secure collaboration.
- Supports code reviews, pull requests, and team discussions.
- Suitable for both small teams and large enterprises, with robust project management features.

3. Ansible

- Ansible is an open-source automation tool widely used for IT configuration management, deployment, and orchestration.
- Does not require agents to be installed on target systems, reducing complexity.
- Uses YAML for defining configurations, making it easy to learn and use.
- Ensures that repeated executions of a playbook yield the same results, preventing unintended changes.
- Manages Linux, Windows, and cloud-based environments.
- Centralized scripts to define automation tasks and workflows.
- Treats infrastructure setups as code, enabling reproducibility and version control.
- Works seamlessly with Git, TFS, and other version control systems.
- Uses SSH for communication and ensures encrypted data transmission.
- Backed by a strong community with a vast repository of pre-built modules and roles.
- Ansible Tower (paid version) provides advanced UI, role-based access, and reporting features.

Managing Contracts

The management of contracts with customers, vendors, partners or staff is known as contract management or contract administration.

Contract management involves overseeing the creation, execution, and evaluation of contracts to maximize operational and financial performance and reduce risks.

Managing contracts is about ensuring compliance with terms and conditions while achieving organizational goals.

The purpose is to establish clear expectations, mitigate risks, and ensure smooth collaboration between stakeholders.

Ensures accountability, clarifies responsibilities, and reduces the potential for disputes.

Software like DocuSign, PandaDoc, and TFS aids in effective contract management.

The Stages of Contract Management

1. Create: The initial stage where the contract is drafted based on the project requirements, terms, and conditions.
2. Negotiate: Both parties discuss and modify the draft to ensure mutual agreement and satisfaction.
3. Approve: Formal approval is obtained from relevant stakeholders to finalize the contract.
4. Finalize: The contract is made legally binding and prepared for execution.
5. Manage: The contract is monitored and managed throughout its lifecycle to ensure compliance and performance.

Challenges of Contract Management

1. Complexity: Managing large volumes of contracts with varying terms can be overwhelming.
2. Compliance Issues: Ensuring adherence to legal, financial, and organizational policies.
3. Communication Gaps: Misunderstandings between stakeholders about contract terms or deliverables.
4. Lack of Visibility: Difficulty in tracking contract performance and milestones without robust tools.
5. Resource Constraints: Inadequate staff or tools to manage contracts effectively.
6. Risk of Breach: Non-compliance with terms leading to disputes or penalties.

Benefits of Contract Management

1. Improved Compliance: Ensures adherence to regulatory and organizational standards.
2. Cost Savings: Avoids penalties and identifies cost-saving opportunities.
3. Enhanced Transparency: Maintains a clear record of terms, obligations, and amendments.
4. Risk Mitigation: Anticipates and addresses potential issues early.
5. Better Collaboration: Facilitates trust and cooperation between stakeholders.
6. Streamlined Processes: Automates repetitive tasks, reducing delays and errors.

Types of Contracts in Software Project Management

1. **Fixed-Price Contracts**: The client pays a predetermined amount regardless of resources used.
    - **Pros**: Predictable cost, easy budgeting.
    - **Cons**: Risk of scope creep or underestimated costs.
2. **Time-and-Materials Contracts**: Payment based on actual time and materials used.
    - **Pros: Flexible, adaptive to project changes.**
    - **Cons: Unpredictable costs.**
3. **Cost-Reimbursement Contracts**: The client reimburses the contractor for costs plus an agreed profit margin.
    - **Pros: Low risk for contractors.**
    - **Cons: High risk for clients due to potential cost overruns.**
4. **Incentive-Based Contracts**: Rewards the contractor for achieving specific performance targets.
    - **Pros: Encourages quality and efficiency.**
    - **Cons: Complex to draft and monitor.**
5. **Agile Contracts**: Designed for iterative development, focusing on deliverables rather than fixed scope.
    - **Pros: High adaptability to changes.**
    - **Cons: May lack clarity in overall scope.**

Write short notes on Team Structure and Virtual Team in a Software Project.

Team Structure in a Software Project

1. Definition:
   Team structure refers to the organizational framework defining roles, responsibilities, and relationships among team members in a software project.
2. Types of Team Structures:
   - Functional Team: Groups are organized by specific skills or functions (e.g., testing, development).
   - Project-Based Team: Entire team is dedicated to a specific project.
   - Matrix Team: Combines functional and project-based structures for flexibility.
3. Importance:
   - Clear communication and accountability.
   - Efficient allocation of resources.
   - Ensures all project activities are well-coordinated.
4. Key Roles:
   - Project Manager: Oversees the project and ensures timely delivery.
   - Developers: Responsible for coding and implementation.
   - Testers: Ensure software quality and bug-free delivery.
   - Designers: Create user interfaces and experiences.

Virtual Team in a Software Project

1. Definition:
   A virtual team is a group of individuals working on a project from different geographic locations using digital tools for collaboration.
2. Features:
   - Team members rarely meet in person.
   - Use of communication platforms like Zoom, Slack, or Microsoft Teams.
   - Operate in different time zones and cultural settings.
3. Advantages:
   - Access to a global talent pool.

- Cost savings on office space and infrastructure.
- Flexibility in working hours.

4. Challenges:
    - Difficulty in maintaining team cohesion.
    - Communication barriers due to time zones and languages.
    - Dependency on technology for collaboration.
5. Best Practices:
    - Regular virtual meetings to track progress.
    - Use of project management tools like Jira or Trello.
    - Clear documentation and role definitions.

Both team structures and virtual teams play critical roles in modern software projects, offering flexibility, collaboration, and efficient resource management.

UNIT 5

Understanding Behaviour

It refers to analyzing how individuals or groups act within an organization or team setting.

Understanding behavior is essential for predicting, guiding, and improving performance in a project environment.

Behavior can be categorized as individual behaviors, Group behaviors, or Organizational behaviors.

Individual behaviors includes personality traits, values, attitudes, and perception.

Group behavior involves teamwork, communication, and collaboration among members.

Organizational behavior encompasses the organization's culture, structure, and leadership.

Personality, emotions, motivation, etc are some internal factors that can influence the behaviour.

Whereas work environment, team dynamics, organizational culture are some external factors that can influence the behaviour.

It helps in resolving conflicts effectively.

It Encourages better communication within teams.

It Increases productivity by identifying and addressing behavioral issues.

Organizational Behavior

The study of how people interact within groups in an organization, aiming to create more efficient and harmonious work environments.

Key Components:

- Individual level: Focuses on motivation, job satisfaction, and personal effectiveness.
- Group level: Studies team cohesion, communication, and decision-making.
- Organizational level: Examines culture, hierarchy, and structural design.

Principles of Organizational Behavior:

- Understanding the dynamics of leadership and influence.
- Recognizing the impact of organizational culture on employee behavior.
- Managing diversity in workgroups effectively.

Selecting the Right Person for the Job

Selecting the right person for the job is a critical aspect of team and organizational success.

It involves identifying a candidate whose skills, experience, and personality align with the job requirements and organizational culture.

Steps for Selecting the Right Person:

1. Understand Job Requirements: Clearly define the skills, qualifications, and responsibilities required for the role.
2. Develop a Job Description: Outline job duties, expected outcomes, and necessary skills.
3. Screen Candidates: Use resumes and applications to filter out unqualified candidates.
4. Conduct Interviews: Assess candidates' technical skills, soft skills, and cultural fit through structured interviews.
5. Administer Assessments: Include technical tests, psychometric evaluations, or case studies to validate a candidate's capabilities.
6. Check References: Contact previous employers or colleagues for feedback on the candidate's past performance.
7. Evaluate Fit with Team and Culture: Ensure the candidate aligns with the organization's values and team dynamics.
8. Make the Offer: Present a competitive offer that matches the candidate's expectations and the role's value.

Recruitment Process: -

The recruitment process involves attracting, identifying, and selecting the best candidate for a specific job. Here's the detailed process:

1. Identifying the Need: The process begins when a vacancy is identified due to growth, attrition, or a new requirement. Define the role's purpose and its contribution to organizational goals.
2. Creating a Job Description and Specification:
    - Job description: Specifies tasks, responsibilities, and working conditions.
    - Job specification: Details qualifications, experience, and personal qualities required.
3. Attracting Candidates: Use multiple channels like job portals, company websites, social media, and recruitment agencies to attract a diverse pool of candidates.
4. Screening and Shortlisting: Review resumes to identify candidates who meet the basic requirements. Use automated tools or manual review to shortlist candidates for further evaluation.

5. Interview Process: Conduct multiple interview rounds (e.g., technical, HR, managerial). Use structured or behavioral interviews to assess skills, experience, and personality.
6. Assessment Tests: Include skill-based tests, case studies, or group discussions to evaluate specific competencies.
7. Background Checks and References: Verify the candidate's professional background, criminal records, and work history. Speak to referees for feedback on the candidate's strengths and weaknesses.
8. Decision and Offer: Compare shortlisted candidates and select the best fit. Extend a formal job offer, including salary, benefits, and joining details.
9. Onboarding: Ensure a smooth onboarding process for the selected candidate to integrate them into the organization.

Oldham-Hackman Job Characteristics Model

The Oldham-Hackman Job Characteristics Model is a framework developed by J. Richard Hackman and Greg Oldham to understand how specific job characteristics impact employee motivation, performance, and satisfaction.

It suggests that well-designed jobs can lead to higher motivation, improved work quality, and greater job satisfaction.

The model identifies **five core job characteristics** that influence three critical psychological states, ultimately affecting personal and work outcomes.

Five Core Job Characteristics:

1. Skill Variety: The degree to which a job requires a variety of activities and skills. Example: A software developer involved in coding, testing, and debugging will find their job more engaging.

2. Task Identity: The extent to which a job involves completing a whole, identifiable piece of work from start to finish. Example: Building an entire website versus only designing a single page.

3. Task Significance: The impact of the job on others' lives or the organization. Example: A healthcare worker helping save lives has a high task significance.

4. Autonomy: The level of independence and discretion an employee has in scheduling tasks and making decisions. Example: A project manager has more autonomy compared to a junior team member.

5. Feedback: The clarity and quality of information received about job performance. Example: Regular feedback from a manager or end-users.

Three Critical Psychological States:

1. Experienced Meaningfulness: Derived from skill variety, task identity, and task significance. Employees feel their work is meaningful and worthwhile.
2. Experienced Responsibility: Comes from autonomy. Employees feel personally responsible for outcomes.
3. Knowledge of Results: Achieved through feedback. Employees understand how well they are performing.

When the core job characteristics lead to desired psychological states, the following outcomes are observed:

- High internal motivation.
- Improved job satisfaction.
- Better job performance.
- Reduced absenteeism and turnover.

The model uses an equation to quantify a job's potential to motivate:

$$\text{MPS} = \frac{\text{Skill Variety} + \text{Task Identity} + \text{Task Significance}}{3} \times \text{Autonomy} \times \text{Feedback}$$

A high MPS indicates a highly motivating job.

Limitations:

- It assumes that all employees are motivated by the same job characteristics.
- Some roles may lack flexibility for redesign.
- It may not account for external motivators like salary or rewards.

The Oldham-Hackman Job Characteristics Model highlights how well-structured jobs can lead to better motivation, job satisfaction, and performance.

Organizations can use this model to redesign jobs and foster a motivated and productive workforce.

Stress

Stress is the body's response to perceived challenges or threats, affecting emotional, physical, and mental well-being.

In a work context, stress results from demands that exceed an individual's capacity to cope.

Stress can be work related, personal, environmental, etc.

Work related stress are tight deadlines, ambiguous roles or responsibilities, unrealistic expectations, high workload or long working hours.

Personal stress can be financial issues, family responsibilities, health problems, etc.

Environmental stress can be poor workplace conditions, rapid technologies changes, etc.

Stress can be positive as well as negative.

The positive stress motivates the individuals to achieve goals and meet challenges.

Whereas the negative stress harms mental and physical health, leading to decreased performance.

Stress Management Techniques

1. Time Management: Prioritize tasks, set realistic deadlines, and delegate responsibilities.
2. Healthy Lifestyle: Regular exercise, balanced diet, and adequate sleep.
3. Work-Life Balance: Promote flexible schedules and personal time for employees.
4. Relaxation Techniques: Mindfulness, meditation, and breathing exercises.
5. Communication and Support: Encourage open dialogue between team members and managers.
6. Training Programs: Provide stress management workshops and coping skills training.
7. Employee Assistance Programs (EAPs): Professional counselling and mental health resources.

Stress is an unavoidable aspect of project management but can be effectively managed through proactive strategies.

Organizations that prioritize stress management foster healthier work environments, improving team performance and achieving project success.

Health and Safety

Health and safety are critical aspects of project management, ensuring the well-being of team members and stakeholders.

It involves creating a safe working environment, minimizing risks, and promoting a culture of health consciousness.

Health refers to ensuring the physical and mental well-being of all project participants.

Safety refers to protecting individuals from physical harm, accidents, or hazards in the workplace.

Importance of Health and Safety in Projects

- Protects employees from injuries and illnesses.
- Improves productivity by reducing absenteeism.
- Builds trust among team members and stakeholders.
- Ensures compliance with legal and regulatory requirements.
- Reduces costs associated with accidents, lawsuits, and insurance claims

Key Health and Safety Measures

- Risk Assessments: Identify potential hazards and evaluate their impact. Example: Analyzing risks in construction sites for falls or equipment malfunctions.
- Safety Training: Provide employees with training on handling equipment and emergency procedures. Example: Fire drills, first-aid training, or chemical safety protocols.
- Personal Protective Equipment (PPE): Ensure the availability and use of safety gear like helmets, gloves, and goggles.
- Workplace Ergonomics: Design workspaces to reduce strain and fatigue. Example: Adjustable chairs and proper lighting for IT professionals.
- Regular Inspections: Conduct audits to ensure compliance with safety standards. Example: Inspecting electrical wiring and machinery.
- Emergency Preparedness: Develop and implement emergency response plans. Example: Evacuation plans for natural disasters or workplace accidents.

Prioritizing health and safety in project management leads to a positive work environment, enhanced team morale, and successful project outcomes.

Effective health and safety measures are not only ethical but also vital for long-term organizational success.

Ethical and Professional Concerns

Ethical and professional concerns focus on maintaining integrity, fairness, and accountability throughout the project lifecycle.

These concerns ensure that the project is conducted responsibly, aligns with societal norms, and adheres to industry standards.

Moral principles that guide decision-making and behavior are called ethics.

Adhering to industry standards, expertise, and responsibilities in a respectful manner is referred as professionalism.

Ethical Concerns in Project Management: -

1. Conflict of Interest: Avoid situations were personal gains conflict with project objectives. Example: Awarding a contract to a relative without proper evaluation.
2. Transparency and Honesty: Provide accurate and truthful project updates and reports. Example: Avoid concealing project delays or budget overruns.
3. Confidentiality: Protect sensitive information, such as client data or trade secrets.
4. Compliance with Laws: Adhere to local, national, and international legal requirements. Example: Following labor laws and environmental regulations.
5. Fair Treatment: Treat all team members and stakeholders with respect and impartiality. Example: Avoiding discrimination based on gender, race, or religion.

Professional Concerns

1. Accountability: Accept responsibility for decisions and outcomes. Example: A project manager taking ownership of budget mismanagement.
2. Competence: Ensure that team members have the necessary skills and qualifications. Example: Assigning roles based on expertise rather than favouritism.
3. Respect for Stakeholders: Engage all stakeholders and consider their perspectives in decision-making. Example: Holding regular meetings to address concerns and feedback.
4. Adherence to Standards: Follow industry best practices, such as ISO standards or Agile methodologies.
5. Environmental and Social Responsibility: Reduce negative environmental impacts and contribute to societal well-being.

Ethical and professional behavior is fundamental to the success and sustainability of projects. It ensures that the project aligns with organizational values, complies with legal standards, and positively impacts society.

Explain briefly the FOUR (4) types of teams in an organization

In project management, different team structures are employed depending on the project's nature and organizational goals.

Each type of team has its unique purpose, characteristics, and advantages.

1. Functional Teams

Teams organized based on specific functional areas, such as finance, marketing, or IT are called as Functional Teams.

In this structure members report to a single functional manager.

Characteristics:

1. Focus on specialized tasks.
2. Clear hierarchy and responsibilities.
3. Knowledge and expertise concentrated in one domain.

Advantages:

- High efficiency due to specialization.
- Expertise is nurtured within the team.

Example: A marketing team responsible for creating and implementing advertising strategies.

2. Cross-Functional Teams

Teams composed of members from various functional areas working towards a common goal is called as Cross-Functional Teams.

Here members may report to their functional manager and a project manager.

Characteristics:

1. Diverse skill sets and expertise.
2. Encourages innovation and creative problem-solving.
3. Temporary or project-based.

Advantages:

- Promotes collaboration across departments.
- Solves complex problems requiring interdisciplinary expertise.

Example: A product development team including members from R&D, marketing, and finance.

3. Self-Managed Teams

Teams with autonomy to make decisions without a direct manager's supervision are called Self-Managed Teams.

Here the members collectively manage tasks and share leadership responsibilities.

Characteristics:

1. High level of independence.
2. Members take ownership of tasks and decisions.
3. Requires skilled and motivated individuals.

Advantages:

- Increases accountability and motivation.
- Reduces the need for supervision.

Example: A software development team deciding on project milestones and timelines.

4. Virtual Teams

Teams working remotely across different locations, often using digital communication tools are called Virtual teams.

Here members may belong to various departments or organizations.

Characteristics:

1. Relies on technology for communication and collaboration.
2. Members often work in different time zones.
3. Flexibility in work hours and locations.

Advantages:

- Access to a global talent pool.
- Reduces overhead costs, such as office space.

Example: A team of graphic designers collaborating on a project from different countries using tools like Slack and Zoom.

Each type of team structure serves a specific purpose within an organization.

Understanding these structures helps in choosing the right team for a project to ensure efficiency, collaboration, and success.

Combining these team types effectively can enhance organizational performance and adaptability.

Objectives of Managing People and Organizing Teams

Managing people and organizing teams effectively is critical for the success of any project. The objectives focus on creating a cohesive and productive work environment that aligns individual and team efforts with organizational goals.

1. Enhancing Team Performance

- Objective: Maximize the efficiency and effectiveness of the team.
- Explanation: Provide proper tools, resources, and guidance to ensure optimal team output and alignment with project goals.

2. Promoting Collaboration and Communication

- Objective: Foster a culture of open and effective communication among team members.
- Explanation: Encourage knowledge sharing, reduce conflicts, and ensure that everyone is aligned with the project's objectives.

3. Selecting the Right Talent

- Objective: Ensure the team consists of individuals with the right skills and expertise.
- Explanation: Focus on recruitment and assigning roles that match individuals' strengths, ensuring high-quality output.

4. Motivation and Job Satisfaction

- Objective: Keep team members motivated and satisfied with their roles.
- Explanation: Use motivational techniques like recognition, rewards, and career growth opportunities to enhance engagement.

5. Managing Diversity

- Objective: Utilize the strengths of a diverse team effectively.
- Explanation: Embrace differences in cultural backgrounds, perspectives, and skills to foster innovation and inclusivity.

6. Ensuring Health and Safety

- Objective: Create a safe and healthy working environment.
- Explanation: Implement policies to reduce stress, prevent workplace injuries, and promote overall well-being.

7. Building Team Cohesion

- Objective: Develop a sense of unity and trust among team members.
- Explanation: Organize team-building activities and encourage collaboration to establish a strong team spirit.

8. Effective Decision-Making

- Objective: Support teams in making informed and timely decisions.
- Explanation: Provide frameworks and guidelines to streamline decision-making processes, ensuring alignment with project objectives.

9. Conflict Resolution

- Objective: Address and resolve disputes among team members constructively.
- Explanation: Ensure conflicts are managed promptly and fairly to maintain harmony and productivity within the team.

10. Leadership and Guidance

- Objective: Provide direction and mentorship to the team.
- Explanation: Inspire and guide team members to achieve their goals while aligning with the overall project objectives.

The objectives of managing people and organizing teams revolve around enhancing team performance, ensuring job satisfaction, fostering collaboration, and achieving project success.

By focusing on these goals, managers can create a high-functioning team that consistently delivers quality results.

Coordination Dependencies

Coordination theory identifies and classifies common dependencies in organizational projects.

These dependencies arise from interactions between tasks, resources, and teams and must be managed effectively to ensure project success.

Here is a list of them:

- Shared Resources: Multiple tasks or team members require access to the same resource, such as equipment, software, or personnel. Conflict arises if resources are not allocated efficiently, leading to delays. Schedule resource usage to avoid overlaps and optimize availability.

- Producer-Consumer Relationships: One task produces an output (producer) that another task consumes as input (consumer). Delays in producer tasks directly affect consumer tasks. Use tools like Gantt charts or Kanban boards to track task progress and align timelines.
- Track-Subtask Dependencies: A larger task is divided into smaller subtasks that must be completed in a specific sequence. Delays in subtasks impact the completion of the main task. Break down tasks in a Work Breakdown Structure (WBS) and monitor progress.
- Accessibility Dependencies: Dependencies where team members or systems need access to shared data, tools, or systems to perform tasks. Limited or delayed access can hinder task completion. Ensure all necessary access permissions and tools are available beforehand.
- Usability Dependencies: Dependencies where the usability of one component impacts the performance of other tasks or teams. Poor usability leads to rework and delays. Implement quality assurance and usability testing before handing over deliverables.
- Fit Requirements: Dependencies where components or deliverables must fit together seamlessly to work as intended. Misalignment in specifications can lead to failures. Use detailed specifications and thorough testing to ensure compatibility.

Communication Genres and Plans

Communication Genres and Plans are essential for ensuring effective communication within a project.

They define the types of communication, the methods used, and the plan to deliver the right information to the right stakeholders at the right time.

Communication genres refer to the various modes of communication used in project management. These are categorized based on formality, medium, and purpose:

1. Verbal Communication

- Definition: Information exchange through spoken words.
- Examples: Team meetings, presentations, one-on-one discussions.
- Use Case: Quick decision-making or clarifications.

2. Written Communication

- Definition: Information shared through documents or text-based mediums.
- Examples: Emails, reports, project charters, meeting minutes.

- Use Case: Formal documentation, maintaining a record of communication.

3. Visual Communication

- Definition: Use of visual tools to convey information.
- Examples: Gantt charts, Kanban boards, infographics, dashboards.
- Use Case: Explaining complex data or tracking progress visually.

4. Digital Communication

- Definition: Communication facilitated by digital tools and platforms.
- Examples: Slack, Zoom, Microsoft Teams, project management software.
- Use Case: Remote team coordination and updates.

A Communication Plan is a document that outlines how information will be shared among project stakeholders. It ensures clear and consistent communication.

Key Components of a Communication Plan:

1. Purpose: Why is the communication needed? Example: Sharing weekly project updates with stakeholders.
2. Audience: Identify the recipients of the communication. Example: Project team, clients, senior management.
3. Method/Medium: Choose the best mode of communication for each audience. Examples: Email for formal updates, Slack for quick discussions.
4. Frequency: Define how often communication should occur. Example: Daily stand-ups, weekly progress meetings, monthly reviews.
5. Content: Specify the type of information to be communicated. Example: Status reports, milestones, risks, and changes.
6. Responsible Person: Assign responsibility for delivering the communication. Example: Project Manager sends weekly progress reports.
7. Feedback Mechanism: Include ways to gather feedback from recipients. Example: Surveys, Q&A sessions after meetings.

What is Leadership? Explain Different approaches of leadership.

Leadership is the ability to influence, inspire, and guide individuals or teams toward achieving a common goal.

Effective leadership involves setting a vision, making decisions, motivating team members, and fostering collaboration.

A leader not only manages tasks but also builds trust, resolves conflicts, and empowers others.

Different Approaches to Leadership are: -

1. Trait Approach

- Focuses on the personal traits and qualities that make an individual an effective leader.
- Key traits are Confidence, decisiveness, intelligence, integrity, and emotional intelligence.
- For e.g. A naturally charismatic individual who inspires trust and enthusiasm.

2. Behavioral Approach

- It emphasizes the behavior of the leader rather than their traits. It looks at what leaders do and how they act.
- Behavior can be task-oriented, or people-oriented.
- The task-oriented focuses on achieving goals and meeting deadlines.
- The people-oriented emphasizes the well-being and development of team members.
- For e.g. A leader who regularly conducts team-building activities to ensure collaboration.

3. Situational Approach

- Leadership style varies based on the situation and the maturity or competence of the team members.
- Situational Approach can be directive for the inexperienced or less confident teams.
- Whereas it can be supportive experienced and self-motivated teams.
- For e.g. A project manager adopting a hands-on approach with new hires but delegating tasks to senior employees.

4. Transformational Leadership

- Inspires team members by creating a vision for the future and motivating them to achieve extraordinary outcomes.

- It includes visionary thinking and focus on personal development of team members.
- For e.g. A CEO inspiring employees with a vision of innovation and sustainability.

5. Transactional Leadership

- Focuses on structured tasks, rewards, and penalties to achieve goals.
- In this leader, rewards for meeting goals whereas disciplinary actions are taken for non-compliance.
- For e.g. A sales manager offering bonuses for meeting quarterly targets.

6. Democratic Leadership

- Encourages team participation in decision-making processes.
- It builds trust and collaboration and encourages creativity.
- For e.g. A software development manager involving the team in deciding the project timeline.

7. Autocratic Leadership

- The leader makes decisions independently, without input from team members.
- It enforces fast decision-making but with lack of team involvement.
- **Example:** A military commander issuing orders during a mission.

8. Laissez-Faire Leadership

- The leader provides minimal supervision and allows the team to make decisions independently.
- It encourages autonomy and works best with highly skilled and self-motivated teams.
- For e.g. A research leader giving scientists freedom to explore innovative solutions.

Explain five fundamental stages of development

1. Project Initiation:

Initiation is the formal start of a project. It usually begins with the issue of a project mandate which briefly describes the purpose of the project and authorises budget spend.

At this stage, you should define the project at a broad level. This often begins with:

- a business case - justifying the need for the project and estimating potential benefits
- a feasibility study - evaluating the problem and determining if the project will solve it.

2. Project Definition and Planning:

Project planning is key to successful project management. This stage typically begins with setting goals. The two most common approaches include:

- the SMART method (specific, measurable, attainable, realistic and timely)
- the CLEAR method (collaborative, limited, emotional, appreciable, refinable)

At this stage, you will also define the project scope and develop a project plan and work breakdown schedule.

3. Project Execution:

Execution simply means putting your project plan into action. It often begins with a project 'kick-off meeting'.

During this phase, you will carry out the tasks and activities from your project plan to produce the project deliverables.

Project managers may direct this work by:

- overseeing a team
- managing budget and resources
- communicating to stakeholders

Careful monitoring and control at this stage can help you keep the project plan on track.

4. Project monitoring and control

- Monitoring and control often overlap with execution as they often occur at the same time.
- They require measuring project progression and performance and dealing with any issues that arise from day-to-day work.
- You can use key performance indicators (KPIs) to determine if your project is on track
- During this time, you may need to adjust schedules and resources to ensure that your project remains on track.

5. Project Closure:

- During this last phase, you will complete your work and dissolve the project.
- Closure doesn't necessarily mean success, but simply the final point of the project – e.g. closure can happen when you cancel projects that fail.
- After closure, you can carry out a post-implementation project review (sometimes referred to as a 'post mortem' meeting).
- This is an opportunity to evaluate what went well and what didn't. Understanding failures, if there were any, can help you learn lessons and improve the way you carry out future projects.

With relevant examples, differentiate between management and leadership

Management and leadership are two distinct concepts that are often used interchangeably, but they focus on different aspects of guiding and influencing people in an organization. Below is a comparison of the two, with relevant examples:

Aspect	Management	Leadership
Definition	The process of planning, organizing, and controlling resources to achieve specific goals.	The ability to inspire and motivate individuals to achieve a vision or shared objectives.
Focus	Emphasis on tasks, systems, and processes.	Emphasis on people, vision, and relationships.
Primary Role	Ensuring efficiency, consistency, and achievement of organizational objectives.	Driving change, innovation, and inspiring others toward a common vision.
Approach	Follows rules, policies, and established procedures.	Focuses on innovation and challenging the status quo.
Nature	Transactional: Deals with short-term objectives and day-to-day operations.	Transformational: Aims to bring long-term impact and inspire followers.
Decision-Making	Rational, logical, and based on data analysis.	Intuitive, visionary, and risk-taking.
Power Source	Positional authority granted by the organization.	Personal influence and charisma.
Motivational Style	Uses rewards, punishments, and authority to motivate.	Encourages intrinsic motivation through vision and trust.
Key Skills	Planning, organizing, problem-solving, and budgeting.	Communication, emotional intelligence, and vision-casting.
Outcome	Ensures stability and sustains the existing systems.	Drives change, innovation, and growth.

Examples

1. Project Execution: Management: A project manager assigns tasks, monitors progress, and ensures deadlines are met using tools like Gantt charts. Leadership: A team leader inspires the team to work passionately toward the project's success despite challenges.
2. Crisis Handling: Management: During a product recall, a manager ensures the process follows compliance guidelines and timelines. Leadership: A leader communicates transparently with stakeholders and rallies the team to focus on resolving the issue collaboratively.
3. Team Building: Management: A manager conducts regular performance reviews and assigns roles based on skills. Leadership: A leader mentors team members, fosters trust, and motivates them to grow professionally.

Management ensures the organization runs efficiently, while leadership inspires individuals to achieve a shared vision. Both are essential for an organization's success and often complement each other in achieving goals.

UNIT 6

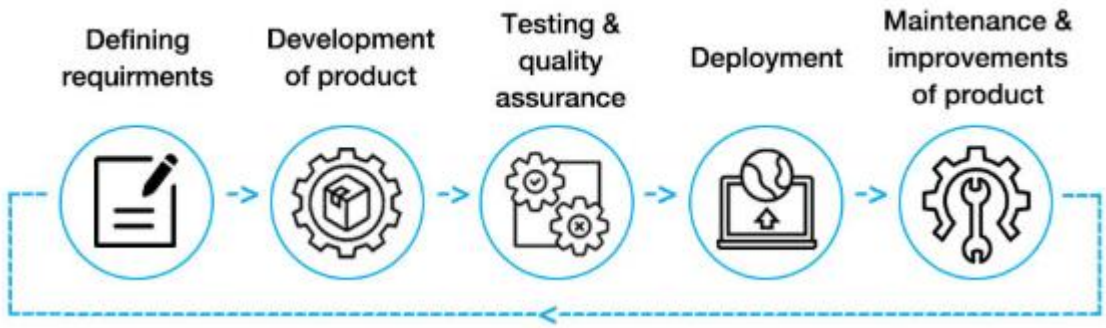
Application Lifecycle Management

Application Lifecycle Management is an integrated system of people, tools and processes that oversee a software application from its initial planning and development, through testing and maintenance, to decommissioning.

By combining and organizing lifecycle elements, ALM improves product quality, optimizes productivity and facilitates the management and maintenance of related products and services.

ALM tools automate software development and deployment processes, help ensure compliance is achieved and maintained and create a standardized environment where all teams involved in the application lifecycle can communicate and collaborate.

Application lifecycle management consists of five phases:



1. Defining Requirements:

The main objective is to collect and define the application's goals, features, and functionality.

Key Activities:

- Engaging stakeholders to identify business needs.
- Creating requirement documents and user stories.
- Prioritizing features based on business impact.

The outcome of this phase is a clear roadmap for the application’s development.

2. Development of product:

The main objective is to build the application based on the defined requirements.

Key Activities:

- Writing and reviewing code.
- Collaborating among developers, designers, and architects.
- Implementing version control for source code.

The outcome of this phase is a functional application or module.

3. Testing and Quality Assurance:

The main objective is to ensure the application meets quality and functional standards.

Key Activities:

- Conducting manual and automated testing (unit, integration, performance).
- Identifying and fixing bugs.
- Verifying that the application works as expected.

The outcome of this phase is a reliable, bug-free application.

4. Deployment:

The main objective is to deploy the application in the production environment for end-users.

Key Activities:

- Setting up continuous integration and delivery (CI/CD) pipelines.
- Monitoring deployment to identify and resolve issues quickly.

The outcome of this phase is that the application is accessible to users.

5. Maintenance & Improvements of Product:

The main objective is to ensure that the application remains relevant and efficient over time.

Key Activities:

- Monitoring the application for bugs and performance issues.
- Gathering feedback from users.
- Introducing updates, patches, and new features.

The outcome of this phase is the enhanced user satisfaction and application longevity.

The dashed arrow indicates the feedback loop, emphasizing that ALM is an iterative process. Improvements and changes from the maintenance phase often lead back to redefining requirements and subsequent cycles.

This structured approach ensures that the application evolves effectively while meeting user needs and organizational goals.

Features to Consider While Choosing ALM Tools

When selecting an ALM tool, consider the following features:

1. Integration Capabilities: Compatibility with other tools such as version control, testing, and CI/CD tools.
2. Traceability: Ability to link requirements, tasks, and tests across all phases of the lifecycle.

3. Collaboration: Facilitation of communication between cross-functional teams.
4. Customization: Options to tailor workflows and processes according to organizational needs.
5. Automation: Support for automating repetitive tasks such as testing, builds, and deployments.
6. Reporting and Analytics: Comprehensive dashboards and reports for progress tracking and decision-making.
7. Scalability: Ability to handle projects of varying sizes and complexities.
8. User-Friendliness: Intuitive interface and ease of use.
9. Security: Robust access controls and data protection measures.
10. Cloud Support: Support for cloud-based deployment and integrations.

Benefits of APM are: -

- Better overview of the workflow.
- Better compliance.
- Faster deployment.
- Higher quality products.

Some popular ALM Tools are:

1. Jama Software: Focuses on requirements management and traceability to ensure alignment between teams and stakeholders throughout the application lifecycle.
2. MeisterTask: A task and project management tool designed for agile teams, offering features like Kanban boards and team collaboration.
3. Codebeamer: Provides a comprehensive ALM solution with features like requirements management, risk tracking, and DevOps integration for regulated industries.
4. Vision: A versatile ALM tool with advanced reporting and project tracking capabilities to manage the complete software lifecycle.
5. Jira: A widely used tool for agile project management and issue tracking, offering integration with other development and testing tools.
6. Microsoft Azure DevOps: A robust ALM platform offering CI/CD pipelines, Git repositories, and project management tools for end-to-end lifecycle management.
7. Tuleap: An open-source ALM tool with support for agile, DevOps, and waterfall methodologies, providing features like version control and test management.

These tools streamline collaboration, enhance productivity, and maintain quality throughout the application lifecycle.

Azure DevOps

Azure DevOps is a software-as-a-service platform that provides DevOps practices and tools for the complete software lifecycle.

Azure DevOps is not limited to internal tools but can be integrated with most other top DevOps tools.

Below are the various services provided by Azure DevOps: -

- Azure Boards: Agile planning, work item tracking, dashboard visualization and reporting.
- Azure Pipelines: Languages and platform agnostic CI/CD with inclusion and containers and Kubernetes.
- Azure Repos: Version control system supporting Git and TFVC.
- Azure Artifacts: Integrated package management with support for Maven, NPM, Python and NuGet packages.
- Azure Test Plans: Providing an integrated testing solution including manual and exploratory.

Features of Azure DevOps Project Management

1. Agile Methodology Support: Provides tools for Scrum, Kanban, and customized workflows to track progress, sprints, and tasks effectively.
2. Work Item Tracking: Enables tracking of work items like user stories, tasks, bugs, and features with clear visibility.
3. Dashboards and Reporting: Offers real-time dashboards, charts, and analytics for monitoring project performance and progress.
4. Backlog Management: Allows prioritizing and managing backlogs with easy drag-and-drop functionalities for smooth planning.
5. Collaboration: Teams can collaborate using built-in tools like Azure Boards, comments, and pull request discussions.
6. Integration with Repos and Pipelines: Seamlessly integrates with Azure Repos, Pipelines, and Test Plans to align development and project management.
7. Customizable Workflows: Enables customization of workflows to fit unique team processes or business requirements.
8. Scalability: Supports small teams to large-scale enterprise projects with advanced tools for scaling.
9. Cross-Platform Support: Accessible from any device and integrates with non-Microsoft tools like Jira or GitHub.
10. Audit and Traceability: Ensures full traceability of work items from idea to deployment, meeting compliance needs.

These features make Azure DevOps a robust solution for project management in agile and DevOps environments.

Azure DevOps Server

Azure DevOps Server (formerly known as Team Foundation Server or TFS) is an on-premises toolset by Microsoft that provides version control, reporting, build automation, project management, and release management capabilities.

It is specifically designed for enterprises that prefer to maintain their infrastructure locally instead of using a cloud-based solution.

Features of Azure DevOps Server

1. Version Control: Supports centralized version control (TFVC) and distributed version control (Git) for tracking and managing code changes.
2. Work Item Management: Allows tracking tasks, bugs, features, and epics. It supports agile methodologies like Scrum and Kanban for project management.
3. Build and Release Management: Automates build processes, integrates continuous integration (CI), and enables continuous delivery (CD) for deploying applications across environments.
4. Test Management: Offers tools for manual and automated testing to ensure the quality of applications.
5. Integration with IDEs: Works seamlessly with Visual Studio, Eclipse, and other IDEs to improve developer productivity.
6. Reporting and Analytics: Provides detailed reports and dashboards for project status, build results, and work item tracking to facilitate informed decision-making.
7. Customizability: Teams can tailor workflows, process templates, and project management approaches to meet specific business needs.
8. Integration with Azure DevOps Services: Offers hybrid setups by integrating with the cloud-based Azure DevOps Services, enabling additional flexibility and scalability.
9. Self-Hosting: Being on-premises, Azure DevOps Server provides better control over data, especially for organizations with strict compliance or security requirements.
10. Scalability: Suitable for both small teams and large enterprises, supporting multiple projects and teams within a single instance.

Azure Repos

Azure repos is a set of version control tools that you can use to manage your code.

Azure Repos provides two types of version control:

- Git: Distributed version control.
- Team Foundation Version control: Centralized version control.

In case of centralized version control, a single copy of your project is maintained, and programmers commit their changes to this central copy.

Whereas in case of Distributed version control, every developer clones a copy of a repository and has the full history of the project.

Azure Pipelines

It uses CI/CD tool for building, testing, and deploying code automatically.

Continuous integration:

- Automatically make sure you're not sending broken code.
- Run tests continuously.
- Increase your code coverage.
- Build faster by splitting test and build runs.

Continuous delivery:

- Automatically deploy code to production.
- Make sure your deployment targets have the latest code.
- Use tested code from CI process.

Characteristics of Azure Pipeline:

- Build Azure Pipeline with two options: -
    - YAML pipeline (yaml file).
    - Classic pipeline with interface editor.
- It is workflow for automated tasks.
- It is a collection of jobs. A task is a set of tasks.
- It consists of steps called tasks, so we map scripts to tasks.

Pipeline jobs will need an agent for computing resources to run guest agents.

There are two types of agents:

Microsoft-hosted agents: -

- Maintenance and upgrades are taken care of for you.
- Fresh VM every time you run a channel.
- Windows Server with Visual Studio, Ubuntu, MAC.
- Not all build software is available in the Microsoft-hosted agent.
- For those that don't exist, in Azure Pipeline you will first need to install the software and run the jobs.

Self-service agents: -

- As a customer, you have provisioning and management to download the Azure pipelines agent and install.
- It gives your more control when installing any dependent software that is needed upfront.

- Azure pipelines can only run code without reducing runtime by not running additional software installation tasks for the build.
- It does not require a target calculation to run.

Azure Board

Azure Boards is an interactive and customizable project planning and management tool.

It is a powerful project management tool in Azure DevOps that helps teams plan, track, and discuss work across the development lifecycle.

It provides a set of features that support Agile, Scrum, and Kanban methodologies, enabling teams to manage work items, track progress, and collaborate effectively.

Features of Azure Boards: -

- Predefined work item types to track features, user stories, bugs and tasks.
- Highly interactive and visual tools.
- Easy customization.
- Built-in discussion and communication.
- Generous cloud storage
- Notification of changes.
- Built-in dashboards and analytics for health monitoring.
- Office M365 integration.
- Third party extensions.
- It is free for up to five users and unlimited stakeholders.

Example:

Consider a development team working on a Customer Management System. In Azure Boards:

- The team creates an Epic for "Customer Profile Management" and breaks it down into Features like "Create Customer Profile".
- These Features are divided into User Stories like "As a user, I want to add a customer's details".
- The team tracks progress using Kanban boards and organizes tasks into sprints, moving them from To Do to In Progress to Done.

Azure Boards provides visibility into the project's progress and ensures that work items are managed effectively.

Azure Artifacts

- Azure Artifacts enable developers and teams to share packages of code from different sources and public registers.
- Azure Artifacts supports several package types such as NuGet, E.g, Python, Maven, etc.
- Azure artifacts storage is pay-as-you-go and free up to 2 GB.

Azure test plans

Azure test plans is a browser-based test management solution that provides all the functionality needed for planned manual testing, user acceptance testing, exploratory testing and stakeholder feedback.

You can:

- Create test plans and test suites.
- Manage test plan execution settings and configuration.
- Run test on any platform with Test Runner.
- Create charts with different pivots such as priority and configuration to track test progress.
- Browse test results.
- Export test plans and test suites for review.
- Azure DevOps security and compliance.

Azure DevOps has paved the way for faster and more agile software development processes by unifying teams, processes and technologies to create a continuously evolving software development lifecycle (SDLC).

Which methodologies are supported by Azure DevOps server to implement Agile project practices?

Azure DevOps Server provides support for various agile methodologies that enable teams to efficiently manage and track their projects throughout the development lifecycle. The methodologies it supports include:

1. Scrum: Scrum in Azure DevOps helps teams work in iterative cycles called Sprints. Work items like Epics, Features, User Stories, and Tasks can be created and managed in a Product Backlog. Teams can plan sprints, track progress using Sprint Boards, and visualize work through Burndown Charts.

2. Kanban: Azure DevOps offers Kanban boards where work items move through various stages. The Kanban approach focuses on visualizing work and limiting Work in Progress (WIP) to ensure smooth workflow and continuous delivery. Cumulative Flow Diagrams provide insights into the process flow.

3. Agile: Azure DevOps supports Agile practices by allowing teams to break down large features into User Stories and Tasks. Backlog management and prioritization of work items are key components. Teams can track progress using Velocity and ensure regular delivery of value through iterative releases.

4. Lean Software Development: Lean focuses on minimizing waste and improving the flow of value. Azure DevOps supports Lean practices with Kanban boards, enabling teams to visualize and manage work. The integration with CI/CD pipelines allows for optimized delivery with minimal delays.

5. Hybrid Approach: Azure DevOps enables a Hybrid Approach by allowing teams to customize their workflows. This includes combining Scrum for sprint planning

and Kanban for ongoing work management, making it adaptable to different team needs.

6. Test-Driven Development (TDD): TDD is an agile development practice that emphasizes writing tests before writing the corresponding code to ensure the software meets its specifications and is of high quality.

In summary, Azure DevOps Server provides tools and flexibility to implement Scrum, Kanban, Agile, Lean, and hybrid methodologies, allowing teams to manage projects effectively and enhance collaboration through a unified platform.

Explain the Customizing the Process Template in Azure DevOps.

Customizing the process template in Azure DevOps allows teams to tailor the work item types, workflows, and field requirements to better align with their development processes and project needs. Azure DevOps provides several built-in process templates, such as Agile, Scrum, and CMMI, and also allows customization of these templates.

Key Steps in Customizing the Process Template:

1. Access Process Templates:
    - Go to Azure DevOps Organization settings.
    - Under Boards, select Process.
    - You'll see the default process templates: Agile, Scrum, CMMI, or a custom template if you have already created one.
2. Creating a New Process:
    - You can create a custom process by cloning an existing one (such as Agile or Scrum).
    - Once cloned, you can modify the work item types, fields, workflows, etc., according to your project's needs.
3. Work Item Type Customization:
    - Work Item Types (WITs) like User Stories, Bugs, Tasks, etc., can be modified.
    - Add or remove fields, change field types, or reorder the fields in the work item form.
    - For example, you can add a Priority field for User Stories or change the state transitions for Bug work items.
4. Field Customization:
    - You can define custom fields for different work item types. For example, adding a Department field in a Feature work item to track which department is responsible for it.

    - Set up default values, rules, or make fields mandatory as per your requirements.
5. Workflow Customization:
    - Modify the state transitions (e.g., moving from New to In Progress, and then to Done) to match your team's process.
    - You can customize the state categories and actions tied to each state.
6. Permissions and Security:
    - Configure security settings and permissions to define who can view, edit, or transition work items within the template.
    - These can be customized for specific teams or individuals.
7. Versioning and Tracking:
    - Azure DevOps allows tracking of changes made to the process templates. It is useful to maintain a history of modifications to ensure consistency across teams.
8. Integration with Other Azure DevOps Services:
    - A customized process template ensures consistency across various services like Azure Boards, Azure Pipelines, Azure Repos, etc.
    - It allows the processes defined in Azure Boards to be linked directly to the deployment pipeline in Azure Pipelines for continuous integration and delivery.

Example of Customization:

Suppose a team wants to track "Security Issues" for each work item in their backlog.

- The team could customize the Scrum Process template to include a Security Issue field in the User Story work item.
- The field could be a dropdown with options such as High, Medium, and Low. This allows team members to categorize work items based on the severity of security concerns.

Customizing the process template in Azure DevOps helps teams align the tool to their specific needs and processes, improving efficiency and ensuring that work items capture the right information.

Traceability

Traceability is the ability to track relationships between work items and artifacts across the development lifecycle.

Traceability in project management, particularly in Agile environments, refers to the ability to link and track all project artifacts, including requirements, work items, code changes, tests, and releases, throughout the development lifecycle.

Having traceability in your DevOps processes is key to successfully delivering and maintaining your applications and systems.

In Azure DevOps, traceability ensures that teams can monitor progress, identify dependencies, and ensure quality.

Traceability ensures that every requirement is addressed, tested, and delivered.

Features of Traceability in Azure DevOps: -

- End-to-End Traceability: Links between requirements, tasks, bugs, test cases, and commits.
- Hierarchical Linking: Use parent-child relationships to manage and track progress.
- Linked Artifacts: Associate work items with pull requests, builds, and releases.
- Query-Based Tracking: Create custom queries to track the status of linked items.
- Audit Trails: Maintain a history of changes to work items for accountability.

In Azure DevOps traceability helps organizations avoid common challenges across various aspects of software development, such as Work Item Tracking, TDD Testing, Azure Pipelines, Check-In Policy, and the Version Control System.

Here's how traceability resolves these issues:

1. Work Item Tracking

- Problem: Misalignment between project requirements and implementation.
- Solution:
    - Azure DevOps links work items (e.g., Epics, User Stories, Bugs) to tasks, commits, and tests.
    - Ensures all requirements are addressed and tracked through completion.
    - Avoids overlooked or incomplete work due to lack of visibility.

2. TDD (Test-Driven Development) Testing

- Problem: Difficulty ensuring all code changes are validated by corresponding test cases.
- Solution:
    - Links test cases directly to user stories or requirements.

    - Ensures all features are covered by tests, preventing untested code from being merged or deployed.
    - Supports real-time monitoring of test results linked to builds.

3. Azure Pipelines (CI/CD)

- Problem: Lack of visibility into what code is deployed in which environment.
- Solution:
    - Tracks the relationship between work items, commits, builds, and releases.
    - Provides visibility into what changes are part of each pipeline stage.
    - Prevents deployment of incomplete or unapproved features.

4. Check-In Policy

- Problem: Code is committed without linking to requirements, causing gaps in tracking.
- Solution:
    - Enforces check-in policies requiring developers to associate commits with work items.
    - Ensures every change is traceable to a specific task or bug.
    - Prevents untraceable changes that could lead to confusion or quality issues.

5. Version Control System

- Problem: Difficulty identifying which changes relate to specific requirements or bugs.
- Solution:
    - Azure DevOps integrates version control (Git or TFVC) with work items.
    - Links code changes to user stories or bugs, making it easy to trace the origin of code modifications.
    - Reduces the risk of regressions or feature mismanagement.

Benefits of Using Traceability in Azure DevOps:

- Improved Visibility: Complete transparency across the development lifecycle.
- Risk Mitigation: Early identification of missing links or unaddressed requirements.
- Compliance and Audit Trails: Maintains history for regulatory and quality checks.
- Collaboration: Teams work cohesively with clear relationships between tasks, tests, and code.

- Efficiency: Speeds up debugging, testing, and deployment processes.

By leveraging traceability, Azure DevOps helps organizations ensure robust project management and maintain alignment across all development phases, thus preventing these common pitfalls.

Visibility

Project status information is important to all project participants, and we don't just mean team members, but also stakeholders and decision makers.

As project managers, we spent to much time searching for information to answer questions about the status of projects, how much work is left and what the latest bug status is.

Visibility in DevOps refers to providing stakeholders, teams, and individuals with real-time insights into the development process, application performance, and operational workflows.

It ensures transparency across the entire lifecycle, enabling teams to identify bottlenecks, track progress, and make data-driven decisions.

Azure DevOps provides three main ways to enable visibility: -

1. Widgets and Dashboards:

- Dashboards in Azure DevOps provide a customizable and shared view of the team's progress and project health.
- Widgets display key metrics such as work item statuses, build results, or deployment progress.
- It is used to Create tailored dashboards for different stakeholders (e.g., development teams, management) and monitor real-time status of sprints, pipelines, or test results.
- Benefits:
    - Centralized view of project metrics.
    - Encourages team collaboration and accountability.
- For e.g. A dashboard showcasing sprint burndown charts, active work items, and build statuses.

2. Queries:

Queries allow users to retrieve, filter, and visualize specific work items or tasks.

It can be used to track tasks by status, priority, or ownership.

It is used to create custom queries for open bugs, unassigned tasks, or delayed items and use queries to automate alerts or trigger notifications for critical issues.

Benefits:

- Quick access to relevant data.
- Helps identify bottlenecks and pending tasks.

For e.g. Query to list all high-priority bugs across projects for the QA team.

3. Power BI:

Power BI integrates with Azure DevOps to provide advanced analytics and interactive reports.

It allows teams to build custom reports using data from work items, pipelines, or test results.

It uses Power BI dashboards for detailed reporting, such as team velocity or deployment metrics and share insights with stakeholders using interactive graphs and charts.

Benefits:

- Supports advanced data analysis.
- Enables strategic decision-making with rich visualizations.

For e.g. A Power BI report showing trends in code quality and build failures over time.

By combining these tools, Azure DevOps ensures comprehensive visibility into the entire development lifecycle.

Collaboration

Collaboration in Azure DevOps refers to enabling seamless communication and coordination among team members, ensuring everyone is aligned on project goals, progress, and challenges.

Collaboration in Azure DevOps: -

- Built-in Collaboration Tools: Azure DevOps offers tools like Azure Boards, pull requests, and work item discussions to streamline team interactions. Example: Developers can discuss code changes directly within pull requests.
- Integration with Communication Platforms: Azure DevOps integrates with tools like Microsoft Teams and Slack to provide notifications and enable discussions. Example: Automated notifications for build failures or code reviews.
- Role of Work Items: Centralized tracking of tasks, bugs, and features ensures clear ownership and accountability. Example: Assigning work items to specific team members and linking them to pull requests.
- Shared Dashboards and Reports: Dashboards and Power BI reports provide a common view of project metrics, fostering collaboration among developers, testers, and managers.
- Git Repositories: Git repositories in Azure DevOps enable collaborative coding with branching, merging, and pull requests. Example: Code reviews improve code quality and team alignment.

Extensibility

Extensibility in Azure DevOps refers to its ability to integrate with external tools and customize workflows to meet specific project needs.

Extensibility in Azure DevOps: -

- Azure DevOps Extensions: Supports extensions from the Azure DevOps Marketplace to enhance functionality. Example: Extensions for Jenkins, SonarQube, or TestRail.
- Integration with Third-Party Tools: Azure DevOps can integrate with popular tools like Jira, Selenium, Terraform, and Kubernetes. Example: Automating deployments using Terraform scripts.
- Custom Pipelines and Tasks: Build and release pipelines can include custom tasks or scripts for unique requirements. Example: Adding a custom security scan in the CI/CD pipeline.
- APIs and Webhooks: Azure DevOps APIs allow teams to automate processes and integrate with custom applications. Webhooks enable real-time event notifications to external systems. Example: Triggering a deployment in response to a Git commit.
- Custom Widgets and Dashboards: Teams can create custom widgets to display specific project data on dashboards. Example: A widget showing custom metrics like code coverage or test pass rates.
- Open-Source Support: Supports open-source tools and languages, making it versatile for diverse tech stacks.
- Scalability and Adaptability: Extensibility ensures that Azure DevOps can grow with organizational needs.

Difference Between Microsoft TFS and Azure DevOps

Feature	Microsoft TFS	Azure DevOps
Definition	A legacy on-premises tool for version control, project management, and CI/CD.	A modern, cloud-based platform offering DevOps services, supporting both cloud and on-prem.
Deployment Model	On-premises only.	Cloud-based with on-premises support via Azure DevOps Server.
Platform Accessibility	Requires installation on a server, accessible within the network.	Accessible from anywhere via the cloud.
Updates and Maintenance	Manual updates and patches required.	Automatically updated by Microsoft.

Feature	Microsoft TFS	Azure DevOps
Scalability	Limited to on-prem infrastructure capacity.	Highly scalable due to cloud-native architecture.
Integration	Limited third-party integration capabilities.	Extensive integrations via Azure DevOps Marketplace and APIs.
Version Control	Supports Git and Team Foundation Version Control (TFVC).	Fully supports Git and offers better Git-based workflows.
CI/CD Pipelines	Basic pipeline features; more complex to configure.	Enhanced CI/CD pipelines with easy-to-use YAML-based configurations.
Collaboration	Relies on basic team collaboration features.	Modern collaboration tools like Azure Boards, Wiki, and integration with Teams/Slack.
Extensibility	Limited extensibility options.	Highly extensible with APIs, extensions, and custom integrations.
Dashboards and Reporting	Basic reporting and dashboards.	Advanced dashboards, Power BI integration, and real-time reporting.
Traceability	Provides basic work item traceability.	Enhanced traceability with linking work items, code, and pipelines.
Ease of Use	Steeper learning curve due to legacy UI and features.	User-friendly interface with modern DevOps workflows.
Marketplace	Does not have an extension marketplace.	Offers an extensive Azure DevOps Marketplace for extensions and plugins.
Cost	Requires upfront licensing costs for the on-prem infrastructure.	Subscription-based pricing with pay-as-you-go for cloud usage.
Future Support	Microsoft is phasing out TFS, encouraging migration to Azure DevOps Server.	Actively supported with new features and updates.

Metrics in Agile Projects

A key performance indicator is a performance measurement used in most organizations to evaluate the success of the organization or the success of a specific activity within the organization.

KPIs are often used to measure the effects of a change project for example, implementing a good DevOps process or to evaluate the progress of a development project.

You can use DevOps online assessment scores as KPIs and compare assessment scores before and after implementing DevOps process improvements.

In this way, you will get information about whether you have improved because of implementing the new process.

You can also use reports in your DevOps toolkit during projects to see if you are continuously improving your work.

Metrics for Developer practices: -

Developer performance metrics are KPIs that can help you understand whether you're successfully working to improve your code.

These measures are useful both from an architectural and design point of view, as well as from a developer's point of view.

Some of the popular metrics used for Developer Practices are: -

1. Code Coverage

- Definition: Measures the percentage of the codebase executed during testing. It helps ensure that the critical paths and functionality of the application are tested.
- Purpose: Indicates the extent of test coverage and highlights untested code.
- Example: If 70% of the code is executed during tests, the code coverage is 70%.
- Benefit: Ensures quality by identifying gaps in testing and promoting thorough test coverage.

2. Code Metrics

- Definition: Quantitative measures of code quality and complexity, such as cyclomatic complexity, maintainability index, and lines of code.
- Purpose: Helps developers evaluate the health of the codebase.
- Example: Cyclomatic complexity measures the number of linearly independent paths in the code.
- Benefit: Encourages writing clean, maintainable, and efficient code by identifying complex or poorly structured code.

3. Compiler Warnings

- Definition: Messages generated by the compiler during code compilation, indicating potential issues or bad practices.
- Purpose: Warns about code that may work but could cause issues, such as deprecated functions or uninitialized variables.
- Example: A warning for an unused variable in the code.
- Benefit: Reduces runtime errors by addressing issues early in the development cycle.

4. Code Analysis Warnings

- Definition: Warnings generated by static code analysis tools that examine the codebase for adherence to coding standards, security vulnerabilities, and best practices.
- Purpose: Identifies potential bugs, security issues, and code smells without executing the code.
- Example: Detecting an unused method or a potential null reference exception.
- Benefit: Improves code quality and security by enforcing consistency and preventing issues before deployment.

These metrics help developers maintain a robust, secure, and high-quality codebase while minimizing defects and improving maintainability.

Metrics for software testing

Testing should be an ongoing part of any development effort, not just an end-of-project phase.

These are good metrics you can use throughout your projects to make sure you have quality testing in place.

Below are several metrics you can use as a KPI for software testing: -

1. Errors per State: Measures the number of errors identified during each phase of the software development lifecycle (e.g., design, development, testing).
2. Reactivated Errors: Tracks the number of previously resolved errors that reappear during regression testing due to incomplete fixes or new code changes.
3. Code Coverage: Represents the percentage of the codebase executed during testing, ensuring that critical code paths are adequately tested.
4. Test Run Results: Records the outcome of test executions (pass, fail, skipped) to monitor testing progress and identify potential issues.
5. Percentage of Requirements Covered by Test Cases: Calculates the proportion of requirements that have corresponding test cases to ensure complete validation of system functionality.

6. Percentage of Requirements Covered by Testing: Tracks the percentage of total requirements that have been successfully tested and validated, confirming system reliability and compliance.

Reports for Metrics in Agile Projects

1. Burn-Down Chart: Displays the remaining work in a sprint or project, helping teams track progress and predict completion dates.
2. Velocity Chart: Shows the amount of work completed during each sprint, allowing teams to assess productivity trends over time.
3. Cumulative Flow Diagram (CFD): Visualizes the flow of work items through various stages (e.g., To Do, In Progress, Done) to identify bottlenecks and ensure steady progress.
4. Defect Density Report: Highlights the number of defects identified per module or code unit, helping prioritize areas for improvement.
5. Bug status report: The bug status report gives you information about the cumulative bug count based on bug state, priority, to whom the bug is assigned and bug severity.
6. Reactivations report: The reactivations report used to determine how many bugs have been resolved or closed too early.
7. Bug trend report: Bug trend report helps you track the rate at which your team is finding, resolving and closing bugs.

Agile Project Management in Azure DevOps and TFS

Both Azure DevOps and TFS (Team Foundation Server) provide robust support for Agile Project Management, enabling teams to plan, track, and deliver work effectively.

They support Agile methodologies such as Scrum, Kanban, and hybrid approaches.

Features of Agile Project Management in Azure DevOps and TFS

1. Work Item Tracking: Teams can create and manage work items like Epics, Features, User Stories, Tasks, and Bugs. Work items are linked to track progress, maintain traceability, and connect to code changes or pipelines.
2. Boards for Agile Methodologies
    - Scrum Boards: Supports sprint planning and management, backlog tracking, and sprint reviews.
    - Kanban Boards: Provides visualization of workflows with customizable columns and WIP (Work-In-Progress) limits.
3. Backlog Management: Helps prioritize tasks and features using drag-and-drop functionality. Organizes work hierarchically (Epics → Features → User Stories) for clear alignment with Agile practices.

4. Sprints and Iterations: Teams can define and manage sprints or iterations, assigning work items to specific timeframes. Features like Burndown Charts and Sprint Goals track sprint progress and identify delays.
5. Dashboards and Reports: Real-time dashboards and built-in reports (e.g., velocity charts, cumulative flow diagrams) provide insights into project health and performance.
6. Collaboration Tools: Enables communication through comments, pull request reviews, and notifications. Integration with Microsoft Teams and Slack improves collaboration.
7. Version Control Integration: Azure Repos and TFS support Git and TFVC for managing code changes, ensuring alignment with work items and development goals.
8. Continuous Integration and Continuous Delivery (CI/CD): Integrates with Azure Pipelines to automate builds, tests, and deployments, ensuring faster and reliable delivery of features.
9. Test Management: Includes manual and automated test plans for validating requirements and maintaining quality.
10. Scalability: Supports small teams as well as large-scale enterprises, enabling effective Agile practices at any organizational level.

Scenario: Developing an Online Shopping System

- Azure DevOps:
    - Product Backlog created in Azure Boards with linked Epics, Features, and User Stories.
    - Teams use Kanban boards for workflow visualization and CI/CD pipelines for seamless deployment.
    - Real-time dashboards track sprint progress, velocity, and test results.
- TFS:
    - Teams manage product backlog and tasks locally using the Agile process template.
    - Dashboards display basic sprint progress and bug resolution status.

Azure DevOps is a more modern, flexible, and scalable solution for Agile project management, offering advanced features and cloud integration.

TFS is suitable for organizations with strict data control requirements but lacks the flexibility and seamless integrations of Azure DevOps.