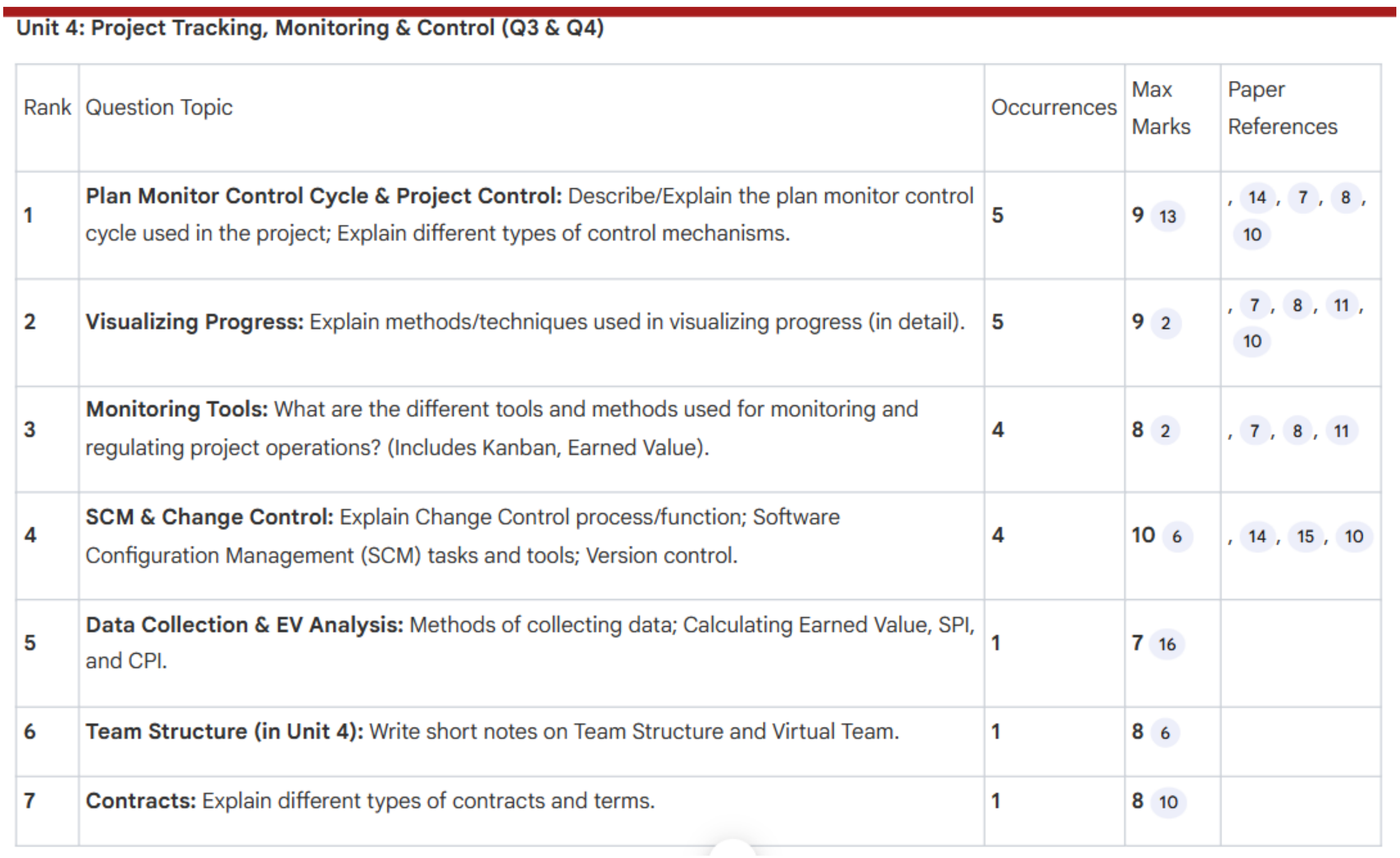
UNIT 4



Unit 4: Project Tracking, Monitoring & Control (Q3 & Q4)

Rank	Question Topic	Occurrences	Max Marks	Paper References
1	**Plan Monitor Control Cycle & Project Control:** Describe/Explain the plan monitor control cycle used in the project; Explain different types of control mechanisms.	5	9 [13]	, [14], [7], [8], [10]
2	**Visualizing Progress:** Explain methods/techniques used in visualizing progress (in detail).	5	9 [2]	, [7], [8], [11], [10]
3	**Monitoring Tools:** What are the different tools and methods used for monitoring and regulating project operations? (Includes Kanban, Earned Value).	4	8 [2]	, [7], [8], [11]
4	**SCM & Change Control:** Explain Change Control process/function; Software Configuration Management (SCM) tasks and tools; Version control.	4	10 [6]	, [14], [15], [10]
5	**Data Collection & EV Analysis:** Methods of collecting data; Calculating Earned Value, SPI, and CPI.	1	7 [16]	
6	**Team Structure (in Unit 4):** Write short notes on Team Structure and Virtual Team.	1	8 [6]	
7	**Contracts:** Explain different types of contracts and terms.	1	8 [10]	

Q1) Explain plan monitor control cycle used in the project in detail with example. **[9]**

the **Plan-Monitor-Control Cycle** is the core process of project management that ensures a project stays on track to meet its objectives. It involves a continuous loop of establishing targets, checking actual progress, and making adjustments when necessary.

1. The Plan-Monitor-Control Cycle

This cycle recognizes that it is improbable that everything will proceed exactly as planned. Therefore, the project manager must constantly reconcile the "Planned" work with the "Actual" work.

Step 1: Planning (Establishing the Baseline)

*Before execution begins, a detailed plan is created. This serves as the **baseline** against which the project will be measured,.*

- **Objective:** To define clear start/end dates, resource allocation, and budget for every activity.
- **Output:** A project schedule, cost baseline, and scope statement. This is the "Target".

Step 2: Monitoring (Data Collection)

Once execution starts, the project manager must gather data on what is actually happening.

- **Process:** This involves tracking the inputs and outputs of the project work.
- **Data Collection Methods:** Data is collected through timesheets, progress reviews, stand-up meetings, or automated project management software,.
- **Key Question:** *What has actually been achieved so far?*

Step 3: Analysis (Variance and Trend Analysis)

The collected data is compared against the original plan (baseline) to determine if there are any deviations.

- **Variance Analysis:** This looks at the difference between "Planned" and "Actual" performance. For example, comparing *Planned Value (PV)* vs. *Earned Value (EV)*.
- **Trend Analysis:** This examines project performance over time to predict future performance. It helps determine if current slippages are isolated incidents or a sign of a deeper problem.

- **Root Cause Analysis:** If a deviation exists, the manager must determine *why* it happened (e.g., technical difficulty, staff illness).

Step 4: Control (Corrective Action)

If the analysis shows that the project is drifting off course (beyond acceptable tolerance limits), the project manager must intervene.

- **Corrective Action:** This involves taking steps to bring the project back in line with the plan. This might include reallocating resources, expediting tasks (crashing), or fixing defects.
- **Re-planning:** If the delay is significant, the plan itself may need to be updated. This creates a new baseline for the remainder of the project,.

2. Suitable Example: "E-Commerce Website Login Module"

Imagine a software project to build a login feature for an online shopping system.

1. Plan (The Baseline)

- **Goal:** Complete the "User Login Module".
- **Schedule:** 10 days (Monday to next Friday).
- **Budget:** 80 hours of developer time.
- **Resources:** 1 Senior Developer.

2. Monitor (Day 5 - Friday of the first week)

- The Project Manager holds a review meeting.
- **Data Collected:** The developer reports that they have completed the database connection but are struggling with the encryption algorithm.
- **Actual Status:** Only 30% of the module is functional.

3. Analysis (The Variance)

- **Comparison:** By Day 5 (50% of the timeline), the project should be 50% complete.
- **Variance:** The project is at 30% completion. It is **20% behind schedule**.
- **Root Cause:** The encryption library selected in the planning phase is incompatible with the current server framework.

4. Control (The Action)

- **Option A (Mitigate):** The Project Manager assigns a second developer to assist with the encryption over the weekend to catch up.
- **Option B (Re-plan):** The Manager issues a **Change Request** to switch to a simpler encryption standard, which requires approval from the client,.
- **Result:** The plan is updated (re-planned) to reflect the new approach or resources, and the cycle begins again for the next week.

---

q2) What is project control? Explain the different types of control mechanism in details. [8]

Project control is a critical phase in project management that involves tracking, reviewing, and regulating the progress of a project to meet performance objectives defined in the project management plan. It occurs concurrently with project execution and ensures that the project produces the desired deliverables within the scope, time, and cost constraints.

The primary goals of project control are to:

- Track the actual progress of activities against the project schedule.
- Detect deviations (variances) where the project is drifting from the plan.
- Identify areas where project management techniques need to be changed and perform necessary modifications.
- Provide appropriate oversight and implement corrective actions to bring the project back on track.

Types of Control Mechanisms

Based on the provided sources, project control mechanisms in software project management are primarily categorized into **Progress Control (Monitoring & Variance Analysis)**, **Change Control**, and **Configuration Control**.

1. Progress Control (Variance Analysis)

This mechanism involves the continuous comparison of actual performance against the planned baseline to identify deviations.

- **Variance Analysis:** Project managers compare "Planned" values against "Actual" values. For instance, in a matrix-based cost-accounting system, every single task is given a project number to track resources and time.
- **Earned Value Analysis (EVA):** This is a key technique used to measure performance. It integrates scope, cost, and schedule measures to assess project performance and progress.
  - It calculates variances such as **Schedule Variance (SV)** and **Cost Variance (CV)** to determine if the project is ahead/behind schedule or over/under budget.
- **Corrective Action:** If the analysis shows a significant deviation (e.g., the project is slipping), the manager takes corrective action. This might involve re-planning, adding resources (crashing), or adjusting the schedule.

2. Change Control

Change control is a formal process used to ensure that changes to the product or project scope are introduced in a controlled and coordinated manner. It prevents "scope creep"—uncontrolled changes that can derail a project.

- **Change Control Board (CCB):** A group of representatives (stakeholders, developers, management) responsible for reviewing, evaluating, and approving or rejecting change requests.
- **The Process:**
  - **Identify Change:** A change is requested (e.g., by a user or developer) via a formal document known as a Change Request.
  - **Analyze Impact:** The project team estimates the effort, cost, and schedule impact of the change.
  - **Decision:** The CCB or project manager decides whether to accept, reject, or defer the change based on its value and impact.
  - **Implementation:** If approved, the plan is updated, and the work is executed.

3. Configuration Control (Software Configuration Management - SCM)

Configuration control focuses on managing the specific software artifacts (code, documents, designs) to prevent confusion caused by multiple versions.

- **Version Control:** This mechanism ensures that changes to software components are tracked. It allows developers to check-in and check-out files, ensuring that two people do not overwrite each other's work.
- **Identification:** Identifying which software items (Configuration Items or CIs) need to be controlled (e.g., source code, requirement documents).
- **Traceability:** It ensures that every version of the software can be traced back to specific requirements and changes, providing a history of what was changed, who changed it, and why.
- **Baselines:** SCM establishes "baselines"—approved versions of the software that serve as a stable basis for future development. Changes to a baseline require formal change control.

4. Risk Control

While often treated as a separate discipline, risk control is a mechanism used to minimize the impact of identified threats during execution.

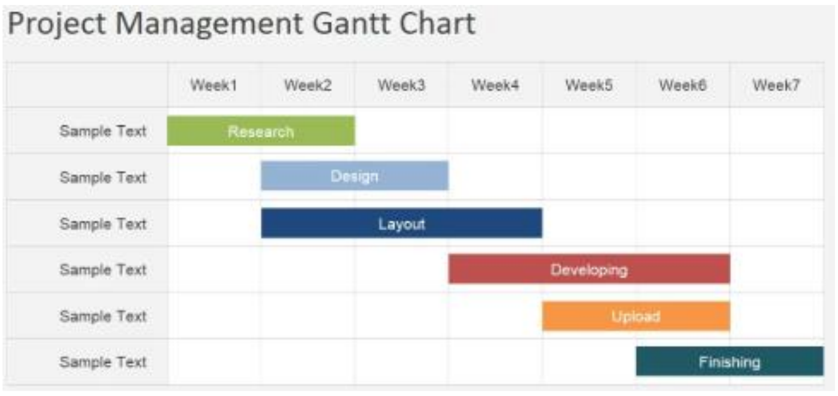
- It involves implementing **Risk Response Strategies** (such as risk mitigation or avoidance) when specific risk triggers occur.
- It utilizes a **Risk Register** to track identified risks and their status throughout the project lifecycle.

Q3) What are the different methods used in visualizing progress. Explain in detail **[9]**

Methods for Visualizing Progress

Visualizing progress involves presenting data in a format that makes the current status of the project immediately apparent. The primary methods include:

1. Gantt Chart (Bar Chart)



The Gantt chart is the most common tool for monitoring project advancement.

• **Concept:** It represents activities as horizontal bars against a time axis. The length of the bar corresponds to the duration of the activity.

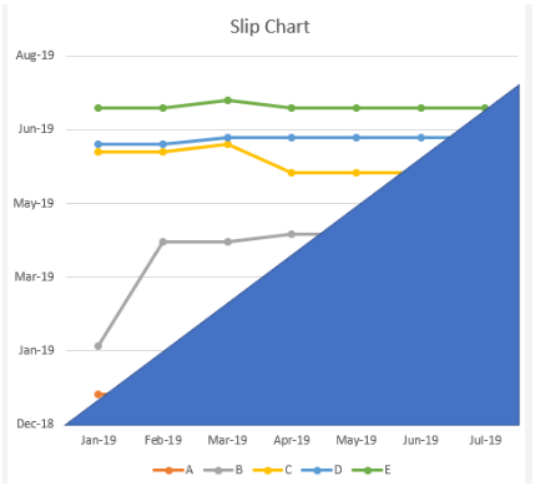
• **Visualizing Progress:** To track progress, the "Planned" schedule is often displayed alongside the "Actual" progress. A "Today Cursor" (a vertical line representing the current date) helps visualize status:

◦ If the "Actual" bar for an activity is to the left of the cursor, it is behind schedule.

◦ If it crosses or touches the cursor appropriately, it is on track.

• **Pros/Cons:** While excellent for checking if individual tasks are active, simple Gantt charts may not clearly show the impact of slippage on future tasks unless linked dynamically in software.

2. Slip Chart



A Slip Chart is specifically designed to visualize project slippage over time, which Gantt charts might obscure if not update d frequently.

• **Concept:** The vertical axis represents the activities, and the horizontal axis represents time.

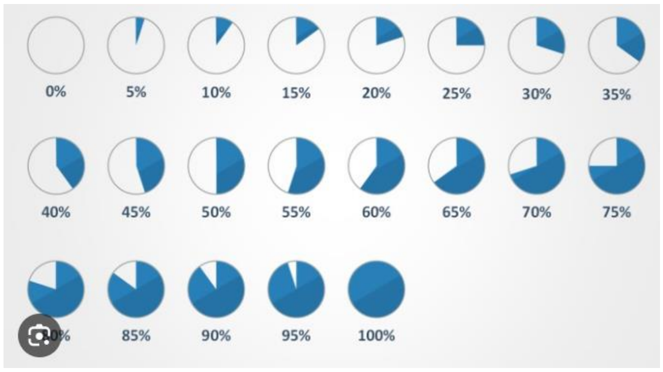
• **Visualizing Progress:** A "Slip Line" is drawn connecting the current estimated completion dates of all active tasks.

◦ **Vertical Line:** If the line is straight and vertical, the project is exactly on schedule.

◦ **Bent Line (Jagged):** The more the line bends to the right, the more the project has "slipped" (delayed). The slope of the line indicates the severity of the delay.

• **Benefit:** It provides a very jagged visual cue that suggests rescheduling is required if the slip is significant.

3. Ball Charts



This method is used to show changes in schedules for milestones or targets.

• **Concept:** It uses circles (balls) to represent milestones.

• **Visualizing Progress:**

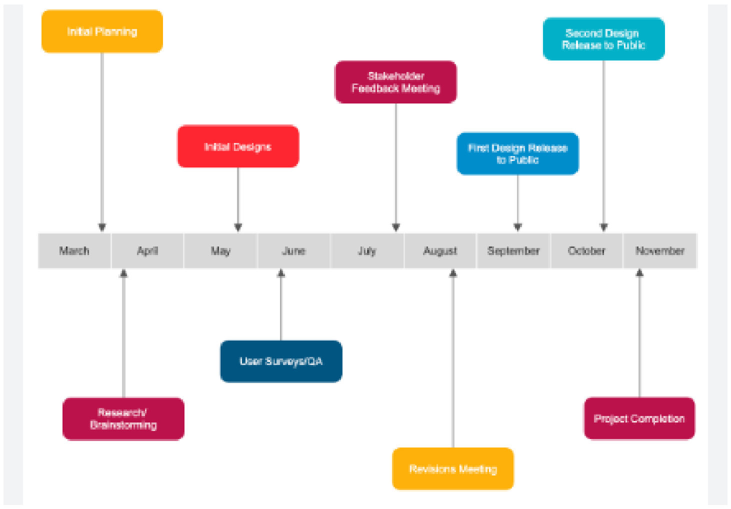
◦ A circle indicates the original planned date.

◦ If a date is revised, a second circle is drawn inside or overlapping the first.

◦ The position of the inner circle relative to the outer one indicates if the deadline has moved earlier or later.

◦ **Red/Green Status:** Sometimes, color coding (Green for on time, Red for slipped) is used within these charts to give an immediate health check.

4. The Timeline

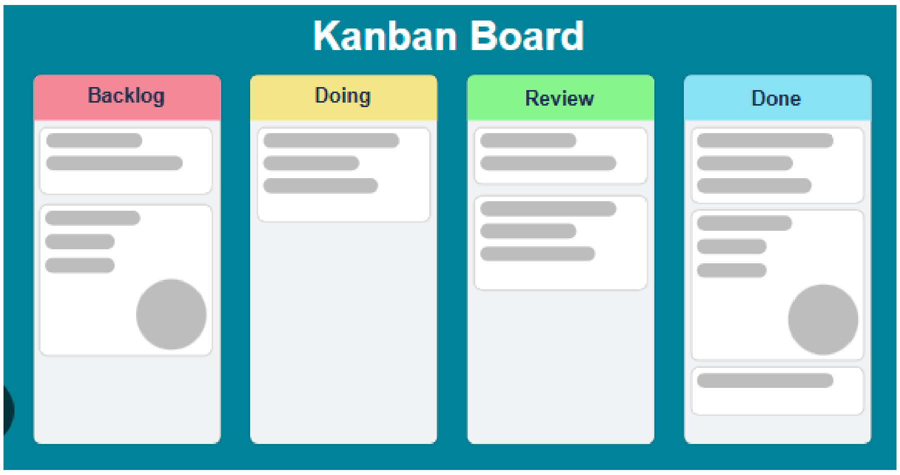


The timeline is a chart that records and visualizes targets and activities as they occur or are planned.

• **Concept:** It charts the evolution of the project schedule over time. It is particularly useful for documenting the history of the project for post-implementation reviews.

• **Visualizing Progress:** It shows what targets were set, when they were met, and how the schedule evolved week by week. It clearly displays project completion date slippage over the course of the project.

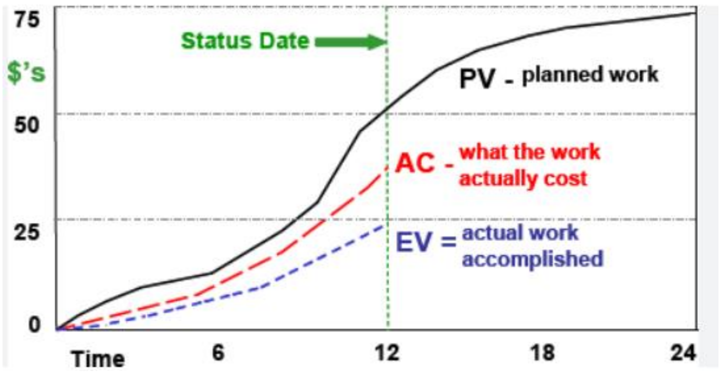
5. Kanban Boards



Kanban boards are a visual management tool used to optimize the flow of work, widely used in Agile projects.

- **Concept:** A board divided into columns representing the workflow stages (e.g., "To Do," "In Progress," "Complete").
- **Visualizing Progress:** Tasks are represented as cards. As work progresses, cards move from left to right.
- **Visual Signals:** It provides instant visibility into:
    - **Work In Progress (WIP):** Limits on how many cards can be in a column prevent bottlenecks.
    - **Bottlenecks:** A pile-up of cards in one column highlights a process issue.

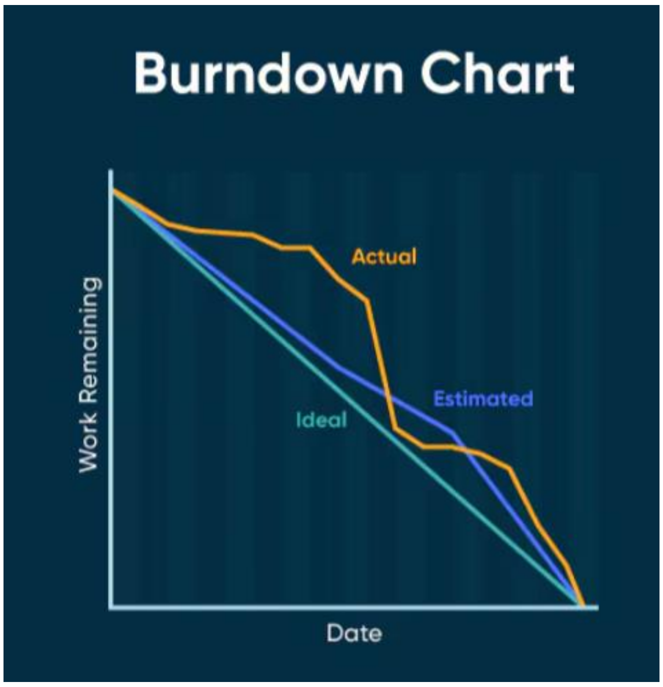
6. Earned Value Charts (S-Curve)



While primarily a quantitative technique, Earned Value Analysis (EVA) is often visualized using line graphs.

- **Concept:** It plots three key metrics against time:
    - **Planned Value (PV):** The baseline (budgeted cost of work scheduled).
    - **Earned Value (EV):** The value of work actually performed.
    - **Actual Cost (AC):** The actual cost incurred.
- **Visualizing Progress:**
    - If the **EV line is below the PV line**, the project is **behind schedule**.
    - If the **EV line is above the AC line**, the project is **under budget** (cost-efficient).
    - This visualization (often looking like an 'S' curve) gives a comprehensive view of scope, time, and cost simultaneously.

7. Burndown Charts



Common in Agile (Scrum), this chart tracks the work remaining.

- **Concept:** It plots the amount of work left (y-axis) against time (x-axis).
- **Visualizing Progress:** The "ideal trend" is a straight diagonal line down to zero. The "actual" line tracks real progress. If the actual line is above the ideal line, the team is behind schedule.

Q4) What are the different tools and methods used for monitoring and regulating project operations? **[8]**

1. Expert Judgment

Project managers and the project team utilize their collective expertise and historical data to interpret current project sta tus.

- **Method:** Stakeholders with specific knowledge assess the raw data (e.g., "we are 50% done") to determine if it is accurate and what it implies for the future.
- **Application:** It is used to decide whether to take corrective action or preventative measures based on project performance information.

2. Earned Value Technique (EVT)

This is a standard quantitative method for integrating scope, cost, and schedule measures to assess project performance.

- **Method:** It compares the **Planned Value (PV)** (what was supposed to be done) with the **Earned Value (EV)** (what was actually accomplished) and the **Actual Cost (AC)**.
- **Utility:** It allows managers to predict future schedule and cost performance, calculating indices like the Schedule Performance Index (SPI) and Cost Performance Index (CPI) to determine if the project is ahead/behind schedule or over/under budget,.

3. Project Management Information System (PMIS)

A PMIS consists of the automated tools and techniques used to gather, integrate, and disseminate the outputs of project management processes.

- **Tool:** Software packages (like Microsoft Project, Jira, or InLoox) are used to establish project plans, manage timesheets, and run reports.
- **Function:** It helps track and regulate variables like cost and resource utilization. It automatically calculates variances and provides data on how many project phases are completed versus planned,.

4. Organizational Project Management

This involves the use of established frameworks and detailed instructions provided by the organization to manage the project.
Method: It includes using standard reporting templates, specific workflow procedures, and defined control points during each stage of the project. This ensures consistency and adherence to quality standards.

5. Visualization Tools

Visualizing progress is a key method for monitoring operations, making the status immediately apparent to stakeholders.

- **Gantt Charts:** Visualizing the project schedule with horizontal bars to compare planned vs. actual progress.
- **Kanban Boards:** A tool to optimize the flow of work, moving tasks through columns (e.g., "To Do," "In Progress," "Done") to visualize bottlenecks and work-in-progress,.
- **Slip Charts & Timelines:** Specific charts designed to visualize slippage of completion dates over time,.

6. Meetings and Reviews

Regular communication is a primary method for regulating operations.

- **Status Review Meetings:** Scheduled meetings (face-to-face, online, or virtual) to exchange information about the project.
- **Stand-up Meetings:** Short daily meetings (common in Agile) where team members report what they did yesterday, what they will do today, and any blocking issues.

7. Data Collection Mechanisms

To monitor effectively, data must first be collected.

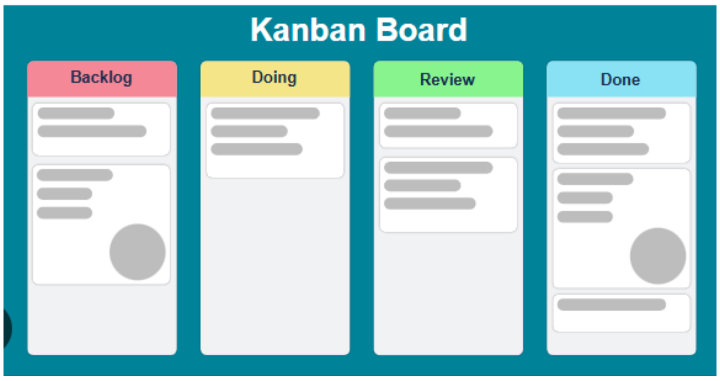
- **Timesheets:** Staff record the time spent on specific tasks, which is essential for cost monitoring.
- **Interviews:** Managers may conduct phone, online, or in-person interviews with team members to gather qualitative data on progress and potential risks.

Q5) Explain the data visualization tools: Kanban boards, project calendars, and earned value charts. What are the two main components of project tracking and control? **[7]**

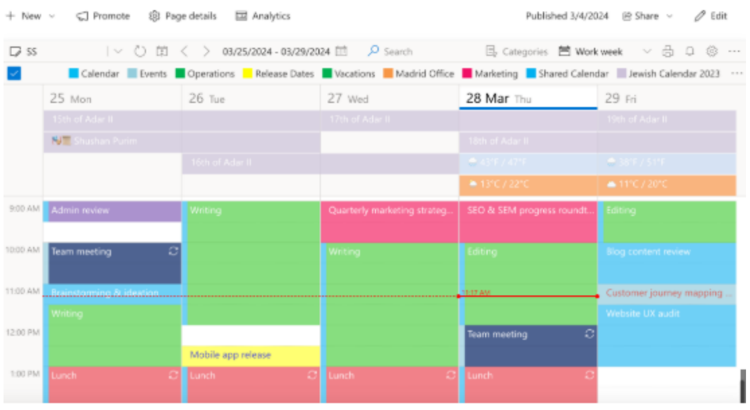
1. Data Visualization Tools

These tools are essential for making project status visible, accessible, and understandable to the project team and stakeholders.

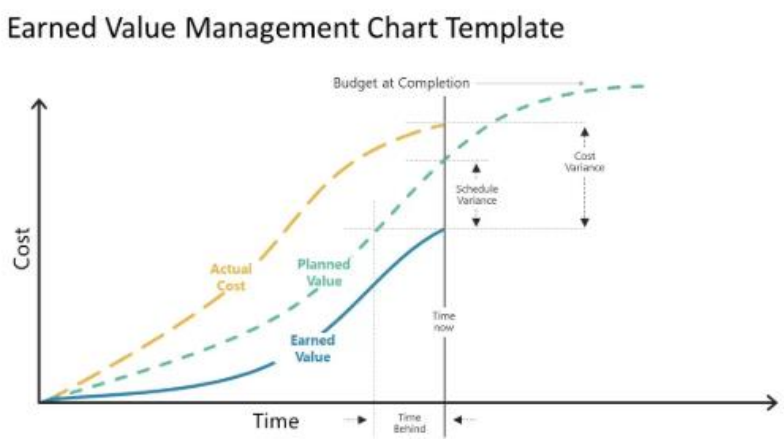
- **Kanban Boards:** A Kanban board is a tool used to visualize work and optimize the flow of the team. It typically consists of a board divided into columns representing the workflow stages (e.g., "To Do," "In Progress," "Complete").
  - **Visual Signals (Cards):** Tasks are represented as cards (stickies or tickets). Team members can see what everyone is working on at a glance.
  - **Work In Progress (WIP) Limits:** Constraints are placed on how many cards can be in a column at one time (e.g., a three-card limit). This prevents bottlenecks and ensures the team focuses on finishing current tasks before starting new ones.
  - **Commitment & Delivery Points:** It visualizes the moment a task is picked up (commitment) and when it is handed over to the customer (delivery).



- **Project Calendars:** Project calendars are used to keep track of upcoming meetings, deadlines, and milestones. They serve as a reminder of important dates and help avoid scheduling conflicts. They are generally categorized into two types:
  - **Classic Calendars:** These are "hand-made" by the user. They offer flexibility by allowing the project manager to specifically choose which tasks or events to display.
  - **Smart Calendars:** These are populated automatically using filters. They act as specific project folders that display items meeting certain criteria, such as "all tasks assigned to Team A" or "all tasks starting this week".



- **Earned Value Charts:** These charts provide a quantitative visualization of project performance by integrating scope, cost, and time measures. They typically plot three key metrics against time on a line graph:
  - **Planned Value (PV):** The budgeted cost of work scheduled (the baseline).
  - **Earned Value (EV):** The value of the work actually completed.
  - **Actual Cost (AC):** The actual expenditure incurred.
  - **Interpretation:** By comparing the lines, managers can instantly see if the project is behind schedule (EV is below PV) or over budget (AC is above EV).

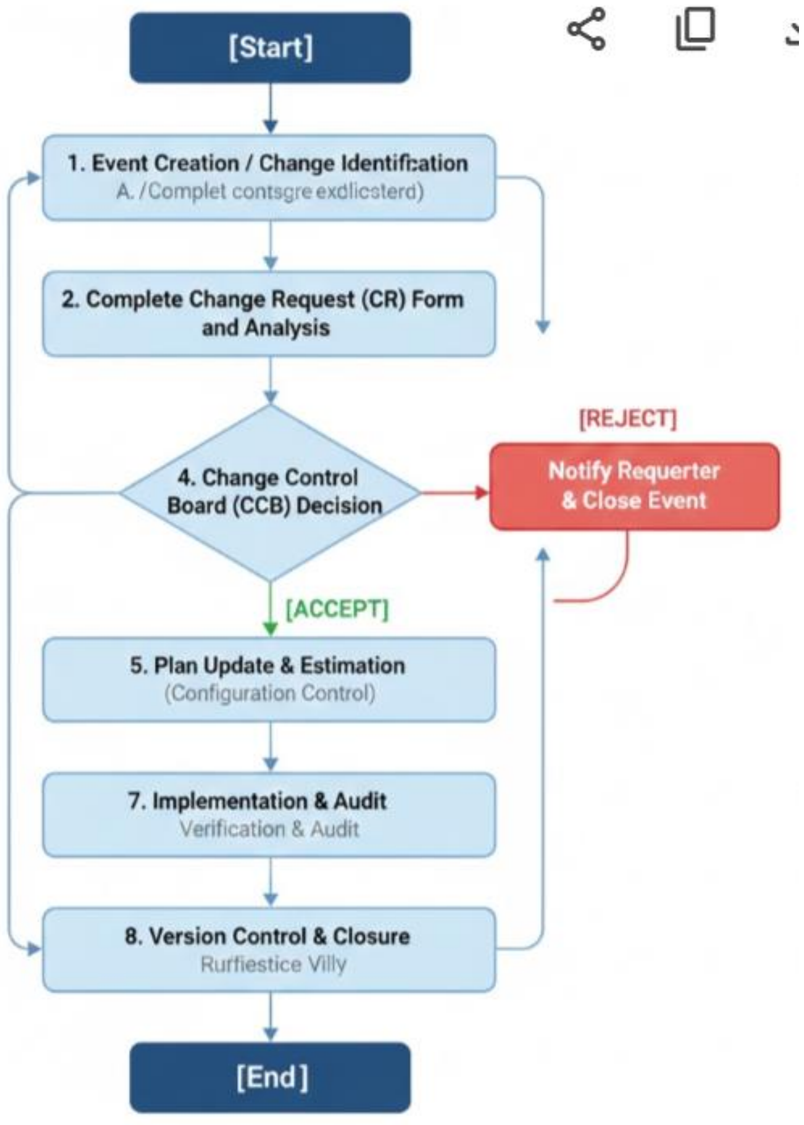


2. Two Main Components of Project Tracking and Control

Project tracking and control are concurrent processes designed to ensure the project meets its objectives. The two main components are:

1. **Monitoring (Project Tracking):** This component involves the **collection and assessment** of project data. It requires tracking the progress of activities, reviewing the status of deliverables, and gathering performance data (such as timesheets or completion percentages) to detect any deviations from the plan.
2. **Control (Project Control):** This component involves taking **corrective action**. Once monitoring detects a deviation (variance) that exceeds acceptable limits, control mechanisms are applied to bring the project back on track. This may involve re-allocating resources, adjusting the schedule, or implementing changes to the project man agement techniques.

Q6) Draw and explain the flowchart for change control function in SCM.[8]


[Start]
1. Event Creation / Change Identification
A. /Complet contsgre exdlicsterd)
2. Complete Change Request (CR) Form and Analysis
4. Change Control Board (CCB) Decision
[REJECT]
Notify Requerter & Close Event
[ACCEPT]
5. Plan Update & Estimation
(Configuration Control)
7. Implementation & Audit
Verification & Audit
8. Version Control & Closure
Rurfiestice Villy
[End]

Explanation of the Change Control Process

The Change Control function ensures that quality and consistency are maintained when changes are made to a configuration object. The process typically follows these sequential phases,,:

1. Event Creation / Initiation

The process begins when a user, developer, or stakeholder identifies a defect or a necessary enhancement.

• **Action:** The individual creates a formal request known as a **Change Request (CR)** or **Request for Change (RFC)**.

• **Details:** This document describes the existing problem or the desired feature and provides justification for the change,.

2. Event Analysis / Assessment

Once the request is logged, it must be analyzed to understand its implications.

• **Impact Analysis:** The project team or technical lead evaluates the potential impact on the project's scope, schedule, and budget.

• **Feasibility:** The team determines if the change is technically and commercially feasible. This step prevents "scope creep" by filtering out unrealistic or unnecessary changes,.

3. Decision Making (Change Control Board - CCB)

The analysis is presented to the **Change Control Board (CCB)**, a group of stakeholders responsible for approving or rejecting changes.

• **Event Acceptance:** If the CCB agrees the change is valuable and feasible, it is approved for implementation.

• **Event Rejection:** If the change is too risky, costly, or unnecessary, it is rejected, and the requester is notified.

• **Decision Factors:** The CCB considers the value of the change against the cost and risk of implementing it,.

4. Configuration of Change Request (Implementation)

If approved, the actual work begins.

• **Check-Out:** The specific Configuration Item (CI) (e.g., a source code file) is "checked out" from the repository to ensure no one else edits it simultaneously.

• **Modification:** The developer implements the required changes.

• **Update Plans:** The project plan, schedule, and cost baseline are updated to reflect the new work required,.

5. Verification and Audit

Before the change becomes permanent, it must be verified.

• **Testing:** The modified item undergoes rigorous testing to ensure the change works as intended and has not introduced new bugs (regression testing).

• **Audit:** A configuration audit may be performed to ensure the change aligns with the specifications.

6. Event Creation Closing (Closure)

Once verified, the process concludes.

• **Check-In:** The modified item is "checked in" to the repository, creating a new version or revision.

• **Baseline Update:** If the change is significant, a new **Baseline** may be established.

• **Closure:** The Change Request is formally closed in the tracking system.

----

q7) Explain version control in SCM with neat diagram. [8]

Version Control (also known as revision control or source control) is a system that records changes to a file or set of files over time so that you can recall specific versions later. It is a fundamental component of SCM that manages the different versions of software artifacts (source code, documents, designs) as they evolve during the project lifecycle.

It serves as a "content tracker" and ensures that improvements, bug fixes, and new features are integrated without overwritin g previous valid work or causing conflicts between developers.

```
+------------------------------------------------------+
|            SCM Repository / Configuration Control     |
|  (Stores Master Copies: Version 1, Version 2, etc.)   |
+------------------------------------------------------+
       |                                     ^
       | (1) Check-Out / Reserve             | (3) Check-In / Restore
       |                                     |
       v                                     |
+------------------------------------------------------+
|                 Developer's Work Space                |
|        (Private Copy for Modification/Editing)        |
+------------------------------------------------------+
       |
       +-----> (2) Modify / Test
```

Explanation of the Diagram and Process

The diagram illustrates how version control manages the flow of software items between the central repository and the developer.

1. **The Repository (Configuration Control):** The core of the system is the repository (or "Configuration Control" environment). It stores the official, authorized versions of the software artifacts (e.g., `Version 1`, `Version 2`). It acts as the single source of truth.

2. **Check-Out (Reserve):** When a developer needs to make changes, they perform a **Check-out** (or "Reserve") operation. This copies the specific file or component from the repository to the developer's private workspace. In some systems (like RCS), this might lock the file so no one else can edit it simultaneously, preventing concurrent access pro blems.

3. **Modification (Developer's Work Space):** The developer works on their private copy in their local environment. They edit, compile, and test the changes. This isolates ongoing work from the stable master version.

4. **Check-In (Restore/Commit):** Once the work is complete and verified, the developer performs a **Check-in** (or "Restore/Commit") operation. This saves the modified file back to the repository, creating a new version (e.g., `Version 3`). The system records *who* made the change, *when* it was made, and *why*.

Types of Version Control Systems

• **Centralized Version Control (e.g., SCCS, RCS):** There is a single central server. Developers check files out from this central place. If the server goes down, no one can version their work.

• **Distributed Version Control (e.g., Git):** Every developer has a full copy of the repository (including its history) on their local machine. This allows for offline work and faster branching and merging.

----

q8)) Explain the different SCM tools and how they are used to track and manage changes to software artifacts. **[10]**

1. Introduction to SCM Tools

Software Configuration Management (SCM) tools are essential for managing the evolution of software products. They allow development teams to identify, organize, and control modifications to software artifacts (such as source code, requirement documents, and designs) throughout the project lifecycle. These tools ensure that changes are recorded, versions are tracked, and valid configurations can be reconstructed at any time,.

2. Key SCM Tools

The sources highlight several specific tools used for configuratio n and version management:

A. Git

Git is a distributed version control system that is free and open-source. It is widely used to handle everything from small to very large projects with speed and efficiency.
How it works: Unlike centralized systems, every developer in Git has a full copy of the repository (including its entire history) on their local machine. This allows for offline work and faster operations.
Key Features:

- **Branching and Merging:** It supports non-linear development, allowing developers to create "branches" to work on features independently and then "merge" them back into the main code base.

- **Distributed Architecture:** Reduces dependency on a single central server.
- **Integrity:** It ensures the integrity of the code using checksums,.

B. Team Foundation Server (TFS) / Azure DevOps

Created by Microsoft, TFS (now evolved into Azure DevOps services) is a comprehensive tool that covers the entire application lifecycle.
***How it works:** It provides a set of collaborative software development tools that integrate version control, build management, and project management.*
Key Features:

- **Integration:** It links version control directly with work items (requirements, bugs), enabling traceability.
- **Flexibility:** It supports both Waterfall and Agile development methodologies.
- **Locking:** It can lock files before a user commits changes to prevent conflicts,.

C. Ansible

Ansible is an open-source software provisioning, configuration management, and application-deployment tool.
***How it works:** It uses "playbooks" (scripts) to describe automation jobs. It connects to nodes (servers) and pushes small programs called "modules" to them.*
Key Features:

- **Agentless:** It connects via SSH and does not require agent software to be installed on the remote nodes.
- **Automation:** It automates repetitive tasks, ensuring that infrastructure configuration remains consistent across different environments,.

D. SCCS (Source Code Control System) & RCS (Revision Control System)

These are older, foundational version control tools mentioned in the sources to explain the concept of "deltas."
***SCCS:** Stores the original file and keeps track of changes (deltas) sequentially.*
***RCS:** Stores the latest version of the file and keeps reverse deltas (changes needed to go back to previous versions). This makes accessing the latest version faster,.*

3. How Tools Track and Manage Changes

SCM tools utilize specific mechanisms to track and manage changes to software artifacts effectively:

1. Version Control and Repositories

- **Repository:** The tools maintain a central (or distributed) database called a repository that stores the "Master" copies of all artifacts.
- **Check-In/Check-Out:** To make a change, a developer "checks out" (or clones) an artifact to their workspace. Once modified and tested, they "check in" (commit) the file back to the repository. The tool records *who* made the change, *when* it was made, and *why*,.

2. Handling Concurrent Access

SCM tools prevent the "overwrite" problem where two developers modify the same file simultaneously.

- **Locking:** Some tools lock a file when one user checks it out, preventing others from editing it until it is checked back in.
- **Merging:** Modern tools (like Git) allow simultaneous editing but require a "merge" process to reconcile differences. If conflicts occur (e.g., both modified the same line), the tool alerts the user to resolve them manually,.

3. Baselining

*Tools allow managers to create a **Baseline**—a snapshot of the software at a specific point in time (e.g., "Release 1.0").*

- Once a baseline is established, it serves as a stable reference point. Changes to a baseline typically require a formal **Change Control** process to ensure stability is not compromised,.

4. Traceability and Auditing

- **Traceability:** Tools link code changes to specific requirements or change requests (CRs). This ensures that every line of code exists to satisfy a specific need.
- **Auditing:** SCM tools provide logs and reports that allow auditors to verify that the software matches the documentation and that unauthorized changes have not been introduced,.

----

q9) What is Software Configuration Management? Explain Tasks in SCM Process. [9]

Software Configuration Management (SCM) is the art of identifying, organizing, and controlling modifications to the software being built by a development team. It is an "umbrella activity" that is applied throughout the software process.

- **Definition:** SCM is a set of activities designed to control change by identifying the work products that are likely to change, establishing relationships among them, defining mechanisms for managing different versions of these work products, controlling the changes imposed, and auditing and reporting on the changes made.
- **Goal:** The primary goal is to maximize productivity by minimizing mistakes and ensuring that the software product evolves in a controlled manner, maintaining integrity and traceability throughout the project lifecycle.

Tasks in the SCM Process

The SCM process involves five distinct tasks that ensure changes are managed effectively. These tasks are:

1. Configuration Identification

This is the foundational task of SCM. Before a system can be controlled, its components must be identified.
Objective: *To determine the scope of the software system and identify which items (Configuration Items or CIs) need to be placed under SCM control.*
Activities:

- Identifying **Computer Software Configuration Items (CSCIs)**, which can include source code, requirements specifications, design documents, test cases, and user manuals.
- Assigning a unique identifier (name and number) to each CI so they can be tracked distinguishably.
- Defining the versioning scheme (e.g., Version 1.0, Version 1.1) to manage the evolution of these items.

2. Baselines

A baseline is a milestone in the software development consisting of a set of configuration items that have been formally reviewed and agreed upon.
Objective: *To serve as a stable basis for further development.*
Significance: *Once a baseline is established (e.g., "Design Baseline" or "Release 1.0"), it can only be changed through a formal* ***Change Control*** *procedure. It represents a "snapshot" of the project at a specific point in time.*

3. Change Control

This task ensures that changes to baselines and configuration items are made in a controlled and authorized manner.
Objective: *To prevent "scope creep" and ensure that changes are technically and commercially feasible before implementation.*
Process:

- **Change Request:** A user or developer submits a request for a change (to fix a bug or add a feature).
- **Evaluation:** The project team analyzes the impact of the change on the schedule, budget, and other system components.
- **Approval/Rejection:** A **Change Control Board (CCB)** reviews the request and decides whether to approve, reject, or defer it.
- **Implementation:** If approved, the CI is "checked out," modified, tested, and then "checked in" as a new version.

4. Configuration Status Accounting (CSA)

CSA involves recording and reporting the status of configuration items and change requests.

Objective: *To provide visibility into the configuration of the system. It answers questions like: What changes have been made? Who made them? When were they made?*

Activities:

- Tracking the status of Change Requests (e.g., "Pending," "Approved," "Completed").
- Generating reports that list the current version of all CIs in the system.
- Ensuring that all team members are working with the correct versions of the software.

5. Configuration Audits and Reviews

This task verifies that the software product matches the configuration information.

Objective: *To ensure integrity and correctness of the release.*

Types of Audits:

- **Functional Configuration Audit (FCA):** Verifies that the software functions (performance, features) match the requirements specified in the documentation.
- **Physical Configuration Audit (PCA):** Verifies that the actual software artifacts (code, design docs) exist and are the correct versions referenced in the configuration records.

Q10) b) Discuss the different change request types and the change approval process. **[10]**

1. Types of Change Requests

Change requests are formal proposals to modify any document, deliverable, or baseline. In software project management, they typically fall into the following categories, which "pave the way for the gathering and documenting of additional requirements":

- **Corrective Action:** This is a deliberate activity ensuring that the future performance of the project work aligns with the project management plan. If a deviation is detected during monitoring (e.g., a task is running late), a corrective action request is made to realign the project with the schedule or budget.
- **Preventive Action:** This involves an intentional activity that ensures the future performance of the project work ensures the project protects itself from potential risks. These requests are proactive measures taken to reduce the probability of negative consequences associated with project risks.
- **Defect Repair:** This is an activity to modify a nonconforming product or component. If a bug or issue is found in the software (e.g., a "loose" database connection or a UI glitch), a request is generated to fix the defect so the product meets quality requirements.
- **Scope Changes (Updates):** These requests involve changes to the formally controlled project documentation or plans, often driven by new requirements from the client. For example, an "upgrade to a new version of the middleware" or adding a feature to support a legacy system. These changes often require adjusting the project scope, cost, or schedule baselines.

2. The Change Approval Process

The change approval process, often managed by a **Change Control Board (CCB)**, ensures that no change is implemented without reviewing its impact on the project's time, cost, and quality. The process typically follows these sequential steps,, :

Step 1: Initiation (Propose Change)

The process begins when a stakeholder, developer, or client identifies a need for a change.

- **Action:** The change is formally documented on a **Change Request (CR)** form.
- **Details:** The form includes a description of the change, the reason for it (justification), and its urgency. This ensures the request is "well-crafted" and clearly communicated to the project team.

Step 2: Assessment (Impact Analysis)
Once initiated, the project team analyzes the request to understand its full implications.

- **Impact Study:** The team estimates the effort, time, and cost required to implement the change. They evaluate how it affects other parts of the system (dependencies) and the overall project schedule,.
- **Feasibility:** The team determines if the change is technically possible and commercially viable.

Step 3: Decision Making (CCB Review)
The analysis is presented to the ***Change Control Board (CCB)*** *or the project manager/sponsor.*

- **Review:** The decision-makers weigh the benefits of the change against its costs and risks.
- **Outcome:**
  - **Accept:** The change is approved for implementation.
  - **Reject:** The change is deemed unnecessary or too risky/expensive.
  - **Defer:** The change is valid but will be considered for a future release.

Step 4: Implementation (Executing the Change)
If approved, the change becomes a "Change Order".

- **Plan Update:** The project management plan, schedule, and cost baselines are updated to reflect the new work.
- **Execution:** The development team checks out the necessary configuration items (e.g., source code), implements the change, and performs unit testing.

Step 5: Verification and Closure
Before the change is officially closed, it must be verified.

- **Testing:** Quality assurance teams test the modification to ensure it works as intended and hasn't introduced new bugs (regression testing).
- **Closure:** Once verified and accepted by the client or product owner, the Change Request is formally closed in the tracking log, and a post-mortem or regression test might be conducted to ensure stability.

----

q11) What are common methods of collecting project data from stakeholder. Explain how to calculate earned value and use it to calculate Schedule Performance Index (SPI) and cost performance index (CPI). **[7]**

Common Methods of Collecting Project Data
Project managers must gather data to track progress, monitor risks, and ensure quality. The sources identify several methods for collecting this data from stakeholders and systems:

1. **Interviews:** Interviews are a primary method for qualitative data collection. They can be conducted in various formats:

◦ **In-Person Interviews:** These provide high confidence in the data but are time-consuming and expensive.

◦ **Phone Interviews:** A faster alternative to face-to-face meetings.

◦ **Online/Web Surveys:** These are cost-effective and allow for self-management by the respondent, though they may suffer from lower response rates or "check-box" answers.

2. **Questionnaires and Surveys:** These are fundamental instruments for gathering information, especially when dealing with a large number of stakeholders or when statistical analysis is required. They utilize specific schedules and structured questio ns to ensure consistency.

3. **Timesheets:** To monitor costs and schedule adherence, staff record the time spent on specific tasks. This is a standard method for tracking resource utilization and calculating actual costs,.

4. **Delphi Technique:** This is a specialized method used to gather consensus from a panel of experts. It involves multiple rounds of anonymous questioning to converge on a reliable estimate or opinion, often used for risk assessment or complex estimation.

5. **Automated Data Collection:** In modern IT projects, data is often gathered automatically through transaction records, website visits, mobile applications, and sensors. This allows for the collection of large datasets ("Big Data") without manua l intervention.

6. **Status Reporting (RAG Reporting):** Stakeholders (like team members) provide status reports, often using a "Traffic Light" system:

- **Red:** Significant issues/delays.
- **Amber:** Recoverable issues.
- **Green:** On target.

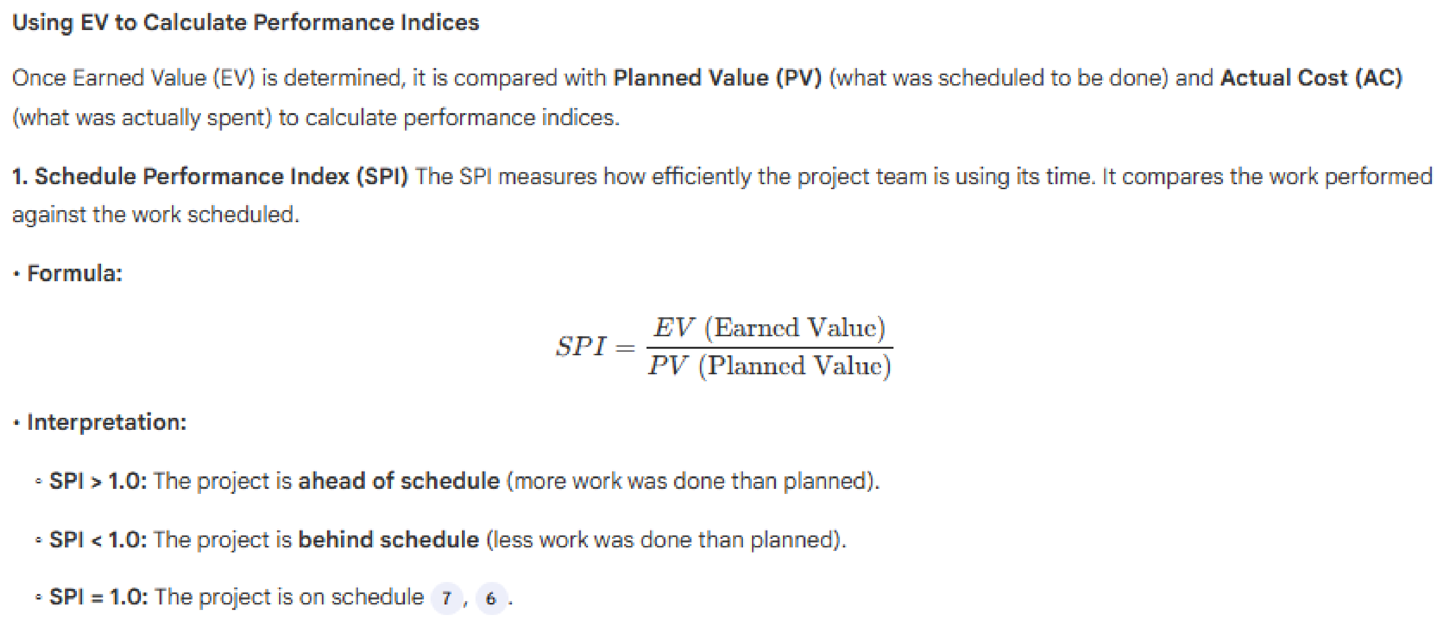
Earned Value Analysis (EVA)

Earned Value Analysis is a quantitative technique used to measure project performance by integrating scope, cost, and time measures. It compares what was planned against what was actually accomplished.

How to Calculate Earned Value (EV)

Earned Value (EV), also known as the **Budgeted Cost of Work Performed (BCWP)**, represents the value of the work that has actually been completed at a specific point in time.

- **Calculation Method:** To calculate EV, you sum the budgeted cost for all work packages or activities that have been completed.
  - **Example:** If a task has a budget of $1,000 and is 100% complete, the EV is $1,000. If it is 50% complete, the EV is $500.
- **Techniques for Assigning Value:** Common methods for crediting EV include:
  - **0/100 Technique:** EV is 0 until the task is finished, then it becomes 100% of the budget.
  - **50/50 Technique:** 50% of the value is credited when the task starts, and the remaining 50% upon completion.



Using EV to Calculate Performance Indices

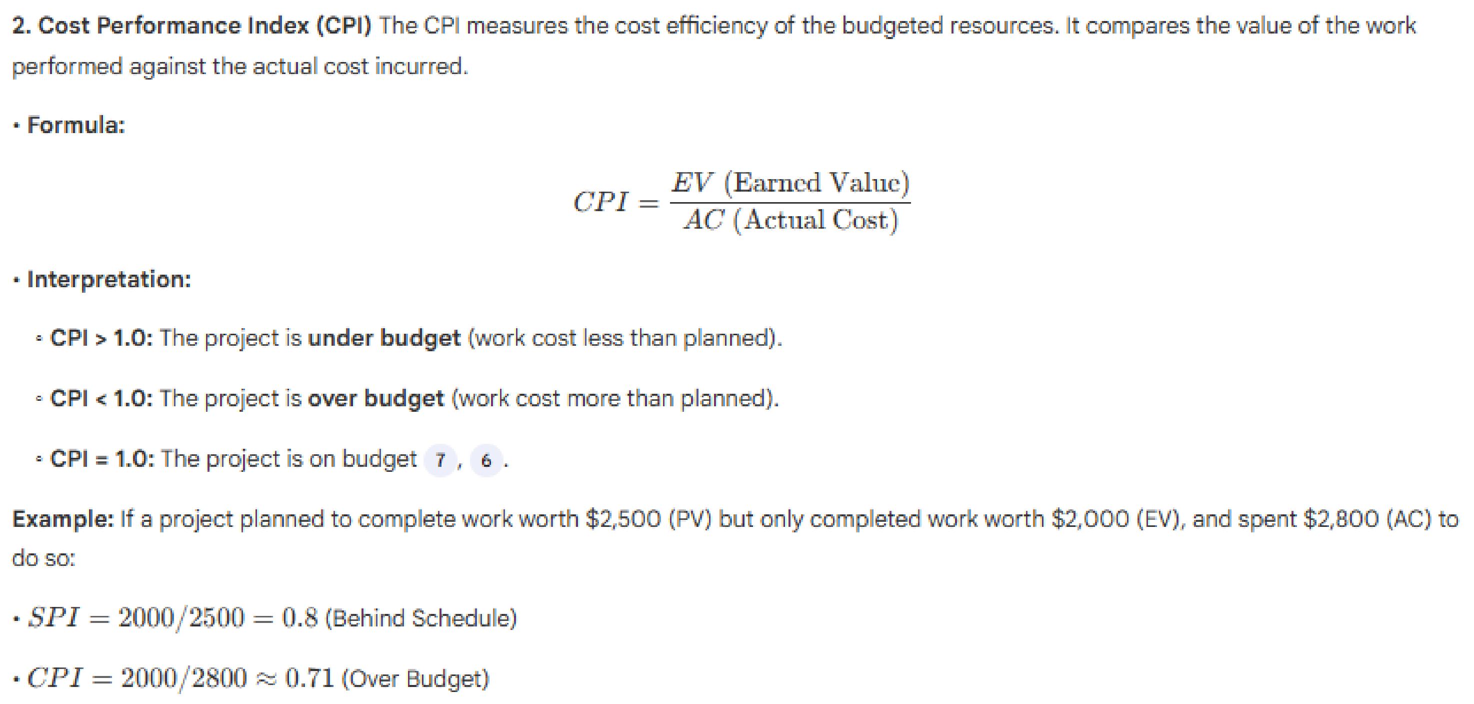
Once Earned Value (EV) is determined, it is compared with **Planned Value (PV)** (what was scheduled to be done) and **Actual Cost (AC)** (what was actually spent) to calculate performance indices.

1. Schedule Performance Index (SPI) The SPI measures how efficiently the project team is using its time. It compares the work performed against the work scheduled.

- **Formula:**

$$SPI = \frac{EV\ (\text{Earned Value})}{PV\ (\text{Planned Value})}$$

- **Interpretation:**
  - **SPI > 1.0:** The project is **ahead of schedule** (more work was done than planned).
  - **SPI < 1.0:** The project is **behind schedule** (less work was done than planned).
  - **SPI = 1.0:** The project is on schedule [7], [6].



2. Cost Performance Index (CPI) The CPI measures the cost efficiency of the budgeted resources. It compares the value of the work performed against the actual cost incurred.

- **Formula:**

$$CPI = \frac{EV \text{ (Earned Value)}}{AC \text{ (Actual Cost)}}$$

- **Interpretation:**
  - **CPI > 1.0:** The project is **under budget** (work cost less than planned).
  - **CPI < 1.0:** The project is **over budget** (work cost more than planned).
  - **CPI = 1.0:** The project is on budget [7], [6].

Example: If a project planned to complete work worth $2,500 (PV) but only completed work worth $2,000 (EV), and spent $2,800 (AC) to do so:

- $SPI = 2000/2500 = 0.8$ (Behind Schedule)
- $CPI = 2000/2800 \approx 0.71$ (Over Budget)

---

q12) Write short notes on Team Structure and Virtual Team in a Software Project. [8]

1. Team Structure

Team structure refers to how individual team members are organized, controlled, and coordinated to achieve project goals. The structure determines the flow of authority, communication channels, and decision-making processes.

A. Organizational Structures The broader context in which a team operates usually falls into one of three categories:

- **Functional Format:** Staff are grouped by specialty (e.g., a department of only programmers). While this fosters technical expertise, it can lead to a lack of focus on specific project deadlines,.
- **Project Format:** Staff are dedicated to a specific project for its duration. This ensures a strong focus on project goals, but team members may feel isolated from their technical peers and worry about job security after the project ends,.
- **Matrix Structure:** A hybrid approach where staff report to two bosses: a Project Manager (for day-to-day work) and a Functional Manager (for technical development and administration). This maximizes resource utilization but can cause conflict due to dual command,.

B. Specific Team Models Within a project, the team itself can be structured in different ways:

1. **Chief Programmer Team:** This is a highly centralized structure where a "Chief Programmer" acts as the technical lead and primary decision-maker. They are supported by specialists (like a librarian or backup programmer). It is efficient for specific tasks but relies heavily on one individual, creating a single point of failure,.
2. **Democratic (Egoless) Team:** A decentralized structure where decisions are made by consensus. Leadership may rotate depending on the task. The code is considered "public" property of the team (egoless), encouraging peer reviews and shared responsibility. It is good for solving complex problems but can be slow due to the need for consensus,,.
3. **Mixed Control Team:** A combination of the above, often used in large projects. It typically involves a hierarchy where a project manager oversees the timeline, but technical decisions are delegated to senior members or sub-teams.

2. Virtual Team (Dispersed Team)

A Virtual Team (often referred to as a **dispersed team** in the sources) consists of members who are geographically separated and collaborate primarily through electronic communication rather than face-to-face interaction.

Key Characteristics:
Geographical Dispersion: Team members may be located in different offices, cities, or countries. They work across different time zones and spaces.

Reliance on Technology: Collaboration relies on email, video conferencing, shared repositories, and instant messaging to coordinate work.

Benefits:
Global Talent Pool: Organizations can hire the best people regardless of their physical location.
"Follow the Sun" Development: Work can continue 24/7. As one team finishes their day in one time zone, they hand off work to a team starting their day in another, potentially reducing project duration.
Cost Reduction: It reduces overhead costs associated with physical office space and relocation.

Challenges:
Communication Issues: The lack of "face-to-face contact" can lead to misunderstandings. Non-verbal cues are lost, and building trust is more difficult,.
Coordination Complexity: Managing different time zones and cultures requires significant effort. "Out of sight, out of mind" can be a risk where remote members feel disconnected,.
Trust and Supervision: Managers cannot physically oversee work (observation is difficult), so trust and output-based management become critical.

---

q13) What is Project Risk Management? What are the RM Processes? [9]
Project Risk Management is the process of identifying, analyzing, and responding to risk factors throughout the life of a pro ject and in the best interests of its objectives. It involves minimizing potential issues that can adversely affect a project's sc hedule, cost, or quality,
Definition: It is often described as "the process of minimizing any potential issues that can adversely affect a project's schedule is known as risk management". It seeks to balance risk and r eward, managing the feasibility of the project.
Key Elements: Risk is characterized by two main elements:

- **Uncertainty:** The risk may or may not happen (probability < 100%).
- **Loss:** If the risk becomes a reality, it results in unwanted consequences or damage,.

The Risk Management (RM) Processes

The Risk Management process provides a framework for dealing with risks. It is a continuous cycle rather than a one-time activity. The process typically consists of the following sequential stages,:

1. Risk Identification

This is the first step where the project team identifies the hazards and potential problems that could threaten the project.
Goal: *To generate a comprehensive list of risks.*
Techniques:

- **Checklists:** Using lists of risks found in previous similar projects.
- **Brainstorming:** Group sessions where stakeholders generate ideas about potential failure points.
- **Domain Analysis:** Identifying risks specific to the business domain.

2. Risk Assessment (Analysis)

Once identified, risks must be analyzed to understand their nature and severity.
Goal: *To estimate the **Probability** (likelihood of occurrence) and the **Impact** (potential damage/loss) of each risk,.*

- **Tools:**
  - **Probability Impact Matrix:** A grid used to map risks based on their likelihood and impact (e.g., High, Medium, Low) to determine their magnitude.
  - **Risk Exposure:** Calculated as `(Potential Damage) x (Probability of Occurrence)`.

3. Risk Prioritization

After analysis, risks are ranked to determine which ones require immediate attention.
Goal: *To focus resources on the most critical risks.*

- **Categories:**

    - **High Risk:** Significant impact; immediate action required.
    - **Medium Risk:** Noticeable impact; requires monitoring and management.
    - **Tolerable/Low Risk:** Little impact; might be accepted with minimal action.

4. Risk Planning

This stage involves deciding how to respond to the major risks identified and prioritized. The primary strategies include,,:

- **Risk Acceptance:** Doing nothing and accepting the consequences if the risk occurs (often for low-priority risks).
- **Risk Avoidance:** Changing the project plan or scope to eliminate the risk entirely (e.g., using a proven technology instead of a new, experimental one).
- **Risk Reduction (Mitigation):** Taking actions to lower the probability or impact of the risk (e.g., implementing backups to reduce data loss risk).
- **Risk Transfer:** Shifting the responsibility to a third party (e.g., buying insurance or outsourcing a risky component).

5. Risk Monitoring and Control

This is an ongoing process of tracking identified risks, monitoring residual risks, identifying new risks, and evaluating the effectiveness of risk response plans.
Goal: To ensure that risk plans are executed correctly and to adjust strategies as the project evolves,.
Activities: Regular risk reviews, audits, and updating the **Risk Register** (a document recording all identified risks and their status).

---

q14) Explain the different types of contracts? List the various typical terms of a contract? [8]

1. Different Types of Contracts

Contracts in software project management generally fall into a few primary categories based on how the price is determined and how risk is distributed between the buyer and the supplier:

1. **Fixed Price Contracts:**
    a. **Description:** In this type of contract, the buyer and the service provider agree on a total fixed price for the requested services or product at the beginning. The price remains constant regardless of the actual effort or cost incurred by the vendor.
    b. **Usage:** It is best suited for projects where the scope and requirements are clearly defined and unlikely to change.
    c. **Risk:** The risk is primarily on the **vendor**. If the project takes longer or costs more than estimated, the vendor's profit margin decreases,,.
2. **Time and Materials Contracts:**
    a. **Description:** The buyer agrees to pay the contractor based on the time spent (e.g., hourly or daily rates) and the materials used. The total cost is not fixed in advance.
    b. **Usage:** This is often used when the scope of work is difficult to define upfront or is expected to change significantly. It allows for flexibility.
    c. **Risk:** The risk is primarily on the **buyer (customer)**, as the total cost can escalate if the project takes longer than anticipated,,.
3. **Fixed Price per Delivered Unit Contracts:**
    a. **Description:** The price is calculated based on a fixed rate per unit of work delivered. Common units in software might include Function Points (FPs) or Lines of Code (LOC).
    b. **Usage:** This requires a reliable method for estimating the size of the software (e.g., counting function points).
    c. **Risk:** The risk is shared. The price per unit is fixed, but the total number of units might vary depending on the final scope,.
4. **Cost Reimbursable Contracts:**
    a. **Description:** The buyer reimburses the seller for the actual allowable costs incurred in performing the work, plus an additional fee representing the seller's profit.
    b. **Variations:**
        i. **Cost Plus Fixed Fee:** The seller receives actual costs plus a fixed fee.
        ii. **Cost Plus Incentive Fee:** The seller receives actual costs plus a fee that is adjusted based on meeting certain performance objectives (e.g., finishing early).
    c. **Risk:** The risk is largely on the **buyer**, as they must cover all legitimate costs,.

2. Typical Terms of a Contract

A software contract typically includes several standard sections and terms to define the relationship and obligations of both parties. Key terms include,:

1. **Definitions:** Clarifies the terminology used in the document to avoid ambiguity.
2. **Form of Agreement:** Specifies the nature of the deal, such as whether it is a lease, a license, or a sale.
3. **Goods and Services to be Supplied:** Detailed description of the hardware, software, documentation, and training to be provided.
4. **Ownership of Software:** Defines who retains the intellectual property rights (copyright) to the software—whether it remains with the supplier or transfers to the client.
5. **Environment:** Specifies the technical environment (hardware, OS) in which the software must operate.
6. **Acceptance Standards:** The criteria that the software must meet for the client to accept the deliverable and authorize payment.
7. **Time Table:** The schedule of deliverables, milestones, and the final deadline.
8. **Price and Payment Method:** The total cost or rate structure and the schedule for payments (e.g., upon completion of milestones).
9. **Legal Requirements:** Clauses covering applicable laws, dispute resolution, and jurisdiction.
