**SECTION A**

**Very Short Answer Questions**

**Attempt all questions (7\*2=14)**

**1. What are the characteristics which make software projects different from other projects?**

* It has a unique and definitive purpose.
* It is temporary in nature.
* It is developed using progressive elaboration
* It almost always involves uncertainty
* It requires resources - often of various kinds and from various sectors

**2. What is the significance of the critical path?**

Critical path for a project is the series of activities that determines the earliest time by which the project can be completed. Identifying the critical path helps project managers focus on key tasks, manage resources effectively, and ensure the project stays on schedule.

**3. Explain in brief sunk cost with proper example.**

Sunk cost refers to money that has already been spent and cannot be recovered, regardless of future outcomes. It should not influence decision-making since it cannot be changed.

Example: If a company invests $50,000 in developing a software project but later realizes it won't be profitable, the $50,000 is a sunk cost.

**4. How does software inspection differ from software testing?**

**Software inspection** is a static process where code, design, or documents are manually reviewed to detect defects without executing the software. It focuses on finding issues like logic errors, coding standards, or design flaws early in development.

**Software testing**, on the other hand, is a dynamic process that involves executing the software to identify bugs or performance issues under various conditions, validating that the software behaves as expected

**5. Define MOI model of leadership as defined by Weinberg.**

The **MOI model of leadership**, as defined by Gerald Weinberg, emphasizes three essential elements for effective leadership:

1. **Motivation (M)**: The ability to inspire and motivate team members to achieve common goals and perform at their best.
2. **Organization (O)**: Structuring tasks, resources, and processes efficiently to ensure smooth operations and team collaboration.
3. **Ideas (I)**: Providing innovative ideas, vision, and guidance to solve problems and move the project or team forward.

Weinberg’s MOI model highlights that a successful leader needs a balance of these three components to lead effectively.

**6. What is the significance of Z-graph in PERT diagram?**

The **Z-graph** in a PERT (Program Evaluation and Review Technique) diagram is used to represent the probability of completing the project within a specific time frame. It shows the relationship between project completion times and the standard deviation of the project duration.

**Significance**: The Z-graph helps project managers assess the likelihood of meeting deadlines by mapping the standard deviation (uncertainty) of the project’s completion time to the probability of finishing by a certain date. This allows for better risk management and more informed decision-making regarding time estimates.

**7. How is cost performance index (CPI) calculated?**

The **Cost Performance Index (CPI)** is calculated using the formula:

* CPI=Actual Cost (AC)\Earned Value (EV)

**EV (Earned Value) is the value of the work performed.**

**AC** (Actual Cost) is the total cost incurred for the work performed.

Can be used to estimate projected cost of completing the project. CPI 1 or 100% would mean that the planned and actual cost are equal i.e. costs are exactly as budgeted. If CPI is less than 1 or below 100%, the project is over budgeted and if it’s higher than 1 or above 100% then project is under budget.

**SECTION B**

**Short Answer Questions**

**Attempt any seven (7) questions out of nine (9) questions (7\*8=56)**

**1. "Software project management is similar to managing other projects and it is far more different as well". Justify this statement with the concept, management process and management control for software projects.**

The statement that "software project management is similar to managing other projects and it is far more different as well" can be justified by understanding both the commonalities and unique aspects of software projects.

### **Similarities:**

**Concept**:

Like any other project, software projects follow basic project management principles such as defining objectives, planning, resource allocation, risk management, and quality assurance. The goal is to deliver a product that meets the stakeholders' requirements, within the scope, time, and budget.

**Management Process**:

Software projects also adhere to the fundamental project management phases: **Initiation**, **Planning**, **Execution**, **Monitoring and Control**, and **Closure**. Project managers in software projects need to schedule tasks, assign responsibilities, and manage timelines just as they would in construction, manufacturing, or any other domain.

**Management Control**:

Controlling costs, monitoring progress, ensuring adherence to timelines, and managing team performance are common across all project types. In software projects, tools like Gantt charts, performance reviews, and reporting mechanisms are used, just as they would be in other industries.

### **Differences:**

**Concept**: Software projects are more **intangible and iterative**, where product features and requirements often evolve. This differs from physical projects where scope and deliverables tend to be more static once finalized. Software project managers must be more adaptable and open to changes.

**Management Process**: Software projects often follow **Agile, Scrum, or other iterative methodologies** rather than traditional linear models like Waterfall. Iterative approaches are designed to manage the uncertainty and flexibility needed in software development, which contrasts with the more rigid processes seen in other industries.

**Management Control**: In software projects, **continuous integration, testing, and deployment** cycles demand constant oversight. Control processes need to be adapted to handle frequent changes, bug fixes, feature updates, and customer feedback. Traditional projects, in contrast, might not require this level of constant revision and real-time adaptability.

### **Conclusion:**

While software project management shares fundamental principles with other project types (e.g., planning, execution, and control), the nature of software development requires specialized methodologies, flexibility in management processes, and tools to handle the evolving, intangible nature of software. This makes it both similar and distinctly different.

**2. Is software review the same as software testing? List different types of reviews. Explain IBEE 1028 generic process for formal reviews.(1+1+6)**

No, software review is not the same as software testing. **Software review** is a static process where documentation, code, or design is examined to identify defects or improvements without executing the software. **Software testing**, on the other hand, is a dynamic process involving executing the software to find bugs or performance issues.

**Different Types of Reviews**

**Code Review**: Examination of source code by other developers to find defects and improve code quality.

**Design Review**: Evaluation of software design specifications to ensure they meet requirements and standards.

**Document Review**: Checking project documentation (e.g., requirements, design documents) for accuracy and completeness.

**Peer Review**: Review conducted by colleagues or team members to provide feedback and identify issues early.

**Inspection**: A formal review process involving detailed examination of work products, often with predefined roles and checklists.

### **IBEE 1028 Generic Process for Formal Reviews**

The **IBEE 1028** (IEEE 1028) standard provides a structured approach to formal reviews. Here’s an overview of its generic process:

**Planning**:

* + Define the review objectives, scope, and criteria.
  + Identify participants and assign roles, such as moderators, reviewers, and authors.
  + Schedule the review meeting and prepare necessary documentation.

**Preparation**:

* + Review materials are distributed to participants before the meeting.
  + Participants prepare by reviewing the documents or artifacts to be examined.
  + Prepare any checklists or review tools to be used during the review.

**Review Meeting**:

* + Conduct the review meeting where participants discuss and examine the artifacts.
  + The meeting is facilitated by a moderator who guides the discussion, manages time, and ensures that all aspects are covered.
  + Participants identify defects, provide feedback, and suggest improvements.

**Issue Resolution**:

* + Document the issues and defects identified during the review.
  + Assign action items to responsible parties for addressing the issues.
  + Follow up on the resolution of issues and verify that corrective actions are taken.

**Reporting**:

* + Prepare a formal review report detailing the findings, issues, and actions taken.
  + The report should include summaries of the review outcomes and recommendations for improvements.

**Follow-Up**:

* + Ensure that the issues identified are resolved and that any necessary changes are made.
  + Conduct follow-up reviews if needed to verify that the issues have been addressed and to assess the effectiveness of the corrective actions.

**3. Software risk management is crucial for the success of a software project. As a project manager, how would you manage risks? Explain.**

Risk is the possibility of loss or damage/injury. Negative Risk involves understanding potential problems that might occur in the project and how they might impede project success. Positive Risk on the other hand are ones that can lead to positive outcomes therefore also referred to as opportunities. Goal should be to minimize potential negative risk while taking potential positive risks.

* Common mistake is to overlook positive risk and only focus on tactical/negative risks
* The importance of good working relationships, especially between project sponsor and project manager

Planning risk management: Deciding how to approach and plan risk management activities for the project. output: Risk Management Plan

Identifying risks: Determining which risks are likely to affect a project and documenting the characteristics of each. output: Risk Register

Performing qualitative risk analysis: Prioritizing risks based on their probability and impact of occurrence. Output: project Document Updates

Performing quantitative analysis: Numerical estimates of effects of risks. output: project Document Updates

Planning risk responses: Taking steps to enhance opportunity and reduce threats. Output: PM plan updates, project documents updates

Controlling risk: Monitoring identified/residual risk, identifying new risk and carrying out risk response while also evaluating the effectiveness of strategies applied. Output: work performance information, changes requests, PM plan updates, organization process assets updates

**4. What is Software Quality Assurance? Explain different software quality assurance approaches.**

Periodically evaluating overall software performance to ensure the software will satisfy the relevant quality standards and requirements. Quality assurance includes all the activities related to satisfying the relevant quality standards for a project.

Different approaches are:

Continuous quality improvement. Kaizen - Kaizen is an approach to creating continuous improvement based on the idea that small, ongoing positive changes can reap significant improvements.



Lean involves evaluating processes to maximize customer value while minimizing waste

Benchmarking generates ideas for quality improvements by comparing specific project practices or product characteristics to those of other projects or products within or outside the performing organization.

**QA using Kanban**

Kanban uses five core properties:

- Visual workflow

- Limit work-in-process

- Measure and manage flow

- Make process policies explicit

- Use models to recognize improvement opportunities

**5. Quality is the totality of functionality and features of a software product that contributes to its ability to satisfy stated or implied needs. Explain software product quality features considering ISO/IEC 25010:2011 standard.**

The International Organization for Standardization (ISO) defines quality as “the degree to which a set of inherent characteristics fulfils requirements”. General definitions from experts are based on two key characteristics: Conformance to requirements and Fitness for use.

**Functional suitability** is the first characteristic, which assesses how well the software provides the necessary functions to meet user requirements. It looks at the completeness of functions, ensuring all necessary features are included, and checks the correctness and appropriateness of these functions to make sure they deliver accurate and useful results.

**Performance efficiency**, which focuses on how well the software performs under specified conditions, particularly regarding its use of resources like memory, processing power, and network bandwidth. It also considers the system's ability to handle varying loads efficiently without significant degradation in performance.

**Compatibility** refers to the software’s ability to function correctly in a particular environment, particularly when interacting with other systems or software products. This ensures the software can coexist with other applications without conflict and can effectively exchange data and services with other systems.

**Usability** is a critical characteristic, as it measures how user-friendly the software is. This includes how easy it is for users to learn the system, how aesthetically pleasing the interface is, and how effectively it helps users avoid mistakes. Usability also ensures that the software is accessible to users with disabilities, making it more inclusive.

**Reliability** is concerned with the software's ability to perform consistently without failure over time. This includes its ability to maintain service availability, tolerate faults, and recover from errors when they occur. A reliable software product ensures users experience fewer interruptions and data loss.

**Security** addresses the ability of the software to protect sensitive data and prevent unauthorized access. This character covers confidentiality, ensuring that data is kept secure from unauthorized users, as well as integrity, ensuring that data remains accurate and uncorrupted. Security also involves mechanisms to track and verify user actions to prevent denial or manipulation of previous operations, contributing to the overall trustworthiness of the system.

**Maintainability** is focused on how easily the software can be modified or updated. This includes the ability to analyze and diagnose defects, make modifications, and test those changes to ensure they work as expected. A maintainable software product is structured in such a way that its components can be easily reused and adapted to new requirements or environments.

**Portability** evaluates how easily the software can be transferred from one environment to another. This involves ensuring that the software can be adapted or installed in new environments with minimal difficulty. A portable software product can also replace other systems with relative ease, making it a more versatile and adaptable solution for various use cases.

**6. What is strategic program management? Explain the concept of stepwise project planning in software project management.**

Strategic planning is the process which involves

● determining long term objectives by analyzing strengths and weaknesses of an organization,

● studying opportunities and threats in the business environment,

● predicting future trends and projecting the need for new products and services.

Often achieved by performing a SWOT analysis i.e. Analyzing Strengths, Weaknesses, Opportunities and Threat

The process of **IT Planning** is crucial for aligning an organization’s technology with its business goals. It can be broken down into four key stages: **IT Strategy Planning**, **Business Area Analysis**, **Project Planning**, and **Resource Allocation**.

1. Software Strategy Planning

This stage focuses on defining the **Software strategy** in alignment with the organization's overall business strategy. It involves setting long-term technology goals that support the business's vision and objectives. Software Strategy Planning outlines the role of IT in the organization, setting priorities for technology investments.

### **2. Business Area Analysis**

Once the SOFTWARE strategy is established, the next stage involves analyzing specific **business areas** to identify where SOFTWARE can have the most significant impact. This is the process of understanding business needs in more detail and translating them into SOFTWARE requirements. Business Area Analysis assesses key departments or functions, such as finance, marketing, operations, or HR, to determine how SOFTWARE solutions can solve problems, streamline processes, or create competitive advantages in those areas.

### **3. Project Planning**

After identifying business areas that require software support, the focus shifts to **Project Planning**. This involves defining the scope, objectives, deliverables, timelines, and budget for each software project. Project planning breaks down the software initiatives into manageable components and outlines the specific tasks that need to be accomplished to achieve the desired outcomes. It also involves risk management, determining project dependencies, setting milestones, and planning for potential challenges that might arise during implementation.

### **4. Resource Allocation**

**Resource Allocation** is the final stage, focusing on assigning the right resources—such as personnel, technology, and budget—to the software projects. It involves determining what human resources (e.g., developers, project managers, analysts), technological resources (e.g., software, hardware), and financial resources (e.g., budget, funding) are needed for successful project completion. This stage also includes capacity planning to ensure that the organization has the necessary skills, infrastructure, and financial support to execute the projects.

OR

**Stepwise project planning** is a structured approach to planning a software project. It helps ensure that all necessary aspects are covered before the actual development work begins, reducing risks and uncertainties. This approach is typically broken down into a series of steps or stages, each contributing to the clarity and structure of the overall project plan. Here's how stepwise project planning works:

1. **Project Objectives and Goals**: The first step involves defining the project’s objectives. This includes understanding what the software is supposed to achieve, the business problems it addresses, and the overall goals in terms of functionality, performance, and quality. Clearly defined objectives help in aligning the project with stakeholder expectations and strategic goals.
2. **Feasibility Study**: The next step is assessing whether the project is viable from technical, financial, and operational perspectives. This involves analyzing risks, resources, and potential obstacles to ensure the project is feasible before significant investments are made.
3. **Project Scope and Requirements**: Once the project is deemed feasible, detailed requirements and scope are defined. This includes specifying what needs to be delivered, the features and functionalities required, and the constraints within which the project must operate. Clear requirements help prevent scope creeps and keep the project on track.
4. **Work Breakdown Structure (WBS)**: The project is then broken down into smaller, manageable tasks or work packages. This helps in organizing and structuring the work that needs to be done, making it easier to assign responsibilities, estimate timelines, and allocate resources.
5. **Scheduling and Resource Planning**: After breaking down the work, project managers create a timeline or schedule. This involves estimating the time required for each task and allocating resources such as personnel, equipment, and budget to ensure the tasks can be completed on time. Tools like Gantt charts and critical path analysis are often used at this stage.
6. **Risk Management Planning**: Identifying potential risks and planning for them is critical to successful project management. Risk management involves anticipating possible issues, such as delays or technical challenges, and developing strategies to mitigate these risks.
7. **Communication and Reporting Plan**: Effective communication is key in project management. A plan is established to ensure all stakeholders, including team members, managers, and clients, are kept informed about project progress, milestones, and any changes to the plan.

**7. Justify that software measurement is a core activity in software project management with the seven-core metrics of software projects.**

**Software measurement** is a fundamental activity in software project management because it provides objective data that project managers can use to make informed decisions, assess performance, and ensure the project is on track to meet its goals. Software measurement helps in tracking progress, identifying risks early, and ensuring that the software meets the desired quality standards.

### **1. Scope Management**

Scope defines the work required to deliver the project. The **size metric**, such as lines of code (LOC) or function points, quantifies the extent of the work. By measuring size, project managers can accurately estimate the scope and track progress against planned scope. This ensures that scope creep is controlled, and the project stays aligned with its objectives. For example, if the planned size of the software is 10,000 LOC, size measurement helps determine how much of the scope has been completed and whether additional scope is being introduced.

### **2. Cost Management**

Cost management involves estimating and controlling the financial expenditure of the project. **Effort and cost metrics**, like person-hours or cost per unit of work, are central to this. Metrics such as the **Cost Performance Index (CPI)** help assess whether the project is within budget. If the CPI is less than 1.0, it indicates the project is overspending. This measurement helps identify cost overruns early, enabling adjustments to keep the project within budget, thus avoiding financial risks.

### **3. Time Management**

Time management ensures that the project is completed on schedule. The **schedule metric**, particularly the **Schedule Performance Index (SPI)**, helps measure whether the project is on track time-wise. For instance, an SPI of 0.9 indicates the project is behind schedule. Using schedule metrics enables timely adjustments, such as reallocating resources or adjusting timelines, to prevent delays. Tracking milestone completion and schedule adherence helps ensure timely project delivery.

### **4. Quality Management**

Quality is critical for the success of software projects. **Quality metrics** such as defect density, test coverage, and code complexity allow the project team to assess the product's quality throughout its lifecycle. Quality metrics ensure that the software meets both functional and non-functional requirements. High defect density or low test coverage signals potential quality risks, allowing for early interventions like increased testing or code reviews to maintain software quality and reduce costly rework.

### **5. Human Resources (HR) Management**

Effective management of the project team is vital for project success. **Productivity metrics** provide insights into how efficiently the team is working, often measured as output (e.g., features or LOC) per person-hour. This helps in balancing workloads, identifying bottlenecks, and making decisions about whether additional staff or training is needed. By measuring productivity, project managers can optimize resource allocation, improving team performance and morale while ensuring project deadlines are met.

### **6. Communications Management**

Clear and accurate communication is essential to keep all stakeholders informed. The use of **software measurement metrics**, such as cost (CPI), schedule (SPI), and defect counts, provides a solid foundation for data-driven reporting. These metrics allow the project manager to provide clear updates on project health and performance. Effective communication ensures transparency, making it easier for stakeholders to understand the project’s current status and any risks or issues that need addressing.

### **7. Risk Management**

Risk management involves identifying and mitigating potential issues that could affect the project. **Defect metrics** and **quality metrics** are particularly useful for identifying technical risks early in the development process, such as high defect rates or low code quality, which could lead to rework or system failures. Metrics like CPI and SPI also help detect cost and schedule risks, respectively. By measuring and monitoring risks, project managers can proactively implement mitigation strategies, such as reallocating resources or adjusting timelines, to reduce the likelihood and impact of these risks.

**8. Define software configuration management? What are the activities covered by SCM? Discuss different software team organization. (1+2+5)**

**Software Configuration Management (SCM)** refers to a systematic approach for managing and controlling changes in software systems. It involves tracking and controlling the versions of software components and documentation throughout the software development lifecycle to ensure consistency, traceability, and integrity of the software product. SCM ensures that changes are made systematically and documented properly, which helps in maintaining the stability of the software as it evolves.

**Activities Covered by SCM**

* **Configuration Identification:** This activity involves defining and documenting the configuration items (CIs) that need to be managed. Configuration items can include source code, documentation, design specifications, and executable files. Identifying and categorizing these items help in tracking and managing them throughout their lifecycle.
* **Configuration Control:** Configuration control manages changes to configuration items. It involves establishing procedures for submitting, reviewing, and approving change requests. This process ensures that changes are evaluated for impact and implemented in a controlled manner, minimizing disruptions and maintaining software integrity.
* **Configuration Status Accounting:** This activity involves recording and reporting on the status of configuration items and changes. It provides an audit trail that tracks the history of changes, including who made the changes, when they were made, and what the changes were. This helps in tracking progress and maintaining a clear record of modifications.
* **Configuration Auditing:** Configuration auditing involves reviewing and verifying that the configuration items and their changes comply with the specified requirements and standards. Audits ensure that the software's configuration is accurate, complete, and consistent with its documentation and requirements.
* **Configuration Management Planning:** This involves developing a configuration management plan that outlines the procedures and policies for managing configuration items. The plan defines how SCM activities will be implemented, the tools and resources required, and the roles and responsibilities of team members involved in SCM.

### **Different Software Team Organizations**

**Functional Organization** is a traditional structure where the team is divided based on specialized functions such as development, testing, and design. In this setup, each function operates independently, with team members reporting to a functional manager. The advantage of a functional organization lies in the deep specialization and expertise that can be developed within each functional area. However, this structure can lead to communication barriers and a lack of coordination between different functions, potentially causing delays and integration challenges when different functional teams need to collaborate on a project.

**Project-Based Organization** focuses on organizing teams around specific projects. In this model, team members from various functional areas come together to form a project team, which is responsible for all aspects of the project from start to finish. This approach enhances focus on project goals and fosters better communication and coordination among team members. By concentrating on a single project, teams can be more agile and responsive to project needs. However, this structure can lead to resource contention if multiple projects are running simultaneously and may result in duplicated efforts if not managed effectively.

**Matrix Organization** combines elements of both functional and project-based structures. In a matrix organization, team members report to both functional managers and project managers. This dual reporting structure allows for flexibility and efficient resource utilization, as team members can be allocated to different projects based on demand and their expertise. It encourages collaboration across different functions and improves coordination. However, the matrix structure can create complexities due to conflicting priorities and potential confusion over reporting lines, which requires careful management to ensure that team members understand their roles and responsibilities clearly.

**9. Say a project depended on a data center vulnerable to fire. It might be estimated that if fire occurred a new computer configuration could be established for $400,000. It might also be estimated that there is a 2% chance that a fire will occur. Installing fire alarms at a cost of $400 would reduce the chance of fire to 0.7%. Will the action of installing alarms be worthwhile?**

To determine whether installing fire alarms is worthwhile, we need to compare the expected cost of fire damage with and without the fire alarms.

Here’s the step-by-step analysis:

**Without Fire Alarms:**

* + Estimated cost of reconfiguring after a fire = $400,000
  + Probability of fire occurring = 2% or 0.02
  + Expected cost of fire without alarms = $400,000 \* 0.02 = $8,000

**With Fire Alarms:**

* + Cost of installing fire alarms = $400
  + New probability of fire occurring = 0.7% or 0.007
  + Expected cost of fire with alarms = $400,000 \* 0.007 = $2,800
  + Total cost with alarms = Cost of installing alarms + Expected cost of fire with alarms = $400 + $2,800 = $3,200

**Comparison:**

* + Expected cost without alarms = $8,000
  + Total cost with alarms = $3,200

### **Conclusion:**

By installing the fire alarms, the total cost is reduced from $8,000 to $3,200. The savings are $8,000 - $3,200 = $4,800. Thus, installing the fire alarms is **worthwhile**, as it reduces the expected costs significantly.

**LONG ANSWER QUESTIONS**

**SECTION C**

**Attempt any two (2) questions out of three (3) questions (2\*15=30)**

1. a. As a software project management expert, what are the necessary improvements needed in the conventional way of software project management? Explain considering drawbacks of waterfall model.

The conventional way of software project management, often represented by the **Waterfall model**, requires several improvements to address its limitations and better align with modern software development needs.

### **Drawbacks of the Waterfall Model**

1. **Inflexibility**: The Waterfall model follows a strict, linear process where each phase must be completed before the next one begins. This rigidity makes it difficult to accommodate changes once a phase is completed, which is a significant drawback in dynamic environments where requirements often evolve.
2. **Late Testing**: Testing is usually done at the end of the project, meaning issues are often discovered late in the development process. This can lead to high costs and extended timelines, as changes made late in the project require revisiting earlier phases.
3. **Lack of Customer Involvement**: In the traditional Waterfall approach, customers are heavily involved at the beginning of the project (during requirements gathering) but may have minimal involvement afterward. As a result, their feedback might only be considered at the end of the project, by which time addressing their concerns can be costly and time-consuming.
4. **Assumption of Requirements Stability**: The Waterfall model assumes that all requirements can be fully defined at the beginning of the project and that they will remain stable. However, this is often unrealistic in software projects where user needs and market conditions frequently change.
5. **Risk of Misalignment**: With long phases and limited feedback loops, there is a risk that the final product may not align with customer expectations or business goals, especially if the requirements have changed during the lengthy development process.

### **Necessary Improvements in Software Project Management**

To address the limitations of the Waterfall model, software project management should evolve in several key areas:

1. **Adoption of Iterative and Incremental Models**: Introducing iterative and incremental models, such as Agile, can improve flexibility and responsiveness to changes. In these models, development is broken down into smaller cycles or sprints, allowing teams to revisit requirements, incorporate feedback, and refine the product continuously. This helps accommodate changes in scope, market conditions, and customer needs more effectively.
2. **Continuous Testing and Integration**: Shifting to practices like continuous testing and integration enables testing to happen throughout the development cycle, rather than at the end. This allows for earlier detection of issues, reducing the cost and effort of fixes, and improving overall software quality.
3. **Increased Customer Collaboration**: Agile methodologies promote regular customer involvement throughout the project. This ensures that customer feedback is continuously integrated, reducing the risk of building a product that does not meet the customer’s needs. Regular iterations and reviews foster collaboration, keeping the project aligned with user expectations.
4. **Focus on Adaptability Over Predictability**: Modern software project management emphasizes adaptability to change rather than strict adherence to an initial plan. By using techniques like backlog management and prioritizing features based on current needs, teams can adapt to evolving requirements and business conditions more fluidly.
5. **Risk Management Throughout the Lifecycle**: In conventional approaches like Waterfall, risk management is often an upfront activity. By contrast, iterative models incorporate continuous risk management practices, identifying and addressing risks throughout the development lifecycle, which helps mitigate potential failures earlier.
6. **Smaller Deliverables and Early Value**: Shifting to methodologies that promote smaller, incremental deliverables enables the early release of valuable features. This allows customers to derive business value from the software early on, while providing feedback that helps shape future development.

**b. Explain the features of software team organization types and their impact on software project management.**

### **Features of Software Team Organization Types and Their Impact on Project Management**

1. **Functional Organization**: In a functional team structure, members are grouped based on their specialized roles, such as development, testing, or design. Each function operates independently, and team members report to a functional manager. This structure fosters deep expertise within each function and is efficient for task specialization. However, it often results in poor communication across teams, leading to delays in integration and decision-making. Functional silos can also limit flexibility in resource allocation and responsiveness to changing project needs.
2. **Project-Based Organization**: In this model, teams are formed around specific projects, with members from different functional areas working together under a project manager. This structure encourages close collaboration and ensures that the project team is dedicated to achieving the project goals. Project-based teams are more adaptable to changes and can respond more quickly to project requirements. The downside is that it can lead to resource contention between projects, especially when multiple projects are running concurrently. Managing resources effectively becomes crucial in this setup.
3. **Matrix Organization**: The matrix structure combines aspects of both functional and project-based organizations. Team members report to both a functional manager and a project manager, balancing between functional expertise and project focus. This structure is more flexible, allowing for efficient resource utilization across projects. It promotes collaboration and ensures that specialized skills are available to multiple projects. However, the dual reporting lines can create confusion and conflict in priorities, requiring clear communication and well-defined roles to manage effectively.

### **Impact on Software Project Management**

The type of team organization directly influences the **management of scope, time, cost, and communication** in a software project. In functional organizations, the challenge lies in cross-team coordination and integrating efforts, which can lead to delays. Project-based organizations improve focus on deliverables but might struggle with resource allocation, impacting cost management. Matrix organizations strike a balance between flexibility and specialization but require careful coordination to avoid conflicts, impacting both communication and decision-making processes.

**2. Earned Value Management (EVM) is a methodology that combines scope, schedule, and resource measurements to assess project performance and progress and commonly used method of performance measurement for projects. How EVM helps the project management team to assess and measure project performance and progress?**

**Earned Value Management (EVM)** is a powerful methodology that helps project management teams assess and measure project performance and progress by integrating scope, schedule, and cost metrics into a single, unified framework. EVM provides insights into how well a project is performing against its baseline plans, enabling managers to make informed decisions and take corrective actions as needed.

Here’s how EVM helps in various aspects of project management:

1. EVM enables the project management team to evaluate how much work has been completed (earned value) compared to what was planned (planned value).
2. The **Schedule Performance Index (SPI)**, calculated as SPI = EV / PV, is a key metric that shows how efficiently the project is adhering to its timeline. An SPI value greater than 1 indicates that the project is ahead of schedule, while a value less than 1 suggests the project is falling behind.
3. The **Cost Performance Index (CPI)**, calculated as CPI = EV / AC, shows how well the project is managing its budget. A CPI value greater than 1 indicates that the project is under budget, while a value less than 1 suggests the project is over budget.
4. The **Estimate at Completion (EAC)** formula combines historical performance data (CPI and SPI) to predict the total cost of the project based on current trends.
5. EVM highlights areas where risks are emerging or where corrective actions are needed.

You are managing a complex software project and have been using Earned Value Management (EVM) techniques to track its progress. The project has a total budget of $500,000 and is expected to be completed in 10 months. After 6 months of work, you gather the following data:

Planned Value (PV) at the end of 6 months: $300,000

Earned Value (EV) at the end of 6 months: $280,000

Actual Cost (AC) at the end of 6 months: $320,000

Now, calculate the following EVM metrics:

a. Cost Performance Index (CPI) at the end of 6 months.

b. Schedule Performance Index (SPI) at the end of 6 months.

c. Estimate to Complete (ETC) at the end of 6 months.

d. Estimate at Completion (EAC) for the whole project.

e. Variance at Completion (VAC) for the whole project.

* **CPI** =EV/AC= 0.875 (Project is over budget)
* **SPI** = EV/PV=0.933 (Project is slightly behind schedule)
* **ETC** = BAC-EV/CPI= $251,429 (Estimated cost to complete the remaining work)
* **EAC** = BAC/CPI=$ 571,429 (Total estimated cost at completion)
* **VAC** = BAC-EAC= -$71,429 (Expected cost overrun)

3. For the project having following details:

a. Find expected duration of the project and variance of each activity.

b. Draw network diagram and find critical path and expected project completion time.

c. What is the probability of completing the project on or before 22 weeks?