Chapter 1
Software Management Practice and Software Economics (3Q) (23 marks)

1. What is a project? What do you mean by project methodology?

Project

- A project is a combination of set of objectives to be accomplished within a fixed period.
- Simply, it is a special task or job with a clear purpose and a specific end date.
- It has a clear beginning and end, and it's carried out to create a product, service, or result.
- Example: Creating a new software application.
- Some of the key points about projects are:
  - i. **Unique Purpose:** Projects have a specific goal.
  - ii. **Temporary:** They have a start and end date.
  - iii. **Collaborative:** Involves different people and teams.
  - iv. **Constraints:** Limited by time, budget, and resources.
  - v. **Risk and Uncertainty:** Can face unexpected challenges.

Project Methodology

- Project methodology, in simple words, is a well-defined plan or set of guidelines that show how a project will be organized, executed, and completed.
- It is a step you need to take to achieve your project's goals.
- Different methodologies may have their own rules and processes, and they help teams work together efficiently, stay organized, and deliver the project successfully.
- Some of the key points of project methodology are:
  - i. **Step-by-step plan**: A clear roadmap that guides the project from start to finish.
  - ii. **Best practices**: Guidelines for effective work and avoiding common mistakes.

iii. **Roles and Responsibilities:** Clearly define who does what in the project.

iv. **Risk Management:** Strategies to identify and handle potential issues.

Example: Agile, Waterfall, Spiral model are common project methodologies.

2. What is a software project? Explain the characteristics of software projects.

- A software project is a specific type of project that involves the development and delivery of a software product, system, or application.
- Software projects encompass activities like planning, designing, coding, testing, and implementing the software within a defined timeframe.
- The primary objective is to produce a functional and high-quality software product that meets the requirements of the stakeholders and end-users.
- Software projects have a clear start and end date and are managed to achieve their objectives within the allocated time, budget, and resources.

The characteristics of software projects are:

1. **Clear Goals:** Software projects have specific objectives, like creating a new app, upgrading a system, or fixing bugs. They know what they want to achieve.

2. **Temporary:** Software projects have a start and end date. They are not ongoing forever. Once they achieve their goal, they're done.

3. **Cross-Functional:** Software projects involve different people with various skills. Like a team of designers, programmers, and testers, working together.

4. **Constraints:** Projects have limits, such as time, money, and resources. They need to stay within these boundaries.

5. **Continuous Evolution:** Software projects keep growing and improving. They update to stay useful and relevant over time.

6. **Customer Collaboration:** Users and stakeholders work closely with the project team. Their feedback helps create better software.

7. **Risk and Uncertainty:** Projects may face unexpected problems. The team plans and adapts to handle these challenges.

8. **Documentation:** Software projects keep records of what they do. This helps in future changes and understanding how things work.

(Asked 2 times)

3. What is software project management? What are the major activities of software project management?

- Software Project Management is the process of planning, organizing, and coordinating resources and activities to develop software products or applications successfully.
- The key points of software project management are:
    1. **Organizing and Leading**: Managing a team to develop software.
    2. **Planning and Coordination**: Creating a roadmap and coordinating efforts.
    3. **Resource Management**: Efficiently using people, tools, and budget.
    4. **Risk Handling:** Identifying and dealing with potential problems.
    5. **Communication**: Keeping everyone informed and aligned.
    6. **Quality Assurance**: Ensuring the software meets the desired standards.

In software project management, the key phases of the project life cycle include planning, execution, monitoring, and control.

i. **Initiation**: Defining project objectives and scope.
ii. **Planning**: A detailed project plan is developed, outlining tasks, resources, timelines, and budgets.
iii. **Execution**: Actively working on the project tasks and building the software product.
iv. **Monitoring**: Regularly tracking project progress and comparing it with the plan.
v. **Control**: Making adjustments to the project plan to stay on track.
vi. **Closure:** This is the final phase where the project is formally completed and closed.

- The major activities of the software project management are:

1. **Project Planning and Tracking:**
   - Creating a detailed project plan with tasks, timelines, and resources.
   - Tracking progress, comparing it with the plan, and making adjustments as needed.

2. **Project Resource Management:**
   - Identifying and allocating the right people, tools, and budget for the project.
   - Ensuring efficient use of resources throughout the project.

3. **Scope Management:**
   - Defining and managing the project's scope, including features and deliverables.
   - Preventing scope creep and maintaining focus on the project's objectives.

4. **Estimation Management:**
   - Estimating the time, effort, and cost required for project tasks.
   - Using estimation techniques to plan and allocate resources effectively.

5. **Project Risk Management:**
   - Identifying potential risks that could impact the project's success.
   - Developing strategies to mitigate or handle these risks.

6. **Scheduling Management:**
   - Creating a project schedule and setting timelines for tasks and milestones.
   - Ensuring tasks are completed on time and the project stays on schedule.

7. **Project Communication Management:**
   - Establishing a communication plan to keep stakeholders informed.
   - Facilitating regular communication among team members and stakeholders.

8. **Configuration Management:**
    - Managing the project's software and documentation versions.
    - Ensuring consistency and controlling changes throughout the project.

4. What is the importance of software project management?

OR

Need for software project management?

The importance of software project management is:

1. **Goal Achievement:**
- Project management ensures that the software project has clear objectives and a roadmap to achieve them.
- It keeps the team focused on the project's goals and guides them toward successful completion.

2. **Efficient Resource Utilization:**
- Project management helps in planning and allocating resources like time, budget, and manpower effectively.
- It reduces wastage, optimizes resource usage, and improves overall productivity.

3. **Risk Mitigation:**
- Project management identifies potential risks that could hinder the project's success.
- It develops strategies to minimize the impact of risks, safeguarding the project from failure.

4. **Clear Communication:**
   : Project management facilitates clear communication among team members and stakeholders, reducing misunderstandings and ensuring everyone is on the same page.

5. **Scope Control:**

- Effective project management helps prevent scope creep, ensuring that the project stays within its defined boundaries and objectives.

6. **Quality Assurance:**

- Project management emphasizes quality standards and best practices.
- It ensures that the software product meets the required level of excellence and fulfills user needs.

7. **Timely Delivery:**

- Project management creates realistic schedules and keeps the project on track to meet deadlines.
- It helps in delivering the software product on time, avoiding delays and customer dissatisfaction.

8. **Continuous Improvement:**

- Project management encourages learning from past projects and using that knowledge to improve future ones.
- It enables teams to refine their processes and deliver better results with each new project.

(Asked 3 times)

5. Explain Triple Constraints of software project management.

OR

Explain the triple constraints of software project management.

OR

What are the different constraints that operate on every project.

- In software project management, the Triple Constraints, also known as the Project Management Triangle, represent the three fundamental factors that impact every project.

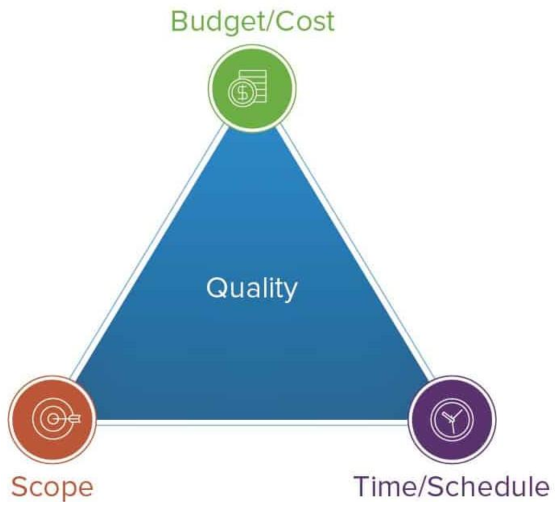


Fig: SPM triple Constraint

They are:

1. **Scope:**

- Refers to the work that needs to be done to deliver the software product, including the features, functionalities, and deliverables.
- Changes in scope can impact other constraints.

2. **Time/Schedule:**
   - Represents the project's schedule or duration.
   - It defines the start and end dates of the project, as well as key milestones and deadlines.
   - Changes in the timeline can affect scope and cost.

3. **Budget/Cost**:
   - Refers to the budget allocated for the software project.
   - It includes all expenses required to complete the project, such as labor, materials, tools, and equipment.
   - Changes in cost can affect the scope and timeline.

These three constraints are interconnected, and any adjustment to one constraint will have implications on the other two.

For example:

i. Increasing the scope of a project may require more time and resources, leading to an increase in cost.
ii. Shortening the project's timeline may require additional resources to meet the deadline, resulting in increased cost.
iii. Reducing the budget may require a decrease in scope or an extension of the timeline.

- Managing the Triple Constraints effectively is essential for successful software project delivery.
- Project managers must strike a balance between these constraints to achieve project objectives and meet stakeholder expectations.

Q. Explain in detail about the quality attribute.

- Quality is a critical aspect of any project and refers to the degree to which the deliverables meet the specified requirements and expectations of the stakeholders.
- It involves the absence of defects or errors and the fulfillment of customer expectations for performance, reliability, security, accuracy and functionality.
- Achieving high-quality results is a key objective in any project to ensure customer satisfaction and meet industry standards.

Q. What are the 5 constraints that operate on every projects.

The five constraints that operate on every project are interrelated and have a significant impact on project planning and execution.

The five constraints are:

1. Scope
2. Time/Schedule
3. Budget/Cost
4. Quality
5. Risk

In addition to the Project Management Triple Constraints, there are two other constraints that are also essential in project management:

Explain 1 2 3 from Que no 5.

Quality:

- Quality is a critical aspect of any project and refers to the degree to which the deliverables meet the specified requirements and expectations of the stakeholders.

Risk:

- The potential uncertainties, issues, and challenges that may impact the project's success.
- Risk management involves identifying, assessing, and mitigating these risks.

These constraints are interrelated, and changes to one constraint often affect the others.

6. Why project fails?

- Projects can fail for various reasons, and some common factors include:

1. **Undefined Objectives**: Lack of clear and specific project goals leads to confusion and directionless efforts.

2. **Poor Planning**: Inadequate or insufficient planning can result in unrealistic timelines, inadequate resources, and scope creep.

3. **Inadequate Resources**: Not having enough skilled team members, budget, or tools hinders project progress.

4. **Scope Creep:** Constantly expanding the project scope without proper management disrupts timelines and resources.

5. **Ineffective Communication**: Poor communication among team members and stakeholders leads to misunderstandings and delays.

6. **Unclear Roles and Responsibilities**: Ambiguity in roles and responsibilities creates confusion and delays decision-making.

7. **Insufficient Risk Management**: Failing to identify and address potential risks can lead to unforeseen issues.

8. **Inadequate Quality Control:** Neglecting quality assurance can lead to defective deliverables and customer dissatisfaction.

9. **Changing Requirements**: Frequent changes in requirements can disrupt project progress and increase costs.

(Asked 4 times)

7. List and explain what the major difficulties in software project management.

OR

Project management issues in web based project.

➢ Major difficulties in software project management include:

1. **Unclear Requirements**: Ambiguous or constantly changing requirements make it challenging to plan and deliver the right solution.

2. **Time Constraints**: Tight deadlines and aggressive schedules can lead to rushed development and compromised quality.

3. **Resource Limitations:** Insufficient skilled resources, tools, or budget can hinder project progress and lead to delays.

4. **Scope Creep:** Constantly expanding project scope without proper control disrupts timelines and causes budget overruns.

5. **Communication Issues:** Poor communication among team members, stakeholders, and clients can lead to misunderstandings and conflicts.

6. **Risk Management:** Inadequate risk assessment and mitigation strategies can result in unforeseen issues and project setbacks.

7. **Technology Complexity:** Dealing with complex technologies and integrations may require specialized skills and experience.

8. **Changing Requirements:** Frequent changes in requirements can lead to rework and hinder project progress.

9. **Client Involvement:** Insufficient or inconsistent client involvement can hinder decision-making and feedback cycles.

(Asked 3 times)

8. What is software planning? Write down the good features of software project planning.
OR
Explain the essential elements for project planning.
OR
Project success factor with examples.

- Software project planning is the process of creating a detailed roadmap for a software project, outlining its goals, tasks, timelines, and resource requirements to ensure successful execution and achieve desired outcomes.
- The features and elements of software project planning are:

i. **Clear Objectives**: Clearly defined and measurable project objectives that outline what the project aims to achieve.

ii. **Detailed Scope**: A well-defined scope that outlines the boundaries of the project, including what will be included and excluded.

iii. **Work Breakdown Structure (WBS):** Breaking down the project into smaller, manageable tasks and sub-tasks, creating a hierarchical structure.

iv. **Realistic Timelines and Milestones:** Establishing a realistic project timeline with milestones to track progress and ensure timely completion.

v. **Resource Allocation:** Identifying and allocating the right resources (team members, tools, equipment) to the project is critical for successful execution.

vi. **Risk Assessment and Mitigation:** Identifying potential risks and developing strategies to mitigate them, ensuring smooth project execution.

vii. **Communication Strategy:** Effective communication is vital for software planning. The plan should define how and when team members and stakeholders will be updated on project progress and developments.

viii. **Quality Assurance:** Establishing processes and procedures to ensure the project delivers a high-quality product or service.

(Asked 2 times)

9. Explain the roles and responsibilities of the project manager.

OR

Explain the important activities that a software project manager performs during software project planning.

- A project manager is a professional responsible for the planning, design, execution, monitoring, controlling, and closure of a project.
- They represent an essential role in the achievement of the projects.
- They play a pivotal role in ensuring the successful delivery of the project's objectives within the defined constraints of scope, time, and budget.
- Project managers are effective communicators, skilled organizers, and capable at managing resources and teams.
- A project manager must be a good leader, medium between client and the team, and the mentor of the team.

The roles and responsibilities of a project manager are:

1. Project Planning
2. Resource Management
3. Team Leadership
4. Risk Management
5. Communication Management
6. Scope Control
7. Time Management
8. Cost Management
9. Quality Assurance
10. Performance Tracking
11. Documentation and Reporting

Q. What are the different roles and skill sets needed for a project?

- Roles and Skill Sets in a Project Team:

1. **Project Manager**: Responsible for overall project planning, execution, and success.
2. **Business Analyst:** Gathers and analyzes project requirements.
3. **Software Developer**: Creates and codes software solutions.
4. **Quality Assurance (QA) Engineer**: Tests and ensures software quality.
5. **UI/UX Designer:** Designs user interfaces and user experiences.
6. **Database Administrator (DBA):** Manages databases.
7. **DevOps Engineer:** Manages software deployment and operations.
8. **Technical Writer:** Creates project documentation.

Q. What skills or qualities are important in selecting a project team?

1. Technical Proficiency
2. Communication Skills
3. Problem-Solving Abilities
4. Team Player
5. Adaptability
6. Time Management
7. Creativity

Q. What are the different ways to organize a software project team.

1. **Functional Teams**: Team members are grouped by their functional roles (e.g., development, testing, design).
2. **Cross-Functional Teams**: Team members from different functions collaborate closely on the project.
3. **Project-Based Teams**: Assembled specifically for a particular project and disbanded after its completion.
4. **Virtual Teams**: Team members work remotely, often from different locations, using digital collaboration tools.
5. **Dedicated Teams**: Team members work exclusively on the project without distractions from other tasks.

(Asked 6 times)

9. Explain the motivation factors identified for a project?

OR
Methods used to increase staff motivation.

- The methods used to increase staff motivation are:

1. **Clear Goals:** Set specific and achievable goals for employees to give them a sense of direction and purpose in their work.

2. **Employee Recognition Programs**: Acknowledge and reward employees' good performance and achievements to boost their morale.

3. **Training and Development**: Offer training and development opportunities to enhance employees' skills and knowledge, making them feel valued and invested in.

4. **Autonomy or Empowerment**: Give employees the freedom and authority to make decisions and take ownership of their work, which increases their motivation and responsibility.

5. **Performance Feedback**: Regularly provide constructive feedback to employees to help them improve and feel supported in their roles.

6. **Opportunities for Growth**: Offer career advancement opportunities and challenging assignments to keep employees motivated and engaged.

7. **Team Collaboration**: Foster a collaborative and supportive work environment to promote teamwork and camaraderie among employees.

8. **Positive Work Culture**: Cultivate a positive work environment with open communication and mutual respect to increase employee satisfaction and motivation.

9. **Team-Building Activities:** Organize team-building activities to strengthen team relationships and create a sense of belonging.

By implementing these methods, organizations can create a positive and motivating work environment, leading to increased employee productivity, job satisfaction, and overall organizational success.

(Asked 2 times)

10. Define the project management life cycle model. Explain any two modes.

OR

Name the different software production process models. bring out a comparison of these process models.

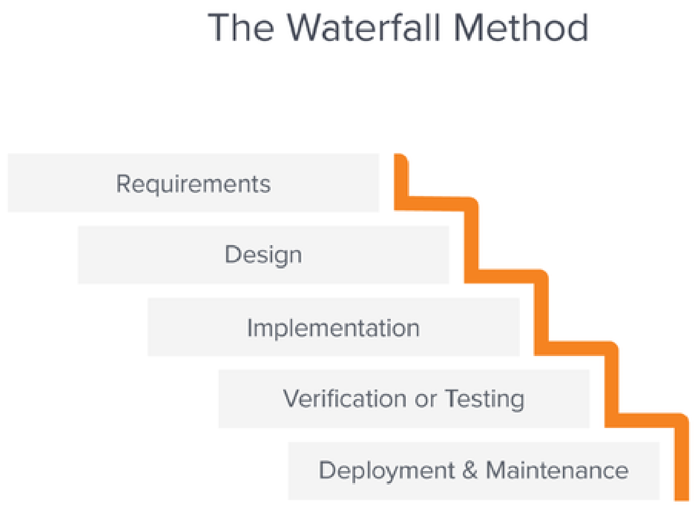
- It is also known as the Software development Life Cycle.
- A Project Management Life Cycle Model is a framework that defines the various phases a project goes through, from initiation to closure.
- Simply, it is a step-by-step process that outlines how software is planned, designed, developed, tested, deployed, and maintained.
- It provides a structured approach to creating high-quality software products, ensuring they meet the desired requirements and are delivered on time and within budget.
- A typical software development life cycle consists of the following stages.
    1. Requirement Analysis
    2. Design
    3. Implementation
    4. Testing
    5. Deployment and Maintenance

Any two software development life cycle models are:

1. Waterfall Model
2. Spiral Model

11. Explain briefly waterfall model with advantages and disadvantages.

- The Waterfall model is a traditional software development approach where each phase of the development process follows a linear and sequential flow.
- The project progresses through defined phases, and each phase must be completed before moving to the next phase.
- It does not allow for backward movement, meaning once a phase is completed, it is challenging to go back and make changes.



Advantages of the Waterfall Model:

1. **Clarity and Simplicity**: The linear and sequential nature of the model makes it easy to understand and manage.
2. **Well-Defined Phases**: Each phase has clear objectives and deliverables, providing a structured approach to development.
3. **Documentation:** Extensive documentation is produced at each phase, making it easier to maintain and transfer knowledge.
4. **Stable Requirements**: Well-defined requirements at the beginning reduce the chances of frequent changes during development.
5. **Resource Allocation:** The Waterfall model allows for efficient resource allocation since each phase has a fixed scope and timeline.

Disadvantages of Waterfall Model:

1. **Lack of Flexibility**: The model is rigid and does not handle changes well, leading to potential challenges with evolving requirements.
2. **Limited Customer Involvement**: Customer feedback is minimal until the end, which may result in misalignment with customer needs.
3. **High Risk**: If issues are discovered late in the process, it can be costly and time-consuming to address them.
4. **Long Development Cycles:** The sequential nature can lead to longer development cycles, delaying the delivery of the final product.
5. **No Early Prototyping**: The model does not allow for early prototypes, which may lead to unexpected design or functionality issues.

12. Explain improvements of Waterfall model.

The improvements of waterfall model are:

1. Plan Everything First:

- ➢ Complete the program design before starting coding to avoid frequent changes.

2. Keep Good Records:

- ➢ Maintain up-to-date documentation for better understanding and future updates.

3. Test Thoroughly:

- ➢ Plan, control, and monitor testing to find and fix problems early.

4. Review and Improve:

- ➢ Review each phase carefully and make improvements whenever possible.

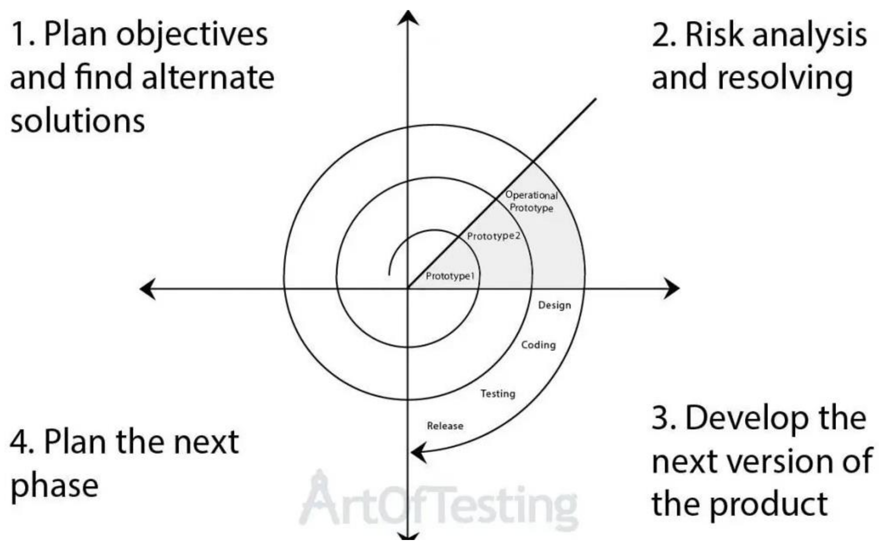
5. Listen to Customers:

- ➢ Involve customers throughout the project to meet their needs and expectations better.

(Asked 2 times)

13. Explain spiral model with its advantages.

- The Spiral model is an iterative and risk-driven approach to software development.
- It combines elements of the Waterfall model and iterative development, focusing on addressing risks throughout the project's life cycle.
- The project progresses in a spiral pattern, with each loop representing a cycle of planning, risk analysis, engineering, and evaluation.


1. Plan objectives and find alternate solutions
2. Risk analysis and resolving
Operational Prototype
Prototype2
Prototype1
Design
Coding
Testing
Release
4. Plan the next phase
3. Develop the next version of the product
ArtOfTesting

Advantages of Spiral Model:

1. **Risk Management**: The model emphasizes risk analysis and mitigation, allowing for better risk management throughout the project.

2. **Flexibility**: The Spiral model is adaptable to changing requirements and can incorporate feedback from stakeholders at each cycle.

3. **Customer Involvement**: Regular reviews and evaluations involve customers, ensuring the delivered product meets their expectations.

4. **Prototyping:** Early prototypes allow stakeholders to visualize the final product and make informed decisions.

5. **Continuous Improvement**: Each cycle provides an opportunity for improvement based on lessons learned from the previous cycle.

Disadvantages of Spiral Model:

1. **Complexity**: The model can be more complex to manage, especially for small projects with limited resources.

2. **Higher Cost**: The emphasis on risk analysis and prototypes may increase project costs.

3. **Time-Consuming**: The iterative nature may lead to longer development cycles.

4. **Expertise Required**: Skillful risk assessment and management expertise are essential to leverage the model's benefits effectively.

5. **Less Predictable:** The iterative nature makes it harder to estimate project timelines and costs accurately.

14. What are the three generations of software development?

Conventional Development (60s and 70s), Transition (80s and 90s), and Modern Practices (2000 and later)

- The three generations of software development refer to the evolution and progression of software development methodologies and practices over time.
- Each generation brought about new approaches to managing software projects.
- The three generations are:

i. **First Generation**: (Conventional or Craftsmanship)

- The first generation of software development, also known as the "Traditional" or "Waterfall" approach, emerged in the 1950s and continued into the 1970s.
- It was rigid and inflexible, with limited feedback from end-users during development.

ii. **Second Generation**: (Transition)

- The second generation of software development arose in the 1980s and 1990s as a response to the limitations of the first generation.
- This generation introduced iterative and incremental methodologies, such as the "Spiral model" and "Rapid Application Development (RAD)."
- These approaches allowed for multiple cycles of development and emphasized customer involvement and feedback throughout the project.
- It was more flexible and adaptive than the Waterfall model.

iii. **Third Generation** (Modern Practices)

- The third generation of software development started to gain prominence in the late 1990s and continues to the present day.
- This generation is characterized by Agile methodologies, with "Scrum" and "Extreme Programming (XP)" being notable examples.
- It encourages frequent iterations and incremental development, allowing teams to respond quickly to changes and deliver software more efficiently.

Each generation built upon the previous one, addressing the challenges and limitations of previous methodologies.

(Asked 5 times)

15. Explain five staffing principles offered by Boehm.

OR

Explain the staffing principle for improving team effectiveness.

OR

How staffing should be done for software project.

The staffing principles offered by Boehm are:

1. **The Principle of Top Talent:**

- Hire the best and most skilled individuals for each role in the project team.
- Top talent brings efficiency and high-quality output to the project.
- Use better and fewer people.
- The quality of the people is important.
- It is better to use a smaller number of people with more skills.

2. **The Principle of Job Matching:**

- Match each team member's skills and strengths with suitable project tasks.
- Ensure that team members are assigned roles that align with their expertise.
- Job matching enhances productivity and job satisfaction.
- The skill sets of people are different.
- The best programmer may not be suitable as an architect or a manager.
- Similarly, an architect or manager is not suitable as a programmer.

3. **The Principle of Career Progression:**

- Provide opportunities for career growth and skill development to team members.
- Encourage continuous learning and advancement within the organization by providing training programs.
- Career progression boosts employee motivation and loyalty.

4. **The Principle of Team Balance:**

- Assemble a well-balanced team with diverse skills and backgrounds.
- Ensure a mix of junior and senior members to foster knowledge transfer.
- Team balance leads to better problem-solving and collaboration.
- Select people who will complement and harmonize with one another.
- They should be friendly and quickly mingle with their co-members of the team.

5. **The Principle of Phase Out:**

- Gradually phase out or remove team members when their specific skills are no longer needed.
- Avoid keeping team members in roles where they are no longer contributing effectively.
- Phase out allows for efficient resource allocation and project adaptability.
- Similarly, Keeping a misfit on the team doesn't benefit anyone.
- A misfit demotivates other team members, will not self-actualize and destroys the team balance.

(Asked 3 times)

16. Explain Conventional project Management.

- ➢ Conventional project management is a traditional and structured approach to planning, organizing, and executing projects.
- ➢ It follows a linear and sequential process where each step is completed before moving to the next.
- ➢ It emphasizes detailed planning, well-defined roles, and adherence to a fixed scope, timeline, and budget.
- ➢ Key characteristics of conventional software management in software project management include:

1. **Waterfall Model:** The Waterfall model is a typical representation of conventional software management, where each phase of the project progresses linearly, and each phase's deliverables are well-defined.

2. **Detailed Planning**: Conventional software management emphasizes complete project planning before development begins. This involves creating detailed project schedules, resource allocation, and budgeting.

3. **Fixed Requirements**: Requirements are gathered and finalized at the beginning of the project, and any changes to requirements are discouraged.

4. **Documentation**: Documentation is a crucial part of the process.. It includes project plans, design documents, test plans, and user manuals.

5. **Sequential Execution**: Conventional projects follow a sequential approach, where one phase must be completed before moving to the next. This approach can lead to longer development cycles.

6. **Limited Customer Involvement**: Customer involvement is typically limited to the beginning and end of the project, with minimal feedback during development.

- While conventional software management has its strengths, such as providing clear structure and predictability, it also has limitations, especially when dealing with complex or rapidly changing projects.
- In response to these limitations, modern software project management approaches, such as Agile methodologies, have emerged to address the need for more flexibility, customer involvement, and iterative development.

17. Explain conventional software management performance.

Conventional software management performance refers to how well traditional practices and principles are applied in software development projects.

1. **Early Problem Detection**: Finding and fixing software issues during the early design phases is much more cost-effective than after the software is delivered.
2. **Scheduling Limits**: While software development schedules can be shortened to some extent, there are practical limits to how much time can be reduced without compromising quality.
3. **Maintenance Costs:** The cost of maintaining software after development is usually twice as much as the initial development cost.
4. **Code Size Impact:** Software development and maintenance costs are influenced by the number of lines of code written.
5. **Human Productivity**: The productivity of team members varies, with some contributing more to the project's success than others.
6. **Programming Effort**: Only a small percentage of the overall software development effort is spent on actual programming.
7. **Contributor Impact**: A small percentage of contributors typically account for a large part of the project's success.

Understanding these principles helps project managers and teams make better decisions to improve their software development process and overall project performance.

(Very Imp)

18. Explain the principle of conventional software engineering.

- The principle of conventional software engineering are listed below:

1. **Good Management is More Important than Good Technology:**
- Effective planning and organization matter more than using advanced technology.

2. **People are the Key to Success:**
- Skilled and motivated team members play a crucial role in project success.

3. **Take Responsibility:**
- Be accountable for your actions and decisions.

4. **Avoid Tricks:**
- Don't take shortcuts or use unreliable methods.

5. **Get it Right Before You Make it Faster:**
- Focus on correctness before trying to make things faster.

6. **Focus on low Coupling and high Cohesion:**
- Design components that work well together and keep things simple.

7. **Don't Test Your Own Software:**
- Let someone else test the software to avoid bias.

8. **Give Products to Customers Early:**
- Deliver versions of the software to customers as soon as possible.

9. **Put Techniques Before Tools:**
- Choose the right methods and practices over relying on specific tools.

(Asked 7 times) Very Imp

19. What are the principles of modern software management.

1. **Architecture-first approach:**

- Start by creating a strong software design or architecture before coding.
- Identify and address potential issues early in the process.
- We can take decisions regarding the design to improve productivity and quality.

2. **Iterative Life-Cycle process:**

- In the iterative life cycle process, we repeat the process again and again to eliminate the risk factors.
- An iterative life cycle mainly has four steps requirement gathering, design, implementation, and testing.
- Identify risks and make improvements in each iteration.
- Continuously refine the software to meet changing needs.

3. **Component-based development:**

- In component-based approach is a widely used and successful approach in which we reuse the previously defined functions for the software development.
- We reuse the part of code in the form of components.
- Reuse pre-built components to speed up development.
- Enhance consistency and efficiency in software creation.
- Leverage existing solutions to solve common problems.

4. **Change management environment:**

- Change Management is the process responsible for managing all changes.
- The main aim of change management is to improve the quality of software by performing necessary changes.
- All changes implemented are then tested and certified.
- Facilitate collaboration among different teams.

5. **Round-trip engineering:**
- Keep code and design models in sync automatically.
- In round trip engineering code generation and reverse engineering take place at the same time in a dynamic environment.
- Round-trip engineering is like keeping your code and design drawings in sync automatically.
- When you make changes to the code, the design gets updated, and vice versa.
- It helps developers work seamlessly between code and diagrams, making software development easier to manage and maintain.

6. **Model-based notation:**
- Use clear models to represent the software design. Eg: UML diagrams
- Improve communication and understanding among team members.
- Facilitate better planning and decision-making.

7. **Objective quality control:**
- The objective of quality control is to improve the quality of our Software.
- Track progress and identify areas for improvement.
- Enhance software reliability and performance.

8. **Demonstration-based approach:**
- In this approach, we mainly focus on demonstration.
- Gather feedback and improve software based on user input.
- Ensure the software meets user expectations.

9. **Evolving level of detail:**
- Plan software development in incremental releases.
- Adapt to changing requirements and priorities.
- This approach allows teams to adapt to changes and priorities while continuously improving the software in manageable increments.

10. **Configurable process:**
- A configurable process is like having a toolbox with different tools that can be adjusted based on the specific needs of each project.
- It involves creating a flexible framework that can be customized to fit different software development projects.

Q. Compare and Contrast the principle of conventional software engineering and modern software management.

(Asked 3 times)

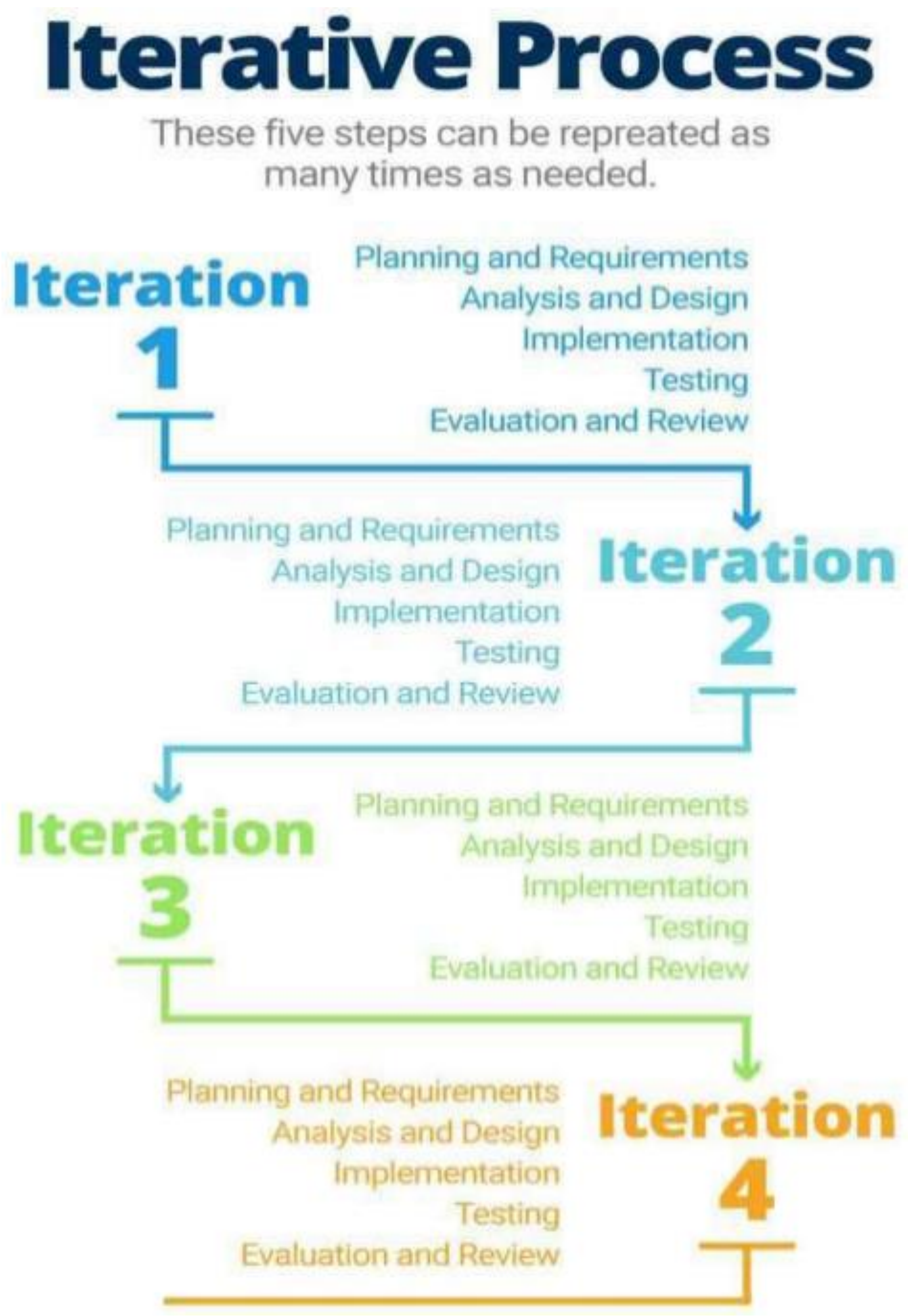
20. Explain Iterative Process

OR

Explain Iterative Process Planning

- Iterative process is the method of breaking the project into small, manageable parts called iterations.
- Each iteration involves planning, designing, coding, and testing a specific set of features or functionality.
- After each iteration, the software is evaluated, and feedback is gathered from users and stakeholders.
- The feedback is used to make improvements and refinements to the software in the next iteration.
- This cycle of planning, building, testing, and refining continues until the software meets all requirements and is ready for release.
- The iterative process allows for flexibility, adaptation to changes, and early identification of issues.

- The iterative process is the practice of building, refining, and improving a project, product.
- Teams that use the iterative development process create, test, and revise until they're satisfied with the end result.
- The benefits and challenges of iterative process:
  - i. Increased efficiency.
  - ii. Increased collaboration
  - iii. Reduced project-level risk
  - iv. More reliable user feedback


Iterative Process
These five steps can be repreated as many times as needed.
Iteration 1
Planning and Requirements
Analysis and Design
Implementation
Testing
Evaluation and Review
Iteration 2
Planning and Requirements
Analysis and Design
Implementation
Testing
Evaluation and Review
Iteration 3
Planning and Requirements
Analysis and Design
Implementation
Testing
Evaluation and Review
Iteration 4
Planning and Requirements
Analysis and Design
Implementation
Testing
Evaluation and Review

(Asked 3 times)

21. What is software economics? Explain about the factors affecting software economics.

OR

What are the basic factor of software economics.

OR

What are the five basic parameter of the software cost model?

- ➢ Software economics refers to the study of how software is created, distributed, and used in a way that considers the costs, benefits, and overall value it provides.
- ➢ It involves understanding the financial aspects of software development, including factors like development costs, pricing, licensing, and the impact of software on businesses and users.
- ➢ Essentially, it's about understanding the money side of software.
  It includes cost estimation, return on Investment (ROI), market demand, software maintenance economics.

The basic factors affecting software economics are:

1. **Size**:
- ➢ Refers to the magnitude or complexity of the software project.
- ➢ Larger projects generally require more resources and time to complete.
- ➢ Smaller projects may be more cost-effective but may have limited functionality.

2. **Process**:
- ➢ The software development process used to create the product.
- ➢ Efficient and well-defined processes can lead to cost savings and improved productivity.
- ➢ Choosing the right development methodology (e.g., Agile, Waterfall) impacts the project's economics.

3. **Personnel:**

- The skills, experience, and number of people involved in the project.
- Highly skilled and experienced developers may produce higher-quality software more efficiently.
- The project team's size can affect development costs and the timeline.

4. **Environment:**

- The tools, infrastructure, and technology used in the software development environment.
- Up-to-date and appropriate tools can enhance productivity and reduce development time.
- The software development environment affects the overall cost and efficiency of the project.

5. **Required Quality:**

- Refers to the level of quality and reliability expected from the software product.
- Higher-quality standards may require additional testing and resources, impacting costs.
- Balancing quality requirements with budget constraints is essential for economic feasibility.

Note:

	Basic factor of software economics	Important trends in improving software economics
1	Size	Reduce Software Product Size
2	Process	Improving software process
3	Personnel	Improving team effectiveness
4	Environment	Improving automation through software environment
5	Required Quality	Achieving required quality

(Not asked yet)

22 What are the important trends in improving software economics?

- The important trends in improving software economics are:

1. **Reduce Software Product Size**: Focusing on building software with only essential features and eliminating unnecessary complexity.
2. **Improve Software Process**: Adopt iterative and incremental development approaches.
3. **Improve Team Effectiveness**: Foster collaboration and provide skill development opportunities.
4. **Improve Automation**: Use automated testing and continuous integration.
5. **Achieve Required Quality**: Implementing testing methodologies, including unit testing, integration testing, and end-to-end testing.

22. How can we reduce software product size?

They are:

1. Modular Design – break into smaller modules
2. Eliminate unused features.
3. Code Refactoring
4. Optimized data structure
5. Code compression and media compression
6. Lazy loading

(Asked 2 times)

23. Explain Economic feasibility.

- Economic feasibility in software project management means determining whether a software project is financially viable and worthwhile.
- It involves evaluating the project's costs and potential benefits to ensure that the investment makes sense from a financial standpoint.
- In simple words, economic feasibility helps answer the question, "Is the software project worth the money and resources we will invest in it?"
- It ensures that the project aligns with the organizational development strategic goals, making it a financially sound decision to pursue the development.

(Asked 3 times)

24. Explain software cost estimation.

- Software Cost Estimation is the process of determining the effort, time, and resources required to develop a software project.
- It helps project managers and stakeholders plan and budget effectively.
- There are different techniques used for cost estimation.
- Each technique has its strengths and weaknesses, and project managers choose the most suitable method based on project characteristics and available data.
- It helps in setting realistic budgets, allocating resources effectively, and making informed decisions for successful software development.
- Some of the techniques include:

1. Lines of Code (LOC):

- LOC estimates the project's size based on the number of lines of code that need to be written.
- It assumes that the more lines of code, the more effort and time required for development.
- However, it can be challenging to accurately predict LOC at the early stages of a project, as the codebase might change during development.

2. Function Points (FP):

- FP measures the functionality of a software system based on user interactions and data processing.
- It considers the number of inputs, outputs, inquiries, files, and interfaces to calculate the total function points.
- FP can be used to compare projects with different programming languages.
- Function Points provide a more meaningful measure of software size compared to Lines of Code (LOC)

3. COCOMO (Constructive Cost Model):

- COCOMO is an algorithmic cost estimation model.
- It is designed to estimate the effort, time, and resources required to develop a software project.
- It is particularly useful in the early stages of a project when detailed information is limited.

- It categorizes projects into three modes: Basic, Intermediate, and Advanced, based on their size and complexity.
- It considers various project attributes, team factors, and historical data to provide a more accurate estimation compared to simplistic size-based models like Lines of Code (LOC).

25. What are the disadvantages of using LOC. What are the the advantages of function point over LOC?

- **Disadvantages** of using Lines of Code (LOC):

1. Inconsistency due to coding style and languages.
2. Focuses on quantity rather than quality.
3. Ignores non-coding efforts.
4. Doesn't measure functionality.

- **Advantages** of Function Points (FP) over LOC:

1. Focuses on functionality and user value.
2. Language-independent for better comparisons.
3. Considers complexity of interactions.
4. Standardized and reliable estimation.
5. Reflects user experience.
6. Helps in resource allocation and budgeting.

Q. Explain Function Point Method - once

Q. Explain COCOMO model – 2 times

Q. What are the cost benefit evaluation techniques? Explain in brief.

1. **Return on Investment (ROI):**

- ROI measures the percentage of return on the initial investment.

2. **Payback Period:**

- The payback period calculates the time required for the project's cash inflows to recover the initial investment.

26. Critically evaluate the reuse of software as a software development option.

- Software reuse is a development approach that involves using existing software components, modules, or code to build new applications.
- It offers several advantages and has been promoted as a way to increase productivity and reduce development time and costs.
- However, there are also some critical considerations and challenges associated with software reuse:

Advantages of Software Reuse:

1. Time and Cost Savings
2. Improved Quality
3. Faster Time-to-Market
4. Consistency and Standardization

Challenges and Considerations

1. Compatibility and Integration
2. Documentation and Support
3. License and Legal Issues
4. Maintainability
5. Customization Overhead
6. Limited Reusable Components

(Asked 2 times)

27. Modern Software Economics

- Modern software economics is an approach that applies economic principles to software projects.
- It involves using data and metrics to make informed decisions, accurately estimating costs for better budgeting, and managing risks effectively.
- It aligns software development with business objectives and encourages a value-driven approach to software projects.
- Here are some key aspects of modern software economics:

1. **Cost Estimation and Budgeting**: Modern software economics helps allocate budgets effectively and ensure that projects are financially viable.

2. **Return on Investment (ROI) Analysis**: It focuses on evaluating the potential benefits and returns that a software project can deliver. This analysis aids in prioritizing projects and choosing those with the highest ROI.

3. **Value-based Development**: Modern software economics emphasizes delivering value to customers and end-users. Projects are aligned with customer needs and business goals to ensure that the software addresses critical requirements.

4. **Agile Practices**: Agile methodologies are integrated into modern software economics to promote flexibility, customer collaboration, and continuous improvement.

5. **Risk Management**: Assessing and mitigating risks is a crucial component of modern software economics. It involves identifying potential challenges and uncertainties that may impact the project's success.

6. **Software Reusability**: Modern software economics promotes the reuse of existing software components and assets to save time, effort, and costs in development.

7. **Cost of Quality**: This aspect focuses on finding a balance between investing in quality assurance and addressing defects. Modern software economics highlights the cost implications of quality decisions.

28. Object oriented software development brings out the scope of improvement in the software project economics discuss.

1. **Code Reusability and Maintenance:**
   - **Explanation**: OOP lets us make reusable building blocks (classes) for software. This means less repeated work, faster development, and less effort to keep things working well.
   - **Impact**: Costs for making new software parts or fixing existing ones go down, improving the project's financial side.

2. **Modularity and Scalability:**

- **Explanation**: OOP helps break big projects into smaller, manageable pieces. This makes it easier to handle big projects and expand them smoothly.
- **Impact**: Large projects become less overwhelming, can be developed at the same time by different teams, and handle growth more easily.

3. **Efficient Change Management:**

- Explanation: OOP allows us to change parts of the software without messing up the whole thing. This makes updating software easier and less risky.
- Impact: Costs for adapting software to changes become lower, as changes are safer and quicker to make.

4. **Faster Development Cycle:**

- Explanation: OOP methods like Agile and Scrum speed up software creation. They help us build working software parts quickly and respond faster to market needs.
- Impact: Quicker development means we can get our software out there faster, attracting customers sooner and potentially earning more.

5. **Improved Quality and Testing:**
- Explanation: OOP designs are tidy and easy to test. This means fewer mistakes and better software quality.
- Impact: Better quality means fewer problems, less time spent finding and fixing bugs, and lower costs for the project.

6. **Flexibility in System Evolution:**
- Explanation: OOP lets us add new things or changes to the software without redoing everything. We can build on what's already there.
- Impact: Updating software becomes cheaper since we don't have to redo everything from scratch.

7. **Enhanced Team Collaboration:**
- Explanation: OOP projects encourage teamwork. Different team members can work on different parts at the same time, making development smoother.
- Impact: Teams work together well, saving time and using resources better.

8. **Better User Satisfaction:**
- Explanation: OOP software matches what users want and need. This makes users happier and more likely to stick with the software.
- Impact: Happy users can bring in more users and better business outcomes, boosting the project's economics.

In short, object-oriented software development brings a bunch of good things to projects: less repeated work, smoother teamwork, quicker updates, better software quality, and happier users. All these lead to better financial results for the project.

Chapter 2
Software Process Primitives and Process Management Framework (4Q) (30 marks)

(Asked 4 times)

1. Explain different life cycle phases.

   OR

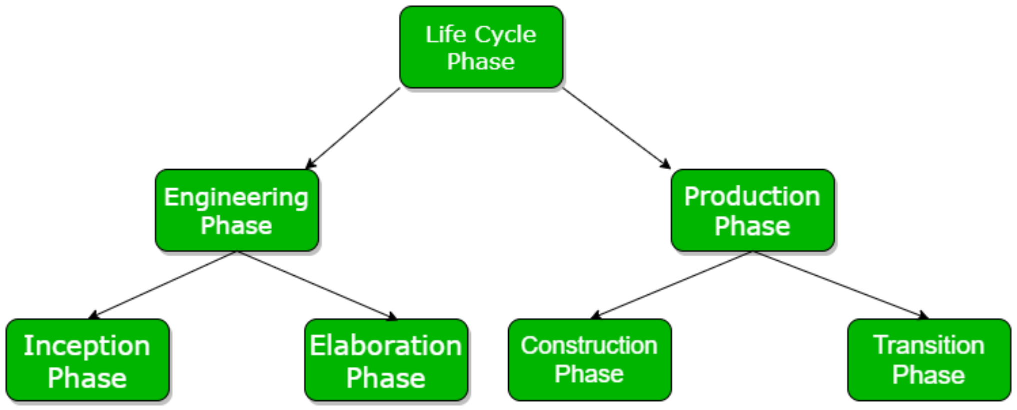
   Explain briefly two stages of the life cycle pashes, engineering and production.

   OR

   What are the differences between engineering and production stages of software life cycle.

➢ The life cycle phase is divided broadly into two categories.

1. Engineering Phase - (Inception and Elaboration Phase)
2. Production Phase – (Construction and Transition Phase)



➢ A life cycle phase in software development is a specific stage or step that the software goes through from its initial idea to its completion and release.

- The phases are explained below:

1. **Engineering Phase**

- Engineering phase involves establishing the goal and defines the overall scope of the project.
- Initial planning, requirements gathering, and understanding user needs.
- This phase involves the small team size.
- Engineering phase is further divided into two phases:

   a. **Inception Phase**

   - This is where the idea for the software is born, and the project is initiated.
   - It involves identifying the problem or need that the software will address and understanding the goals and requirements.
   - The outcome of this phase is a clear concept of what the software should achieve and a high-level plan to move forward.

   b. **Elaboration Phase**

   - In this phase, the initial idea is expanded and detailed planning takes place.
   - The team creates a comprehensive schedule, budget, and resource allocation.
   - The team conducts in-depth analysis, breaking down requirements into smaller tasks, and designing the software architecture.
   - It aims to have a well-defined plan before proceeding to the next phase.

2. **Production Phase**

- In the Production phase, the actual development of the software takes place.
- Implement the software based on the design and requirements from the engineering phase.
- Coding, testing, and building all the components and features of the software.
- A fully functional software product ready for release to users.
- It is further divided into 2 Phases:

a. Construction Phase

- In this phase, the actual development of the software takes place.
- The team writes code, designs user interfaces, and builds the software system based on the defined requirements.
- Regular testing and feedback are carried out to ensure quality and adherence to the plan.
- The outcome of this phase is a working version of the software that fulfills the specified requirements.

b. Transition Phase

- The software is prepared for deployment in the transition phase.
- The final product is thoroughly tested and is made ready for the end-users.
- Training is provided to users, and documentation is completed.
- The stable software product is delivered to the customer, and the project is formally closed.

Q. Explain different stages of project life cycle.

They are:

i. Initiation
ii. Planning
iii. Execution
iv. Monitoring
v. Control
vi. Closure

(Asked 2 times)

2. What are the purposes, activities and the evaluation criteria of the phases of the life cycle process within the production stage.

- The production stage of the software development life cycle encompasses the Construction and Transition phases.
- Each phase serves distinct purposes, involves specific activities, and can be evaluated based on certain criteria.

1. **Construction Phase:**

Purpose:

- The main purpose of the Construction phase is to build the software product based on the detailed design and requirements specified in the planning phase.

Activities:

i. **Coding**: Writing code to implement the functionalities of the software.
ii. **Unit Testing**: Testing individual units or components to ensure they work as intended.
iii. **Debugging**: Identifying and fixing defects or issues in the software code.
iv. **Documentation**: Creating technical documentation to support future maintenance and updates.

Evaluation Criteria:

i. **Code Quality**: Evaluating the quality of the code based on coding standards and best practices.
ii. **Test Coverage**: Assessing the percentage of code that is covered by testing.
iii. **Progress**: Tracking the completion of planned coding tasks and deliverables.

2. Transition Phase:

Purpose:

- The purpose of the Transition phase is to deploy the software to the end-users and ensure a smooth transition from development to operational use.

Activities:

i. **System Testing**: Performing end-to-end testing to verify that the entire system functions correctly.
ii. **User Acceptance Testing (UAT):** Allowing end-users to test the software and provide feedback.
iii. **Training**: Providing training to users and support staff to use the software effectively.
iv. **Deployment**: Rolling out the software to the production environment.

Evaluation Criteria:

i. **User Satisfaction**: Gathering feedback from users about their experience with the software.
ii. **Successful Deployment**: Ensuring that the software is successfully deployed without major issues.

3. Describe 5 project management processes and how they support each phase of the project life cycle.

The five project management processes and how they support each phase of the project life cycle are:

1. Project Planning:

Supports Inception and Elaboration: During these phases, project planning is crucial to define the project's scope, objectives, and requirements. It involves creating a detailed project plan, setting up timelines, allocating resources, and identifying potential risks and mitigation strategies.

2. **Risk Management:**

Supports Inception, Elaboration, and Construction: Risk management is essential in identifying and addressing potential risks throughout the project life cycle. Inception and Elaboration focus on risk assessment and planning, while Construction involves monitoring and controlling risks during development.

3. **Quality Management:**

Supports Construction and Transition: Quality management ensures that the software is developed to meet the defined requirements and standards. During the Construction phase, quality checks are performed regularly, and any defects are addressed. In Transition, final quality assurance and testing take place before delivering the product to the customer.

4. **Change Management:**

Supports Elaboration and Construction: Change management helps manage changes to project requirements and scope. During Elaboration, any changes in the initial plan are evaluated, and their impacts are assessed. In Construction, change management ensures that modifications are controlled and documented to prevent scope creep.

5. **Communication Management:**

Supports the Entire Life Cycle: Effective communication is crucial at every phase of the project. It helps in clarifying requirements, updating stakeholders on progress, and resolving issues. It ensures that all team members are aligned with project objectives and informed about any changes or updates.

(Asked 5 times)

4. What are the artifact and artifact sets? Describe them with their purpose and notations.

OR

What do you understand by artifacts? What are the two sets? Explain anyone sets.

- Artifacts and artifact sets refer to the documents and deliverables created during various phases of the project.
- Artifacts are the documents and files produced during software development, while artifact sets are groups of these artifacts that are organized based on their relevance to specific phases or milestones in the project.
- They are essential for keeping track of project progress, facilitating collaboration, and ensuring the successful completion of the software development process.

Artifacts

- Artifacts are documents or files that capture important information, decisions, and progress throughout the software development process.
- They serve as a record of the project, helping teams communicate, collaborate, and maintain a clear understanding of what needs to be done.
- Artifacts can include requirements documents, design diagrams, code files, test plans, and user manuals.

Artifact Sets:

- An artifact set is a collection of related artifacts that are grouped together based on their relevance to a particular phase or milestone in the project.
- These sets help organize and manage the project's documentation, making it easier for team members and stakeholders to find the information they need.
- An artifact set for the inception phase might include a project vision document and initial requirement specifications, while a construction phase artifact set could include coding standards and code review documents.

Artifacts are organized into two sets:

1. **Engineering Set**

- An engineering artifact set in software development refers to a collection of documents and deliverables that are created during the engineering phase of the project.
- The various artifacts in this are:
  a. **Requirement artifact sets**
    - Its purpose is to capture and document the functional and non-functional requirements of the software, outlining what the software should do and how it should perform.
    - Notation: Software Requirements Specification (SRS), Use Case Diagrams, User Stories.

  b. **Design artifact sets**
    - Its purpose is to describe the architecture and design of the software system, including the overall structure, data flow, and interactions between components.
    - Notation: System Design Document, UML Diagrams (Class Diagrams, Sequence Diagrams), Data Flow Diagrams.

  c. **Implementation artifact sets**
    - Its purpose is to provide the actual source code and programming instructions for building the software.
    - Notation: Source Code Files (e.g., .java, .cpp), Compiled Executables

  d. **Deployment artifact sets**
    - Its purpose is to support the deployment and installation of the software in the production environment.
    - Notation: Deployment Plan, User documentation

2. **Management Set**

- A management artifact set in software development comprises documents and deliverables related to project planning, scheduling, and overall management.
- These artifacts are crucial for tracking progress, ensuring the project stays on track, and facilitating effective communication among team members and stakeholders.

- The various artifacts are:
  a. **Planning artifact sets**
     - Its purpose is to define the project scope, objectives, schedule, resources, and risks, providing a roadmap for the entire software development process.
     - Notation: Project Plan, Gantt Charts, work breakdown structure

  b. **Operational artifact sets**
     - Its purpose is to monitor and control the software project's progress and performance, ensuring it stays on track and meets its goals.
     - Notation: Project Status Reports, Progress Metrics, Status Assessment, Change Requests, Environment

(Asked 6 times)

5. Explain the software architecture on the basis of management perspective and technical perspective.

- The software architecture is the overall design of the software system.
- It defines how different components of the software interact and work together to achieve the desired functionality. Think of it as the blueprint or plan that guides the development of the software.
- The architecture baseline is not just a document but a collection of information from various engineering sets.
- Software architecture is crucial in complex systems.
- Poor architecture and immature processes can lead to project failure.

From a management perspective, architecture has three aspects:

i. Architecture as design concept
ii. Architecture baseline
iii. Architecture description.

From a technical perspective, architecture has four views.

i. Design view
ii. Process view
iii. Component view
iv. Deployment view

1. **Management Perspective of Software Architecture:**

From a management perspective, software architecture plays a crucial role in project success. It involves three different aspects:

a. **Architecture as Design Concept**:
   - This aspect focuses on creating a detailed design of the software system.
   - It serves as a foundation for the actual coding and development process.
   - It helps in identifying potential design challenges and solutions early in the project.

b. **Architecture Baseline**:
   - The architecture baseline is not just a single document; rather, it's a collection of information from different engineering artifacts.
   - It aims to satisfy all stakeholders that the project's vision, functionality, and quality can be achieved.
   - Essentially, it confirms that the planned software is feasible and aligns with the project's goals.

c. **Architecture Description**:
   - This aspect involves creating a human-readable representation of the software architecture.
   - It extracts essential information from the design models and presents it in a clear and understandable way.
   - It helps stakeholders, including managers, grasp the big picture of software design.
   - It enables understanding of how different components will interact and work together.
   - It supports effective communication among the development team and stakeholders.

2. Technical Perspective of Software Architecture

- The technical aspect of software architecture is all about how a software system is designed and organized to work effectively.
- It includes decisions about which components will do what, how they will interact with each other, and how the software will be deployed on hardware and systems.
- It helps developers understand how to build the software and how all the pieces will fit together.
- It ensures that the software is well-structured, easy to maintain, and able to perform its tasks smoothly.

Technical Perspective of Software Architecture has four key views:

i. **Design View**: It shows the main structures and functions of the software's design, focusing on the big picture.

ii. **Process View**: This view helps understand how different parts of the software work together and how they control each other.

iii. **Component View**: It describes how the software is organized into different parts and how these parts interact.

iv. **Deployment View:** This view tells us how the software components are placed and used on the actual hardware and systems.

Q. Sometimes only one of them is also asked.

(Asked 2 times)

6. What is meant by software process workflow? What are the key principles of software process workflow?

- A software process workflow is a systematic, step-by-step guide that outlines the activities and tasks to be performed during the software development process.
- It defines the sequence in which the tasks should be executed, ensuring that the development team follows a structured approach.
- It assigns responsibilities for each task to specific team members, clarifying who is responsible for what.
- Many workflows are designed to be iterative and incremental.

- A typical software process workflow might include the following stages:

1. Management
2. Environment
3. Requirements Gathering
4. Design
5. Implementation
6. Testing
7. Deployment

The key principle of software process workflows are:

1. Architecture-first approach
2. Iterative life-cycle process
3. Round trip engineering
4. Demonstration based approach.

Note: They are the top 4 principles of modern software management from chapter 1

(Asked 4 times)

7. Explain status monitoring. What are the parameters for status monitoring of software projects?

- Status monitoring in software projects refers to the process of regularly tracking the progress, performance, and overall situation of the project.
- It includes regular updates and check-ins to understand how things are progressing.
- It helps identify any delays or obstacles that might be slowing down the project's progress.
- It helps project managers and stakeholders stay informed about how well the project is progressing and whether it is on track to meet its goals.

Parameters for status monitoring of software projects include:

i. **Schedule**: Monitoring the project schedule involves tracking whether tasks are being completed on time and if any delays are occurring.

ii. **Budget**: Monitoring the budget involves keeping track of project expenses and ensuring that the project stays within the allocated budget.

iii. **Quality**: Monitoring the quality of the software being developed is essential to ensure that it meets the required standards and satisfies user needs.

iv. **Scope**: Monitoring the scope ensures that the project stays focused on its original objectives and that there are no unnecessary changes or additions that may cause scope creep.

v. **Risks**: Monitoring risks involves identifying potential risks and assessing their impact on the project. It helps in implementing mitigation strategies to avoid negative outcomes.

vi. **Resource Utilization:** Monitoring resource utilization ensures that the project is making efficient use of resources like manpower, equipment, and tools.

vii. **Milestones**: Monitoring project milestones helps in tracking the progress of the project and achieving important checkpoints.

viii. **Communication**: Monitoring communication involves ensuring that information flows effectively among team members and stakeholders, promoting collaboration and understanding.

By regularly monitoring these parameters, project managers can make informed decisions, identify potential issues early, and take necessary actions to keep the project on track for successful completion.

(Asked 5 times)

8. Explain milestones. Differentiate major and minor milestones.

- Milestones are specific points in a project's life cycle used to measure progress and signal important events or achievements.
- They act as checkpoints to track the project's progress and ensure that it is on track to meet its objectives.
- They provide visibility of project progress.
- Meeting milestones indicates successful progress, while failure to meet them prompts corrective actions.
- There are two types of milestones:
  i. Minor milestones
  ii. Major milestones

- Minor milestones represent smaller accomplishments within the project, while major milestones denote significant achievements that mark the completion of major project phases.

i. Minor milestones

- These are smaller, intermediate checkpoints that represent the completion of important tasks or sub-tasks within the project.
- Minor milestones serve as checkpoints for monitoring progress within an iteration.

- Minor milestones are usually achieved more frequently throughout the project's lifecycle.
- Minor milestones are intermediate achievements that occur between major milestones.
- They serve as progress indicators and help to keep the project moving forward steadily.
- Achieving minor milestones ensures that smaller goals are met on schedule, contributing to the successful completion of the overall project.
- For example, completing a specific feature like development of user profile page, completing the design phase of a software module or finishing a round of testing can be considered minor milestones.

ii. Major milestones

- Major milestones, on the other hand, represent significant accomplishments that mark the completion of major phases or stages in the project.
- They occur less frequently and usually have a more impact on the project's overall progress.
- Major milestones are reached at the end of each development phase (e.g., inception, elaboration, construction, transition).
- They address system-wide events and are crucial for verifying that the phase goals have been achieved.
- They ensure that the project is proceeding as planned and help keep all stakeholders informed.
- Examples of major milestones could be finishing a key module like completion of user authentication system, successful integration of all software modules, or the final delivery of the product to the client .

Q. Difference between minor and major milestones.

	Minor Milestone	Major Milestone
1	It indicates incremental progress within project phases.	It signifies completion of significant project phases.
2	It reflects completion of smaller tasks or components.	It marks the achievement of substantial project goals.
3	It contributes to the project's advancement gradually.	It has a high impact on the project's overall progress.
4	It can sometimes be accomplished by individual efforts.	It often require multiple tasks to be completed before.
5	It is important, but missing one doesn't affect the project.	It is vital for project success and major goal achievement and missing one affects project progress.
6	It help build up to major milestones.	It indicates a major shift in project direction.
7	Implementing a specific UI element.	Launching the core product feature.

(Asked 2 times)

9. Explain the general status of plans, requirements, and product across the major milestones.

- At major milestones in a project, the general status of plans, requirements, and the product is assessed to ensure the project is on track.

Plans: Check if the project is following the initial schedule, budget, and resources outlined in the plan.

Requirements: Review if all the desired features and functions are being developed as per the defined requirements.

Product: Evaluate the state of the product being developed, ensuring it's meeting quality standards and matches the intended design.

- These assessments help ensure that the project is progressing as expected, and any necessary adjustments can be made to keep everything on course.

10. What is a difference between a deliverable and a milestone.

Deliverable: A deliverable is something that the project team creates or produces during the project. It's a tangible result, like a document, code, design, or any completed work that's meant to be given to stakeholders.

Milestone: A milestone is a significant point in the project timeline. It's like a marker that indicates progress. It's not something you hand over, but rather a key event or achievement that shows you're moving forward in the project.

In short, deliverables are the things you make, while milestones are the points that show you're making progress.

(Asked 4 times)

11. Explain periodic status assessment. What activities are covered in this assessment?

- Periodic status assessment involves checking the progress, performance, and overall situation of a project or task on a regular basis.
- It's done at predefined time intervals, such as weekly, bi-weekly, or monthly, to ensure consistent monitoring.
- It helps to track how much work has been completed and if the project is moving forward as planned.
- Any issues, roadblocks, or challenges that have arisen since the last assessment are identified.
- Based on the assessment results, decisions can be made to adjust plans, resources, or strategies as needed.
- It serves as an opportunity to communicate updates, concerns, and achievements to team members and stakeholders.
- Periodic assessments contribute to continuous improvement by catching problems early and making ongoing refinements

Some common activities in periodic status assessment are:

1. **Progress Review:**
   - Checking the completion status of tasks and milestones.
   - Comparing the current progress with the planned timeline.

2. **Task Updates:**
   - Collecting updates from team members about their ongoing tasks.
   - Identifying tasks that are completed, in progress, or delayed.

3. **Issue Identification:**
   - Identifying any roadblocks, challenges, or issues that have arisen.
   - Discussing the nature of the issues and potential solutions.

4. **Resource Allocation:**
   - Evaluating the allocation of resources (people, time, budget) to tasks.
   - Ensuring that resources are being utilized effectively.

5. **Risk Assessment:**
   - Reviewing the project's risk register and identifying any new risks.
   - Discussing mitigation strategies for existing and potential risks.

6. **Quality Check:**
   - Assessing the quality of work completed so far.
   - Checking if work meets the required standards and specifications.

7. **Communication and Reporting:**
   - Sharing project updates with stakeholders and team members.
   - Creating reports or presentations to communicate the project's current status.

8. **Documentation:**
   - Updating project documentation with the latest information.
   - Keeping records of assessment findings and decisions made.

9. **Goal Alignment:**
   - Checking if the project is still aligned with its original goals and objectives.
   - Discussing any shifts in priorities or project scope.

10. **Future Planning:**
    - Discussing the upcoming tasks, milestones, and goals.
    - Planning for the next phase of the project.

Q. What is the difference between a software process model and a software process? How do you improve software process.

1. Software Process Model:

Definition: It's a high-level framework that outlines the stages and activities of the entire software development cycle.

Purpose: Guides the overall approach, sequence, and structure of the development process.

Examples: Waterfall, Agile, Spiral.

Focus: Defines the overall methodology and stages of software development.

2. Software Process:

Definition: It's the specific set of activities, tasks, and methods undertaken during the development of software.

Purpose: Involves implementing the detailed steps outlined in the chosen process model to create a software product.

Example: Requirement analysis, coding, testing, deployment.

Focus: Involves executing practical tasks to create the actual software.

How to Improve Software Process:

1. Regular Evaluation
2. User Feedback
3. Training.
4. Automation
5. Collaboration
6. Documentation

Chapter 3
Techniques of Planning, Controlling and Automating Software Process (4Q) (30 marks)

(Asked 2 times)

1. What do you mean by software project planning ? Why is it necessary to plan software projects ?

- Software project planning is the process of creating a detailed roadmap for developing a software.
- It involves setting goals, breaking tasks into smaller parts, estimating time and resources, and creating a schedule to ensure the project is well-organized and successful.

Software project planning is crucial for several reasons:

1. **Direction and Focus**: Planning provides a clear roadmap for the project team, outlining what needs to be done, how, and when. It ensures that everyone is aligned with the project's goals and objectives.

2. **Resource Management**: Planning helps allocate resources effectively, including human resources, time, and budget, to ensure their optimal utilization throughout the project.

3. **Risk Management**: A well-structured plan includes risk assessment and mitigation strategies, helping to identify potential issues in advance and plan for their resolution.

4. **Stakeholder Expectations**: Planning sets realistic expectations for stakeholders regarding project deliverables, timelines, and outcomes, fostering effective communication.

5. **Quality Assurance**: Planning defines quality standards and methodologies, ensuring that the final product meets the required quality criteria.

(Asked 4 times)

2. What broad activities does software project planning include?

Or

What are the broad activities that encompass software planning? List the step involved in detail panning.

- The broad activities included in software planning are:

1. Defining Objectives:
- Clearly stating the project's goals, scope, and objectives.

2. Requirements Gathering:
- Collecting and documenting detailed requirements from stakeholders.

3. Resource Estimation:
- Estimating the resources, such as team members, tools, and budget, required for the project.

4. **Scheduling**:
- Creating a timeline for each task, establishing milestones, and setting project deadlines.

5. Risk Analysis and Mitigation:
- Identifying potential risks and planning strategies to mitigate them.

6. Quality Planning:
- Outlining quality standards, testing procedures, and methods to ensure the final product's quality.

7. Communication Plan:
- Defining how communication will take place among team members, stakeholders, and clients.

8. Change Management:

- Establishing protocols for managing changes to requirements, scope, or project plan.

Q. What are the planning guidelines?

- Planning guidelines in software project management are a set of principles and recommendations that help project managers and teams create effective and well-structured project plans.
- These guidelines ensure that the planning process is thorough, realistic, and aligned with the project's goals and objectives. Here are some key planning guidelines:
- Explain 8 points of que no 2

3. How iteration planning process is done? What are its different steps.

- Iteration planning is a process used in Agile software development to plan and manage work for a specific time period, usually referred to as an "iteration" or "sprint."
- The goal is to prioritize and select user stories or tasks from the product backlog to be completed within the iteration.

- Here are the steps involved in the iteration planning process:

1. Review Backlog:

- Understand product backlog items.
- Prioritize based on importance.

2. Select Work:

- Choose user stories for the iteration.
- Consider team's capacity.

3. Task Breakdown:

- Divide user stories into smaller tasks.
- Specify task details.

4. **Estimate and Plan:**
- Estimate task effort (e.g., story points).
- Arrange tasks, considering dependencies.

5. **Commitment and Agreement:**
- Team commits to completing tasks.
- Share plan with stakeholders for alignment

(Asked 3 times)

4. What is Work Breakdown Structure(WBS)? Explain the importance and limitations of WBS.

- A Work Breakdown Structure (WBS) is a way to break down a complex project into smaller, manageable tasks or components.
- It's like creating a hierarchy of tasks, with each task being broken down into sub-tasks until you reach small, easily achievable pieces of work.
- It helps us understand the project's scope and shows how everything fits together, from big phases to tiny tasks.
- It helps in organizing, planning, and tracking the project's progress.

- Here's how the hierarchy of a WBS typically works:
- **Project**: This is the highest level of the hierarchy, representing the entire project itself.
- **Phases or Major Deliverables**: The project is divided into major phases or deliverables that need to be accomplished. These are significant milestones in the project's progress.
- **Tasks**: Each major deliverable is broken down into smaller task. These tasks are more manageable units of work that contribute to achieving the major deliverable.
- **Sub-Tasks:** Tasks are further broken down into sub-tasks. Sub-tasks are specific actions that need to be performed to complete the task. They are the smallest work units in the hierarchy.

- The hierarchy flows from the general (project) to the specific (sub-tasks).
- Each level is dependent on the level above it.
- This hierarchical structure helps in organizing and understanding the project's scope, tasks, and their interdependencies.

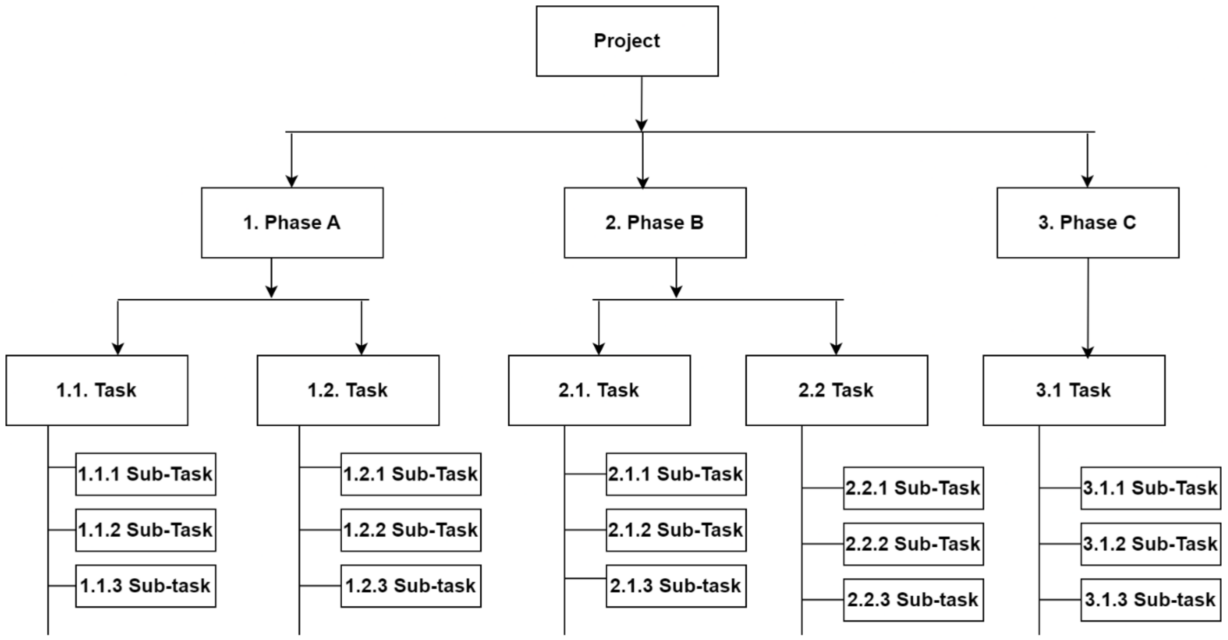


Fig: Work Breakdown Structure

Example:

Project: Develop a New Mobile App

Phase: Design and Development

Task: User Interface Design

Sub-Task: Create Wireframes

Sub-Task: Design Color Scheme

Sub-Task: Design Navigation

Task: Back-end Development

Sub-Task: Set Up Server Environment

Sub-Task: Create APIs

Sub-Task: Implement Database Connectivity

Advantages:

1. **Clarity**: WBS provides a clear visual breakdown of tasks, making it easier to understand project scope and responsibilities.
2. **Organization**: It organizes complex projects into manageable parts, helping teams stay focused on specific tasks.
3. **Communication**: WBS enhances communication by ensuring everyone understands their roles and tasks.
4. **Estimation**: It makes estimating project duration, resources, and costs more accurate as tasks are well-defined.
5. **Tracking**: WBS helps track progress, as completed tasks are checked off, giving a clear view of project advancement.
6. **Resource Allocation**: With tasks divided, resources can be assigned more efficiently, preventing overloading or underutilization.
7. **Risk Management:** WBS allows identifying potential risks early, aiding in planning preventive measures.

Disadvantages:

1. **Time-Consuming**: Creating a detailed WBS can be time-consuming, especially for complex projects.
2. **Overemphasis on Detail**: Too much detail in the WBS can lead to excessive focus on minor tasks, losing sight of the big picture.
3. **Maintenance**: Changes in project scope or tasks require updating the WBS, which can be cumbersome.
4. **Limited Flexibility**: A rigidly defined WBS might struggle to accommodate changes or unexpected events.
5. **Lack of Context**: It doesn't show task relationships, dependencies, or the flow of work.
6. **Misunderstanding**: If not properly communicated, the WBS might be misunderstood, leading to confusion.
7. **Complexity Management**: In large projects, the WBS itself can become complex, making management challenging.

(Asked 2 times)

5. Explain Conventional Work breakdown Structure and Evolutionary Work Down Structures.

1. **Conventional WBS:**

- It's a structured approach where you plan and define all tasks upfront.
- The entire project is broken down into smaller, well-defined tasks from the beginning.
- You create a comprehensive project plan before starting the execution phase.
- This approach suits projects where the scope is well-defined and unlikely to change significantly.
- **Example** in Web Development:
  Imagine you're building a website for an online store. You list out all tasks like designing the homepage, creating product pages, implementing a shopping cart, setting up payment gateways, and testing each functionality before implementation.

2. **Evolutionary WBS:**

- It's a flexible approach that starts with a basic plan and allows for adjustments as the project progresses.
- Instead of planning everything in detail upfront, you start with a basic idea and add more details as you go along.
- It's ideal for projects where requirements might change or when you're working in an agile environment.
- Example in Web Development:
  You're developing a content-heavy website for a news organization. Instead of planning every detail upfront, you start with the main sections like Home, News, Sports, and gradually add sub-sections and features as editors and readers provide feedback. This allows you to respond to changing needs and priorities.

In both approaches, the key is to choose the one that aligns with the project's characteristics and your team's working style.

(Asked 4 times)

6. What is Work Breakdown Structure? What is Responsibility Matrix? How are they related?

Work Breakdown Structure

- A Work Breakdown Structure (WBS) is a way to break down a complex project into smaller, manageable tasks or components.
- It's like creating a hierarchy of tasks, with each task being broken down into sub-tasks until you reach small, easily achievable pieces of work.
- It helps us understand the project's scope and shows how everything fits together, from big phases to tiny tasks.
- It helps in organizing, planning, and tracking the project's progress.

Responsibility Matrix

- A Responsibility Matrix, also known as a RACI matrix, is a tool used to define and communicate roles and responsibilities within a project or process.
- The term "RACI" stands for Responsible, Accountable, Consulted, and Informed.
- It shows who's doing what, who's in charge, who needs to be asked, and who just needs to know.
- With RACI, everyone knows their role, so there's less confusion and better teamwork.
- It helps us avoid mix-ups and ensures everyone knows their part in making the project work.

Relationship between WBS and Responsibility Matrix:

- The WBS outlines the tasks and components of a project, while the Responsibility Matrix defines who is responsible, accountable, consulted, or informed for each task.

- These two tools are closely related and often used together for effective project management:

1. **Assignment of Roles**: The Responsibility Matrix helps assign roles to tasks identified in the WBS. It clarifies who will be responsible for completing each task and who needs to be consulted or informed.

2. **Clear Accountability**: By using both tools together, project managers can ensure that each task in the WBS has a clear owner and that roles and responsibilities are well-defined.

3. **Alignment**: The WBS and Responsibility Matrix ensure that project participants have a shared understanding of what needs to be done, who is involved, and who holds ultimate accountability.

4. **Communication**: Together, these tools facilitate effective communication within the project team, minimizing confusion about roles and responsibilities.

In essence, the WBS provides the structure of the project's tasks, and the Responsibility Matrix assigns accountability and communication roles for each of those tasks, ensuring a well-organized and smoothly managed project.

(Asked 4 times)

7. Cost and Schedule Estimation Process.

Or

Explain forward looking and backward looking.

Or

Explain Top-Down estimating and Bottom Up estimating.

- Cost and Schedule Estimation Process is the systematic method used to predict the amount of time and money required for completing a project.
- It involves analyzing various factors, historical data, and project requirements to provide accurate estimates for both project duration and associated costs.
- This process helps project managers and teams plan and allocate resources effectively, make informed decisions, and ensure that projects are completed within the desired timeframe and budget.

1. Macro Analysis Technique (Top Down Approach or Forward Looking):

- This approach starts with a high-level overview of the project.
- Estimates are made based on historical data, industry benchmarks, and expert judgment.
- You estimate the project as a whole before breaking it down into smaller components.
- It's like looking at the big picture first and then diving into the details.
- Example: Estimating the cost and time for the entire mobile app development based on previous similar projects, industry standards, and expert opinions.

2. Micro Analysis Technique (Bottom Up Approach - Backward Looking):

- This approach focuses on breaking the project into smaller tasks or components.
- You estimate the effort, time, and cost for each individual task.
- These detailed estimates are then aggregated to provide an overall estimate for the project.
- It involves a more thorough analysis of the project's components.
- Example: Estimating hours needed for UI design, coding features, testing, etc., then adding them for a detailed project estimate.

(Asked 6 times)

8. Process Automation Tools

OR

What are the tools needed for process automation.

OR

What are the different tools and components needed to automate the software development process?

- Process automation uses computer tools to make project management tasks quicker and smoother.
- Instead of doing things by hand, software does them automatically.
- Automation saves time by handling repetitive tasks like assigning tasks, sending reminders, and updating progress.
- When teams don't need to do repetitive tasks, they can focus on more important work.
- It helps save time, reduce errors, and ensures consistency in project management activities.

Commonly used tools for automations are:

1. **Project Management Software:**
   - Tools like Microsoft Project, Trello or Jira provide features to automate task assignment, tracking progress, setting deadlines, and generating reports.

2. **Version Control Systems:**
   - Tools like Git automate the process of tracking changes in code, coordinating collaboration among developers, and managing different versions of the software.

3. **Continuous Integration/Continuous Deployment (CI/CD) Tools:**
   - Tools like Jenkins, automate the process of building, testing, and deploying software, ensuring frequent and reliable releases.

4. **Automated Testing Tools:**
   - Tools like Selenium, JUnit, etc automate the testing of software, helping to identify bugs and ensure quality.

5. **Configuration Management Tools:**
   - Tools like Puppet, or Chef automate the deployment and configuration of software and infrastructure.

6. **Workflow Automation Tools:**
   - Tools like Zapier automate the integration and communication between different apps and services used in the project.

7. **Documentation Tools:**
   - Tools like Google Docs provide automation features for creating, sharing, and updating project documentation.

8. **Communication Tools:**
   - Tools like Slack, Microsoft Teams, or Discord automate communication among team members, allowing for real-time collaboration.

9. **Reporting Tools:**
   - Tools like Tableau or Power BI automate the process of generating and visualizing project progress and performance reports.

(Asked 5 times)

10. What is software baseline and its significance? Describe various baselines.

- A software baseline refers to a stable and approved version of software artifacts and associated documentation at a specific point in time during the development process.
- It serves as a reference point for future changes and provides a reliable foundation for further development and quality assurance activities.
- It's a clear starting point that everyone agrees on for future work.
- It helps compare how the project evolves over time.
- There are generally two classes of baseline:

1. External Product Release
2. Internal Testing Releases

1. External Product Release

➢ This baseline represents a significant version of the software that is intended for release to external users or customers.

➢ It includes all the necessary components, features, and documentation required to provide a functional and usable software product to the intended audience.

2. External Testing Release

➢ This baseline is focused on the version of the software that is used for internal testing and quality assurance purposes.

➢ It may not be the final product but serves as a testing ground for identifying and rectifying defects before the external release.

There are three level of baseline Releases.

1. Major Baseline
2. Minor Baseline
3. Interim baseline

1. **Major baseline**

- ➢ A major baseline signifies a significant milestone in the software development process.
- ➢ It indicates a substantial change or enhancement in the software's functionality or architecture.
- ➢ Major baselines often involve fundamental shifts in the software's features or capabilities.

2. **Minor baseline**

- ➢ A minor baseline indicates a smaller update or improvement to the software.
- ➢ It might include bug fixes, minor enhancements, or improvements to existing features.
- ➢ Minor baselines are aimed at maintaining the software's quality and addressing user feedback.

3. **Interim baseline**

- ➢ An interim baseline represents an intermediate version of the software that is not intended for external release but serves specific development or testing purposes.
- ➢ It can be used to address specific issues, conduct focused testing, or implement temporary changes.

The significance of software baselines is:

1. **Controlled change**: Baselines provide a reference to manage and control changes effectively.
2. **Version Tracking**: Baselines track different software versions for historical records.
3. **Quality Assurance**: Baselines help assess software quality and performance accurately.
4. **Risk Management**: Baselines aid in identifying deviations and managing risks.
5. **Clear Communication**: Baselines ensure all stakeholders refer to the same software version.
6. **Support and Maintenance**: For troubleshooting, baselines identify the software version, enabling quicker issue resolution.

Q. Explain process automation with 4 important environment disciplines.

- Process automation is using tools and technology to make software development tasks easier and faster.
- It's like using machines to do work that humans used to do by hand.
- Four important environment disciplines are:

1. **Round-trip Engineering**:

- It's like keeping the design and code in sync automatically.
- Changes in one are reflected in the other, so things stay consistent.

2. **Change Environment**:

- This is about keeping track of changes in the software.
- Imagine having a system to ask for and approve changes, so everyone knows what's happening.
- Configuration Control Board (CCB) reviews and approves changes to maintain control.

3. **Infrastructures**:

- Think of it as having a toolbox with all the tools you need for building software.
- This toolbox includes templates, guides, and resources that make the process smoother.

4. **Stakeholder Environments**:

- This is about making sure everyone involved in the project understands what's happening.
- It's like making sure all the players in a game are on the same page.

Q. Explain about round trip engineering: (Asked 2 times)

- Keep code and design models in sync automatically.
- In round trip engineering code generation and reverse engineering take place at the same time in a dynamic environment.
- Round-trip engineering is like keeping your code and design drawings in sync automatically.
- When you make changes to the code, the design gets updated, and vice versa.
- It helps developers work seamlessly between code and diagrams, making software development easier to manage and maintain.

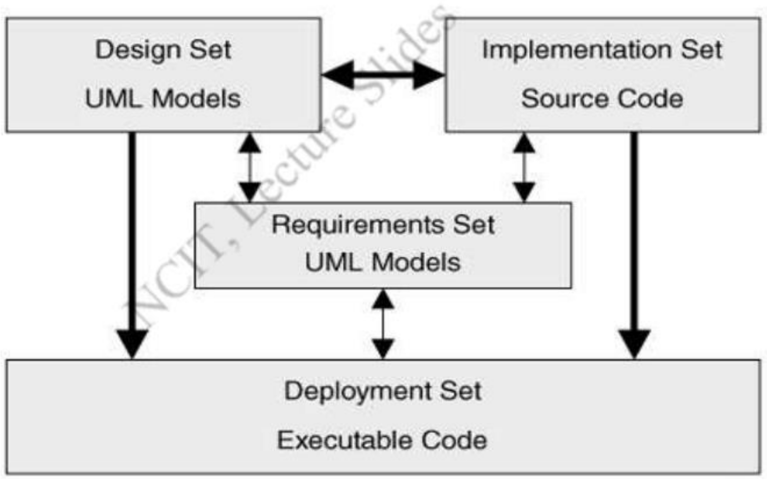


Fig: Round trip engineering

Q. Explain Configuration Control Board.

- A configuration control board (CCB) is like a team of decision-makers for software projects.
- A CCB is a group of knowledgeable people from different parts of the project. They come together to make important decisions about the software.
- When there's a proposed change in the software, the CCB reviews it. They decide if the change is good or if it could cause problems.
- The CCB makes sure that changes are necessary and won't disrupt the project. They weigh the benefits against the risks.
- They keep track of all the changes and make sure they are properly documented. This helps avoid confusion.
- The CCB helps maintain the quality of the software by making sure only approved changes are allowed. This prevents mistakes from sneaking in.

Q. Distinguish Change Control and version Control.

Change Control:

- Change control focuses on managing and approving changes to the software during its development lifecycle.
- It involves assessing and deciding whether proposed changes should be implemented to avoid disruptions.
- Applied to changes in requirements, design, features, and other aspects of the project.
- If a client wants to add a new feature to the software, change control evaluates the impact and decides if it's feasible.
- The goal is to maintain stability and ensure that changes don't negatively affect the project's progress.
- Example tools: Trello and Jira.

Version Control:

- Version control manages different versions or copies of the software's source code and related files.
- It's used to track changes to the code and manage collaborative development.
- Applied to source code, documents, and other files that make up the software.
- Developers use version control to track their changes, collaborate with others, and revert to previous versions if needed.
- It helps maintain a history of changes, enables collaboration, and ensures that everyone is working with the same codebase.
- Example tools: Git and subversion

Q. Explain different levels of automation.

- There are three levels of automation.

1. Meta-level Automation:

- Organizational strategy and long-term planning.
- Establishing an infrastructure with tools and guidelines.
- Enabling consistent practices across projects.
- Aligning software development with business goals.
- Focuses on maximizing ROI and overall efficiency.

2. Macro-level Automation:

- Project-specific policies and practices.
- Use of tools for requirements management, testing, etc.
- Creating an environment to support project execution.
- Ensures adherence to project constraints and goals.
- Facilitates collaboration and controlled development.

3. Micro-level Automation:

- Day-to-day activities of the project team.
- Tools for coding, debugging, testing, etc.
- Reducing manual effort and errors.
- Improving individual and team productivity.
- Enhancing the quality and consistency of work.

Q. Explain followings:

1. PERT (Program Evaluation and Review Technique):

- PERT is a project management tool used to schedule, organize, and coordinate tasks in a project.
- It involves identifying critical paths, estimating task durations, and managing uncertainties.
- PERT helps in optimizing project timelines and resources allocation.
- It's commonly used for complex projects with interdependent tasks.

2. CASE (Computer-Aided Software Engineering):

- CASE refers to a set of tools and software used to support various phases of software development.
- It assists in requirements analysis, design, coding, testing, and maintenance.
- CASE tools enhance collaboration among development teams and improve software quality.
- These tools automate manual tasks, reducing errors and speeding up the development process.

11. What are the various seven core metrics. Elaborate them.

OR

Explain all core metrics.

- Core metrics are essential measurements used to understand and track different aspects of a software project.
- They help us keep an eye on how the project is progressing, whether it's meeting its goals, and if the quality of the work is up to the mark.
- These metrics provide important information that guides decisions and helps the project stay on the right track.
- There are seven core metrics that should be used in iterative development.

It is divided into in two indicators:

1. Management Indicators
   1. Work and Progress
   2. Budgeted Cost and Expenditure
   3. Staffing and team Dynamics
2. Quality Indicators
   1. Change Traffic and Stability
   2. Breakage and Modularity
   3. Rework and Adaptability
   4. Mean Time Between Failure and Maturity

1. **Management Indicators**
   i. **Work and Progress:**
      - Measures the amount of work completed over time.
      - Tracks the progress of tasks, activities, and deliverables.
      - Helps in assessing project advancement and potential delays.

   ii. **Budgeted Cost and Expenditure:**
      - Monitors the cost incurred over time against the budget.
      - Assists in controlling expenses and managing financial resources.
      - Provides insights into the project's financial health.

   iii. **Staffing and Team Dynamics:**
      - Observes changes in team composition over time.
      - Tracks additions, departures, and role shifts of team members.
      - Helps in understanding team dynamics and addressing staffing needs.

2. **Quality Indicators:**

   iv. **Change Traffic and Stability:**

- Examines the frequency of changes made to the software.
- Measures the stability of the project by tracking changes.
- Aids in assessing the impact of frequent changes on stability.

   v. **Breakage and Modularity:**
- Calculates the average breakage (failures) per change made.
- Focuses on the impact of changes on the system's modularity.
- Helps identify design flaws affecting system reliability.

   vi. **Rework and Adaptability:**
- Assesses the average rework required per change.
- Indicates how adaptable the system is to changes.
- Highlights areas that need improvement for better adaptability.

   vii. **Mean Time Between Failure (MTBF) and Maturity:**
- Measures the average time between occurrences of failures.
- Reflects the system's maturity and reliability over time.
- Indicates the overall quality and stability of the software.

These core metrics provide essential insights into the management and quality aspects of iterative software development. By monitoring these indicators, project teams can make informed decisions, enhance productivity, and ensure project success.

(Asked 2 times)

12. What do you mean by project organization? Roles and responsibilities of project organization.

- Project organization refers to the structured arrangement of individuals and teams responsible for carrying out a project.
- It defines roles, responsibilities, communication channels, and reporting relationships within a project.

Roles and Responsibilities of Project Organization:

1. **Project Manager**: Overall leader responsible for planning, executing, and closing the project. Manages resources, budget, and risks.
2. **Project Team:** Individuals responsible for specific tasks and deliverables. Includes developers, designers, testers, etc.
3. **Stakeholders:** Those with an interest in the project's outcome. Can include clients, end-users, sponsors, and management.
4. **Business Analyst:** Gathers and analyzes requirements, communicates with stakeholders to understand needs.
5. **Technical Lead:** Guides technical decisions, supervises developers, ensures adherence to best practices.
6. **Quality Assurance:** Ensures quality by testing and verifying that deliverables meet requirements.
7. **Risk Manager:** Identifies potential risks and develops strategies to mitigate them.
8. **Documentation Manager:** Responsible for maintaining project documentation and knowledge sharing.
9. **Support Team:** Provides post-implementation support and maintenance.

13. Explain Line of Business organization with different roles and responsibilities.

- A Line of Business (LOB) organization is a type of organizational structure where teams are formed based on specific business functions or areas of expertise.
- Each line of business focuses on a particular product, service, or market segment.
- Line of Business (LOB) organization typically has a hierarchical structure, with various roles organized in a hierarchical order.
- The hierarchy allows for clear lines of authority, reporting, and decision-making within each line of business.
- This hierarchical structure ensures that each role has well-defined responsibilities and is accountable to higher-level roles.

The roles and responsibilities are:

1. Head manager
2. Product manager
3. Sales and marketing team
4. Research and Development Teams
5. Operation Teams
6. Customer Support Team
7. Finance and Accounting Team
8. QA Team

Q. What are the different ways to organize a software project team.

1. **Functional Teams**: Team members are grouped by their functional roles (e.g., development, testing, design).
2. **Cross-Functional Teams**: Team members from different functions collaborate closely on the project.
3. **Project-Based Teams**: Assembled specifically for a particular project and disbanded after its completion.
4. **Virtual Teams**: Team members work remotely, often from different locations, using digital collaboration tools.
5. **Dedicated Teams**: Team members work exclusively on the project without distractions from other tasks.

Q. Describe the balanced, functional and project matrix organizational structure.

1. **Balanced Matrix Structure:**

- In a balanced matrix structure, power and authority are shared equally between the functional manager (department head) and the project manager.
- Employees report to both managers.
- Employees have two bosses - one from their functional department and one from the project team.
- Both managers collaborate and make decisions together, leading to a balanced approach.
- Resources like employees and equipment are shared between projects and functional areas.
- This structure allows for flexibility in resource allocation and project management.

2. **Functional Matrix Structure:**

- In a functional matrix structure, the functional manager holds more authority, and project managers act as coordinators or facilitators.
- Employees mainly report to their functional manager.
- The functional manager has greater influence on decisions, and projects align with the functional goals.
- Project managers coordinate activities and resources across departments, but they don't have full authority.
- Functional managers control resources, which can lead to competition for resources between projects.
- Employees in functional departments build expertise in their domain.

3. Project Matrix Structure:

- In a project matrix structure, project managers have significant authority, and project teams are separate from functional departments.
- Employees have dual roles in projects and their functional departments.
- Project managers have strong decision-making power, and they manage their teams independently.
- Project managers control project-specific resources, reducing resource conflicts.
- Teams work closely together, focused on project goals and deadlines.
- Employees balance their project responsibilities with their functional roles.
- Direct communication within project teams speeds up decision-making.

14. Explain the software assessment team activities in simple words and points.

- The software assessment team is responsible for evaluating and improving the software development processes within an organization.
- Their activities involve:

1. **Process Evaluation**: Assessing current software development processes to identify strengths and weaknesses.

2. **Defining Standards:** Establishing best practices and standards for software development across the organization.

3. **Gap Analysis**: Identifying gaps between current practices and industry standards, and proposing solutions to bridge those gaps.

4. **Training and Education**: Providing training and educational resources to help teams adopt new practices and improve their skills.

5. **Monitoring Progress**: Tracking the implementation of new processes and measuring their impact on project outcomes.

6. **Feedback Collection**: Gathering feedback from development teams to continuously refine and improve processes.

7. **Risk Management**: Identifying potential risks and vulnerabilities in software projects and suggesting mitigation strategies.

8. **Documentation**: Ensuring that processes, standards, and improvements are well-documented for future reference.

9. **Communication**: Facilitating communication between different teams and stakeholders to ensure everyone is aligned with the changes.

(Asked 2 times)

15. What are the activities of the software architecture team? What are their expertise?

- The software architecture team is responsible for designing the overall structure and framework of a software system.
- Their activities and expertise include:

Activities:

1. **System Design**: Creating the high-level structure of the software system, including components, modules, and their interactions.

2. **Requirements Analysis**: Understanding and translating user needs into architectural specifications.

3. **Choosing Patterns**: Selecting design patterns and architectural styles that best fit the project's requirements.

4. **Collaboration**: Working closely with stakeholders, developers, and other teams to ensure a cohesive architecture.

5. **Documentation**: Creating clear and comprehensive architectural documentation to guide development.

6. **Reviewing Designs**: Reviewing and providing feedback on designs proposed by development teams.

7. **Technology Evaluation**: Assessing and recommending technologies and tools for implementing the architecture.

Expertise

1. Technical Proficiency
2. Design patterns
3. Communication Skills
4. Problem Solving
5. Domain Knowledge
6. Risk Assessment

(Asked 2 times)

16. How would you tackle the issue related to management of complex software?

- Tackling the management of complex software involves several strategies to ensure successful development and delivery.

- Here's how you can approach this challenge:

1. **Clear Requirements Gathering**: Start by thoroughly understanding the project requirements and objectives. Engage stakeholders to gather clear and complete specifications.
2. **Break Down Complexity**: Divide the software into manageable components or modules. This helps in handling complexity in smaller, more understandable parts.
3. **Effective Project Planning**: Develop a detailed project plan that outlines tasks, timelines, resources, and dependencies. A well-structured plan can streamline complex tasks.
4. **Experienced Team:** Build a team with diverse skills and experience. A team with expertise in different areas can collectively address complex challenges.
5. **Iterative Development:** Adopt an iterative development approach. Break the project into smaller phases, allowing regular testing, feedback, and adjustments as needed.
6. **Communication and Collaboration**: Maintain open communication among team members, stakeholders, and users. Collaboration helps in resolving complex issues more effectively.
7. **Version Control and Documentation**: Implement version control for source code and maintain detailed documentation. This ensures everyone is on the same page and simplifies tracking changes.
8. **Quality Assurance:** Implement thorough testing and quality assurance processes. Identify and fix issues early to prevent them from escalating into complex problems.
9. **Risk Management**: Identify potential risks and create mitigation strategies. Addressing risks early can prevent them from becoming major roadblocks.

(Asked 2 times)

17. Suppose you are a project manager of a large development project. The top management informs you would have to do with a fixed team size (i.e. constant number of engineers) throughout the duration of your project. What will be the impact of this decision on your project? Explain your answer.

- The decision to work with a fixed team size throughout the project can have several impacts on the project's dynamics, timeline, and overall success.
- Here's how this decision might affect the project:

1. **Resource Limitation**:

- With a fixed team size, you'll have a limited number of engineers to work on the project.
- This can potentially slow down development, as there might be limitations on how many tasks can be worked on simultaneously.

2. **Workload Distribution:**

- The fixed team size could lead to challenges in distributing the workload evenly.
- Some team members might become overloaded with tasks, while others might have relatively lighter workloads.
- This can impact productivity and lead to burnout among some team members.

3. **Project Schedule:**

- The project's schedule might be affected, especially if there are time-sensitive tasks that require a quick turnaround.
- With a fixed team size, you might need to carefully prioritize tasks to ensure critical deadlines are met.

4. Flexibility:

- The ability to respond to changes in project requirements or scope might be limited.
- Changes could lead to a strain on resources or require shifting priorities, potentially affecting the project's overall direction.

5. Risk Management:

- Addressing unforeseen challenges or issues might become more difficult due to resource constraints.
- The project manager will need to plan for potential risks and have contingency measures in place.

6. Innovation and Quality:

- A fixed team size might limit the capacity for innovation and quality improvement.
- There might be fewer resources available for research, testing, and exploring alternative solutions.

7. Communication and Collaboration:

- A smaller team might lead to stronger collaboration among team members, but it could also limit the diversity of skills and perspectives available for problem-solving.

8. Project Scope:

- The project's scope might need to be managed carefully to align with the available resources.
- It might not be feasible to accommodate additional features or functionalities without extending the timeline.

Q. Explain peer inspection.

- Peer inspection, also known as peer review, is a collaborative process used in software development to ensure the quality and correctness of work.
- It involves team members reviewing each other's work. This can include source code, design documents, requirements, and other artifacts.
- It encourages knowledge sharing among team members.

Chapter 4
Modern Approach to Software Project and Economics (1 or 2 Q) (8-15 marks)

(Asked 2 times)

1. Explain the essential elements for successful software development.

Successful software development requires several essential elements to ensure a smooth and effective process. Here are the key factors:

1. **Clear Objectives and Requirements**:
- ➢ Having a well-defined goal and understanding the requirements of the software is crucial.
- ➢ Clear objectives help guide the development process and keep the team focused.

2. Effective Planning:
- ➢ Thorough project planning involves setting timelines, allocating resources, and outlining tasks.
- ➢ A well-structured plan helps manage expectations and ensures the project stays on track.

3. **Skilled Team:**
- ➢ A capable and well-coordinated team with the necessary skills and expertise is vital.
- ➢ Each team member should understand their roles and responsibilities.

4. Communication:
- ➢ Effective communication among team members, stakeholders, and users is essential.
- ➢ Regular updates, feedback loops, and addressing concerns help maintain alignment and resolve issues promptly.

5. **Risk Management:**
- Identifying potential risks and planning strategies to mitigate them is important.
- This prevents unexpected hurdles that can disrupt the development process.

6. **Iterative Development:**
- Adopting an iterative approach allows for continuous improvement and adaptation based on feedback.
- This helps in delivering a more refined and user-friendly product.

7. **Quality Assurance:**
- Implementing testing and quality control processes ensures the software meets the desired standards, functions correctly, and is free of major defects.

8. **Documentation:**
- Comprehensive and well-organized documentation helps in understanding the software, its design, functionality, and maintenance needs.

2. Explain the elements of modern software project management principles and their constraints.

Modern software project management principles encompass various elements that guide effective project execution.

Here are the key elements along with their constraints:

1. **Agile Methodology:**
   - Agile is about flexibility and teamwork. It focuses on adapting to changes and involving customers throughout the process.
   - **Constraint**: Agile needs constant feedback from stakeholders, which can be hard in complex projects.

2. **Iterative Development:**
   - This means working on the project in small steps, learning from each step, and making improvements as you go.
   - Constraint: Working in iterations might lead to adding too much work or not seeing the whole project clearly.

3. **Customer-Centric Approach:**
   - This means putting customers at the center. Their needs are the priority, and they're involved in decisions.
   - Constraint: It's challenging to balance what customers want with project limits like time and resources.

4. **Continuous Delivery:**
   - Continuously giving working parts of the software to customers so they can see progress and provide feedback.
   - **Constraint**: Frequent releases need careful testing to make sure each update works well.

5. **Risk Management:**
   - Identifying problems that could happen and planning to handle them before they affect the project.
   - Constraint: You can't predict every problem, and some issues might still come up unexpectedly.

6. **Collaboration and Communication:**
- Working together and sharing information among team members and stakeholders.
- Constraint: Miscommunication or not working well together can slow things down or cause confusion.

7. **Quality Focus:**
- Making sure the software is good and useful by checking its quality at every step.
- **Constraint**: Finding the balance between quality and the time and resources available can be tough.

8. **Empowerment and Autonomy:**
- Giving team members the freedom to make decisions and take responsibility for their work.
- **Constraint**: Giving too much freedom might lead to confusion or lack of coordination.

3. Explain the best practices of software management with examples.

The best practices of software management are:

1. Formal Risk Management:

- Identifying potential problems and planning to deal with them.
- Example: Predicting that a new software component might take longer to develop due to unfamiliar technology. The team plans extra time for research and testing.

2. Agreement on Interfaces:

- Clear understanding of how different parts of the software will work together.
- Example: Making sure that the user login system connects smoothly with the user profile system for consistent user data.

3. Formal Inspections:

- Regularly checking work to find and fix problems early.
- Example: Reviewing code before integration to catch errors or bugs before they cause issues.

4. Metric-based Scheduling and Management:

- Using data to plan and track project progress.
- Example: Tracking task completion times to estimate project completion.

6. Program-wide Visibility of Progress versus Plan:

- Sharing project progress compared to the initial plan.
- Example: Regular updates on software development status and schedule alignment.

7. Defect Tracking Against Quality Targets:

- Keeping track of mistakes found and ensuring fixes meet quality goals.
- Example: Addressing user-reported errors to ensure proper software functionality.

8. **Configuration Management:**

- Organizing and tracking software changes during development.
- Example: Using version control to track code changes and maintain the latest version.

9. People-aware Management Accountability:

- Ensuring team members understand responsibilities and receive support.
- Example: Team members know their roles and can seek help when needed.

(Asked 3 times)

4. Object oriented software engineering is rapidly displacing conventional software development approaches. Justify your answer.

OR

Explain how the management of an object oriented development project would differ from a traditional project.

- Object-Oriented Software Engineering (OOSE) is gaining popularity over conventional software development due to its structured approach and various benefits.
- Managing an object-oriented development project differs from a traditional one across different aspects and are explained below:

1. **Design and Architecture:**

- OOSE: Focuses on creating modular, reusable components that interact through well-defined interfaces.
- Traditional: Often follows a monolithic approach with less emphasis on componentization and modularity.

2. **Team Collaboration:**

- OOSE: Encourages collaboration among specialized team members, each responsible for specific components or classes.
- Traditional: May involve more siloed work where each team handles specific parts without close interaction.

3. **Abstraction and Encapsulation:**

- OOSE: Leverages abstraction to hide complexity and encapsulation to protect data integrity.
- Traditional: Might lack the same level of abstraction and encapsulation, potentially leading to data leaks and complex code.

4. **Code Reusability:**

- OOSE: Promotes reusable classes and components, reducing redundant coding efforts.
- Traditional: Can lead to duplicated code due to less emphasis on reusability.

5. **Software Design Patterns:**

- OOSE: Utilizes design patterns for solving common software design problems.
- Traditional: May not leverage design patterns as extensively, potentially leading to ad hoc solutions.

6. **Change Management:**

- OOSE: Adaptable to changes in requirements due to its modular and flexible nature.
- Traditional: Can be less adaptable to changes, requiring more effort for modifications.

7. **Quality Assurance:**

- OOSE: Enhances maintainability and quality through structured design and testing.
- Traditional: Might require additional efforts to ensure code quality and maintainability.

8. **Documentation:**

- OOSE: Emphasizes documenting relationships between components, classes, and objects.
- Traditional: Documentation might be less structured and detailed.

In conclusion, object-oriented software engineering offers a more structured and adaptable approach compared to conventional methods. Its emphasis on modularity, abstraction, and design patterns contributes to more efficient development. Managing an object-oriented project requires coordinating specialized teams, focusing on code reusability, and ensuring adherence to design principles. This approach aligns well with the dynamic and evolving nature of modern software development.

(Asked 2 times)

5. Explain the cultural shift in terms of modern process transition.

OR

Paradigm Shift

- A cultural shift refers to a significant change in the way projects are approached, managed, and executed due to evolving cultural norms, values, and practices.
- This shift impacts how processes are designed, implemented, and embraced by the workforce, and it often aligns with contemporary trends and expectations.
- It signifies a departure from traditional methods and a move towards embracing new approaches that align with the changing cultural landscape.
- Cultural shifts play a crucial role in software project management, as they can impact team dynamics, communication, collaboration, and overall project success.

When discussing cultural shifts in terms of modern process transition, several aspects come into play:

1. **Embracing Agility:**
- **Explanation**: Organizations are open to flexible and collaborative methods like Agile.
- **Impact**: Processes adapt quickly to changes.

2. **Open Communication:**
- **Explanation**: Teams openly share ideas and feedback.
- **Impact**: Better collaboration and more innovative solutions due to a culture of transparent communication.

3. **Flat Hierarchies:**
- **Explanation**: Decision-making is spread across teams, not just top management.
- **Impact**: Everyone has a say, aligning with an inclusive culture.

4. **Empowerment and Ownership:**
- **Explanation**: Employees take charge of their work and decisions.
- **Impact**: Teams are motivated and accountable in a culture of ownership.

5. **Continuous Learning:**
- **Explanation**: Focus on ongoing learning and skill development.
- **Impact**: A culture of improvement and staying updated with changes.

6. **Adapting to Change:**
- **Explanation**: Being open to process changes based on feedback and trends.
- **Impact**: Processes stay relevant, reflecting a culture of adaptability.

7. **Customer-Centric Approach:**
- **Explanation**: Centering processes around customer needs and feedback.
- **Impact**: Products and services match user expectations due to a culture of understanding customers.

8. **Collaborative Problem-Solving:**
- **Explanation**: Teams work together to solve challenges.
- **Impact**: Effective solutions arise from diverse inputs, reflecting a culture of collaboration.

(Asked 2 times)

6. Cultural shifts play an important role. What are the different indicators which has to be taken into consideration?

- Cultural shifts indeed play a crucial role in software project management, as they can impact team dynamics, communication, collaboration, and overall project success.
- When considering cultural shifts, various indicators should be taken into consideration to ensure a smooth and effective project execution.
- These indicators include:

1. **Communication Styles**:
- Understand how team members prefer to communicate, whether it's through direct conversations, written documentation, or digital platforms.
- Ensure that communication channels align with the cultural preferences of team members.

2. Hierarchy and Decision-Making:
- Cultural norms often influence the hierarchy within a team and how decisions are made.
- Recognize whether decision-making is centralized or distributed, and adapt project management practices accordingly.

3. Time Orientation:
- Different cultures have varying perspectives on time, such as being future-oriented, present-oriented, or past-oriented.
- Recognize how deadlines and time commitments are perceived and managed by team members.

4. Work-Life Balance:
- Some cultures emphasize a strong work-life balance, while others prioritize longer working hours.
- Acknowledge these differences and establish expectations for availability and work hours.

5. **Team Collaboration:**

- Assess the level of collaboration and cooperation expected within the team.
- Some cultures emphasize teamwork , while others may value individual contributions more.

6. **Conflict Resolution:**

- Different cultures have distinct approaches to addressing conflicts.

7. **Training and Orientation:**

- Provide training or orientation sessions that address cultural differences and encourage open discussions about cultural perspectives within the team.

(Asked 3 times)

7. How would cultural shifts managed successfully to a modern software management process adjusted? List the different points.

- Managing cultural shifts successfully while adjusting to modern software management processes involves several key points:

1. **Clear Communication:**

- Communicate the reasons behind the cultural shift and the benefits it brings to both the organization and individuals.
- Address concerns and provide regular updates.

2. **Employee Involvement:**

- Involve employees in the process.
- Seek their input, listen to their feedback, and involve them in shaping the new cultural norms.

3. **Training and Education:**

- Provide training to help employees understand the new processes and methodologies.
- Offer opportunities for skill development and continuous learning.

4. **Gradual Transition:**
   - Introduce cultural changes gradually rather than abruptly.
   - This allows employees to adapt at their own pace and reduces resistance.

5. **Celebrate Successes**:
   - Recognize and celebrate milestones and successes achieved through the cultural shift.
   - This reinforces positive behaviors and motivates others to embrace the changes.

6. **Provide Resources**:
   - Ensure that employees have the resources they need to adapt to the new processes.
   - This includes tools, training materials, and support.

7. **Feedback Mechanisms:**
   - Establish mechanisms for employees to provide ongoing feedback about the cultural shift and its impact.
   - Use this feedback to make adjustments as needed.

8. **Monitor and Adjust:**
   - Continuously monitor the progress of the cultural shift.
   - Be prepared to make adjustments based on feedback and outcomes.

9. **Persistence:**
   - Cultural shifts take time.
   - Be patient and persistent in reinforcing the new behaviors and values.

Q. What is software risk?

- Risk in software projects refers to the possibility of something going wrong that could affect the successful completion of the project.
- They include things that might go wrong, like delays, budget overruns, or technical problems.
- It's like identifying potential problems before they happen so you can plan to avoid or handle them.
- Identifying risks helps you plan how to manage or mitigate them to reduce their impact.
- By addressing risks beforehand, you can be better prepared and increase your project's chances of success.

Q. What is Risk management?

- Risk management is the process of:
  Identifying: Recognizing potential uncertainties and challenges that could affect a project.
  Assessing: Evaluating the impact and likelihood of these risks on the project's success.
  Mitigating: Developing strategies to minimize or avoid negative impacts and ensure project success.

Q. What are the top ten risks in software management?

The top ten risks are:

1. Unclear Requirements
2. Scope Creep
3. Schedule Delays
4. Budget Overruns
5. Technical Challenges
6. Lack of Resources
7. Communication Breakdown
8. Risk Management
9. Team Collaboration
10. Changing Requirements

Q. How will you identify risks in the context of software projects.

- Identifying risks in a software project involves systematically looking for potential problems that could impact the project's success.
  1. Brainstorming
  2. Past Experience
  3. Expert Advice
  4. Risk Categories
  5. Feedback
  6. External factors

Q. How is the risk evaluated in software projects?

- Risk evaluation in a software project involves assessing the potential impact and likelihood of identified risks to determine which risks need more attention and planning.
- They are:

  1. Impact assessment

  2. High-priority risks

  3. Expert Judgement

  4. Review and Update

Q. Explain any two methods of risk reduction technique.

- Two risk reduction techniques are:

1. Prototyping:
- Build a simplified version of the software early on.
- Stakeholders can see and test the system's design and functionality.
- Identify issues and gather feedback before full development.
- Reduces the risk of delivering a product that doesn't meet user needs.

2. Parallel Development:
- Multiple teams work on different project parts at the same time.
- Minimizes bottlenecks and dependencies.
- Accelerates development and ensures timely delivery.
- Reduces the risk of delays due to sequential work.

Q. Explain the risk management/mitigation activities.

The risk management activities are divided into two categories:

1. **Risk Assessment:**

   a. **Risk Identification:**
      - Identify possible issues that could affect the project.
      - Consider various factors that might cause problems.

   b. **Risk Analysis:**
      - Examine the identified risks closely.
      - Understand how severe they could be and how likely they are to happen.

   c. **Risk Prioritization:**
      - Rank risks based on their potential impact and likelihood.
      - Focus resources on the most critical risks.

2. **Risk Control:**

   a. **Risk Management Planning:**
      - Create a detailed plan to tackle identified risks.
      - Lay out strategies to handle risks and assign roles.

   b. **Risk Monitoring:**
      - Keep an eye on the project's progress.
      - Track the identified risks and watch for new ones.

   c. **Risk Resolution:**
      - When risks happen, put the planned strategies into action.
      - Implement mitigation plans or fallback strategies.