GUJARAT TECHNOLOGICAL UNIVERSITY

BE - SEMESTER–VII (NEW) EXAMINATION – WINTER 2021

MARKS

Q.1 (a) Why is Software Project Management important? **03**

Software project management is important because it helps to ensure that a software project is completed on time, within budget, and to the required level of quality. It also helps to improve communication among team members, identify and manage risks, and make the best use of resources. By having a clear plan in place, software project managers can help to ensure that the project stays on track and meets its objectives. Additionally, software project management helps to provide transparency and accountability for the work being done, which can be especially important in a large or complex project.

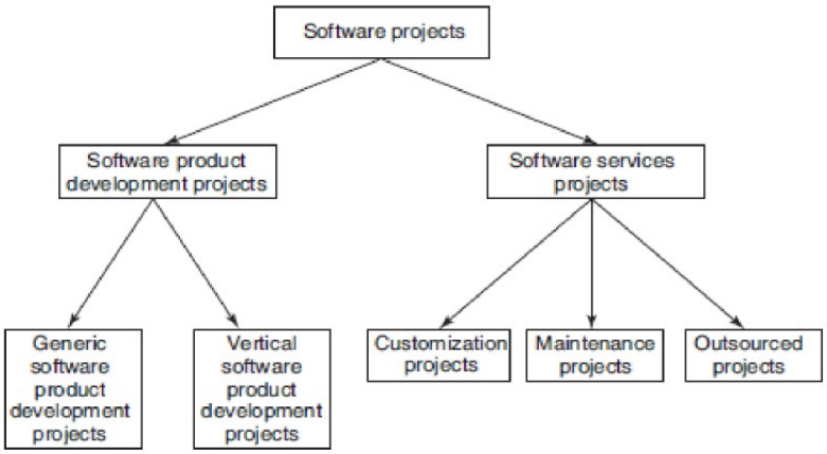
(b) Discuss ways of Categorizing Software. **04**

Ways of Categorizing Software Projects

• Different characteristics of a project could affect the way in which it should be planned and managed.

Some of these are:

– Compulsory vs voluntary users
– Information systems vs embedded systems
– Software products vs software services
– Objective driven development vs product driven development.



(c) What is a project? Discuss activities covered by software project management. **07**

Software Project Management consists of many activities, that includes planning of the project, deciding the scope of product, estimation of cost in different terms, scheduling of tasks, etc.

The list of activities are as follows:

1. Project planning and Tracking
2. Project Resource Management
3. Scope Management
4. Estimation Management
5. Project Risk Management
6. Scheduling Management
7. Project Communication Management
8. Configuration Management

1. Project Planning: It is a set of multiple processes, or we can say that it a task that performed before the construction of the product starts.

2. Scope Management: It describes the scope of the project. Scope management is important because it clearly defines what would do and what would not. Scope Management create the project to contain restricted and quantitative tasks, which may merely be documented and successively avoids price and time overrun.

3. Estimation management: This is not only about cost estimation because whenever we start to develop software, but we also figure out their size(line of code), efforts, time as well as cost.

- Size of software
- Quality
- Hardware
- Communication
- Training
- Additional Software and tools
- Skilled manpower

4. Scheduling Management: Scheduling Management in software refers to all the activities to complete in the specified order and within time slotted to each activity. Project managers define multiple tasks and arrange them keeping various factors in mind.

5. Project Resource Management: In software Development, all the elements are referred to as resources for the project. It can be a human resource, productive tools, and libraries.

6. Project Risk Management: Risk management consists of all the activities like identification, analyzing and preparing the plan for predictable and unpredictable risk in the project.

7. Project Communication Management: Communication is an essential factor in the success of the project. It is a bridge between client, organization, team members and as well as other stakeholders of the project such as hardware suppliers.

8. Project Configuration Management: Configuration management is about to control the changes in software like requirements, design, and development of the product.

Q.2 (a) What is Software Prototyping? **03**

Software prototyping is the process of creating a preliminary model of a software system. It is an iterative process that involves creating a simplified version of the software, testing it, and refining it based on the feedback received.

Software prototyping can be an effective way to quickly test ideas and gather feedback early in the development process. It can help to identify and resolve issues before they become more difficult and expensive to fix later on. It can also be a useful way to communicate ideas to stakeholders and get buy-in for the project.

(b) Outline the steps of Project Planning. **04**

Here are the general steps involved in project planning:

1. Define the project: The first step in project planning is to define the project. This includes understanding the business case and objectives, as well as identifying the stakeholders and their needs and expectations.
2. Create a project plan: A project plan is a detailed document that outlines the steps and resources needed to complete the project. It includes the project scope, schedule, budget, and resources.
3. Identify risks: During the planning phase, it is important to identify and assess potential risks that could impact the project. This includes identifying the likelihood of the risk occurring and the potential impact on the project.
4. Allocate resources: Once the project plan has been created, resources (such as personnel, equipment, and materials) need to be allocated to the various tasks and activities outlined in the plan.
5. Assign tasks: The next step is to assign specific tasks to team members and make sure that everyone knows what is expected of them.
6. Monitor and control progress: The project plan should be used as a tool to track progress and make any necessary adjustments to keep the project on track.
7. Review and evaluate: At the end of the project, it is important to review and evaluate the project to identify any lessons learned and ways to improve future projects.

(c) Discuss function point analysis with suitable example. **07**

Function point analysis is a software measurement technique used to estimate the size and complexity of a software system. It was developed in the 1970s as a way to standardize the measurement of software size, and it is still widely used today.

Function points are a measure of the functionality provided by a software system. They are calculated by counting the number of user inputs, user outputs, and user inquiries (also known as transactions) in the system, and weighting them based on their complexity. The resulting function point count can then be used to estimate the size of the software and the effort required to develop it.

For example, consider a simple registration system for a website. The system includes a form for users to enter their name, email address, and password, and a submit button to send the information to the server. There are also two user outputs: a confirmation message that appears after the form is submitted, and an error message that appears if the form is not filled out correctly.

To calculate the function points for this system, we would count the number of user inputs (1), user outputs (2), and user inquiries (0). Based on the complexity of each type of function, we might assign weights of 3, 4, and 3 respectively. The total function point count for the system would then be (1 x 3) + (2 x 4) + (0 x 3) = 11 function points.

Function point analysis can be a useful tool for estimating the size and complexity of a software system, but it is important to note that it is only an estimate and may not be accurate in all cases. Factors such as the experience of the development team and the complexity of the system can impact the accuracy of the function point estimate.

OR

(c) Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e **number of Lines of Code**. It is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality. It was proposed by Barry Boehm in **07**

1981 and is based on the study of 63 projects, which makes it one of the best-documented models. The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort & Schedule:

- **Effort:** Amount of labor that will be required to complete a task. It is measured in person-months units.
- **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, and months.

Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of the constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below. Boehm's definition of organic, semidetached, and embedded systems:

1. **Organic –** A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
2. **Semi-detached –** A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, and knowledge of the various programming environment lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered of Semi-Detached types.
3. **Embedded –** A software project requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.
    1. Basic COCOMO Model
    2. Intermediate COCOMO Model
    3. Detailed COCOMO Model
4. **Basic Model –**

1. The above formula is used for the cost estimation of for the basic COCOMO model, and also is used in the subsequent models. The constant values a,b,c and d for the Basic Model for the different categories of system:

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

1. The effort is measured in Person-Months and as evident from the formula is dependent on Kilo-Lines of code. The development time is measured in months. These formulas are used as such in the Basic Model calculations, as not much consideration of different factors such as reliability, expertise is taken into account, henceforth the estimate is rough. Below is the JavaScript program for Basic COCOMO

```
// Javascript program to implement basic COCOMO

// Function to calculate parameters of Basic COCOMO
function calculate(table,n,mode,size)
{
    var effort,time,staff,model;

    // Check the mode according to size

    if (size >= 2 && size <= 50)
        model = 0; // organic

    else if (size > 50 && size <= 300)
        model = 1; // semi-detached

    else if (size > 300)
        model = 2; // embedded


    console.log("The mode is ",mode[model]);

    // Calculate Effort
    effort = table[model][0] * (size ** table[model][1]);

    // Calculate Time
    time = table[model][2] * (effort ** table[model][3]);

    // Calculate Persons Required
    staff = effort / time;

    console.log("Effort = ",effort," Person-Month");
    console.log("Development Time = ",time," Months");
    console.log("Average Staff Required = ",Math.round(staff)," Persons")
}


var table = [[2.4,1.05,2.5,0.38],[3.0,1.12,2.5,0.35],[3.6,1.20,2.5,0.32]]
var mode = ["Organic","Semi-Detached","Embedded"]
var size = 4;
calculate(table, 3, mode, size);

// This code is contributed by satwiksuman.
```

Output:

```
The mode is Organic

Effort = 10.289 Person-Month

Development Time = 6.06237 Months

Average Staff Required = 2 Persons
```

1. **Intermediate Model** – The basic Cocomo model assumes that the effort

is only a function of the number of lines of code and some constants evaluated according to the different software systems. However, in reality, no system's effort and schedule can be solely calculated on the basis of Lines of Code. For that, various other factors such as reliability, experience, Capability. These factors are known as Cost Drivers and the Intermediate Model utilizes 15 such drivers for cost estimation. Classification of Cost Drivers and their attributes: **(i) Product attributes –**

- Required software reliability extent
- Size of the application database
- The complexity of the product
- Run-time performance constraints
- Memory constraints
- The volatility of the virtual machine environment
- Required turnabout time
- Analyst capability
- Software engineering capability
- Applications experience
- Virtual machine experience
- Programming language experience
- Use of software tools
- Application of software engineering methods
- Required development schedule

2. **Detailed Model –** Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort. The Six phases of detailed COCOMO are:
    1. Planning and requirements
    2. System design
    3. Detailed design
    4. Module code and test
    5. Integration and test
    6. Cost Constructive model

Q.3 (a) What is critical path analysis? **03**

Critical path analysis is a project management technique used to identify the most important tasks in a project and the dependencies between them. It helps to identify the critical path of the project, which is the sequence of tasks that must be completed on time in order for the project to be completed on schedule.

To perform critical path analysis, you first need to create a list of all the tasks that need to be completed in the project and the dependencies between them. This can be done using a project management software tool or by creating a network diagram.

Once you have identified the tasks and dependencies, you can then calculate the critical path for the project. This involves determining the duration of each task and the earliest and latest start and end times for each task based on the dependencies. The tasks on the

critical path are those that have the least amount of slack, or flexibility, in their schedule. If any of these tasks are delayed, it will impact the overall project schedule.

(b) Illustrate the use of Gantt Chart for project scheduling with example. **04**

A Gantt chart is a graphical representation of a project schedule that shows the start and end dates of tasks, as well as any dependencies between tasks. It is named after Henry Gantt, who developed the concept in the early 20th century.

A Gantt chart is a useful tool for visualizing the progress of a project and identifying potential bottlenecks or areas that are at risk of delays. It can also be used to communicate the project plan to team members and stakeholders. By regularly updating the Gantt chart as tasks are completed, project managers can track the progress of the project and make any necessary adjustments to keep it on track.

Here is an example of a simple Gantt chart for a project to develop a new website:

Task | Start Date | End Date
Define project scope | 1/1/2022 | 1/5/2022
Design website layout | 1/6/2022 | 1/10/2022
Develop website content | 1/11/2022 | 1/20/2022
Develop website functionality | 1/21/2022 | 2/5/2022
Test and debug website | 2/6/2022 | 2/15/2022
Launch website | 2/16/2022 | 2/20/2022

In this example, the tasks are shown in the left column and the start and end dates are shown in the middle and right columns. The tasks are also color-coded to show their status

(c) Discuss various risk management approaches. **07**

There are several approaches that can be used for risk management in a software development project. These include:

1. Risk identification: This involves identifying and listing all the potential risks that could impact the project. This can be done through brainstorming sessions, reviewing project plans and documents, and conducting stakeholder interviews.
2. Risk assessment: Once risks have been identified, they need to be assessed in order to prioritize them. This involves analyzing the likelihood of the risk occurring and the potential impact on the project. Risks with a high likelihood and high impact should be given higher priority.
3. Risk response planning: After the risks have been prioritized, the next step is to develop a plan for how to respond to them. This may involve taking actions to prevent the risk from occurring, or developing contingency plans in case the risk does occur.
4. Risk monitoring and control: Once the risk response plan has been implemented, it is important to monitor the risks on an ongoing basis and make any necessary adjustments to the plan as the project progresses.
5. Risk review and evaluation: At the end of the project, it is important to review and evaluate the risk management process to identify any lessons learned and ways to improve future risk management efforts.

OR

Q.3 (a) Describe risk components and risk drivers. **03**

• The project manager identifies the risk drivers that affect the following risk components

- Performance risk - The degree of uncertainty that the product will meet its requirements and be fit for its intended use.
- Cost risk - The degree of uncertainty that the project budget will be maintained.
- Support risk— The degree of uncertainty that the resultant software will be easy to correct. adapt, and enhance.
- Schedule risk - The degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

• The impact of each risk driver on the risk component is divided into one of four impact

levels

- Negligible, marginal, critical, and catastrophic

• Risk drivers can be assessed as impossible, improbable, probable, and frequent

(b) Explain use of PERT in project management. **04**

PERT (Program Evaluation and Review Technique) is a project management tool used to plan, coordinate, and control complex projects. It is a network diagramming technique that shows the interdependencies between tasks and helps to identify the critical path of a project.

To use PERT, you first need to define the tasks that need to be completed in the project and the dependencies between them. This can be done using a project management software tool or by creating a network diagram.

PERT can be a useful tool for visualizing the interdependencies between tasks in a project and identifying the critical path. It can also be used to assess the impact of potential changes to the project plan and to identify potential bottlenecks or areas that are at risk of delays. By regularly updating the PERT chart as tasks are completed, project managers can track the progress of the project and make any necessary adjustments to keep it on track.

(c) Explain Product breakdown structure and Work Breakdown Structure with suitable examples. **07**

A product breakdown structure (PBS) is a hierarchical representation of the deliverables and work required to complete a project. It breaks down the project into smaller, more manageable pieces, starting with the overall project goal and working down to the individual tasks and deliverables.

A work breakdown structure (WBS) is similar to a PBS, but it focuses on the work required to complete the project rather than the deliverables. It is a hierarchical representation of the tasks and subtasks required to complete the project, starting with the overall project goal and working down to the individual activities.

Here is an example of a PBS for a project to develop a new website:

- Project goal: Develop a new website for a small business
    - Deliverable 1: Home page
    - Deliverable 2: About us page
    - Deliverable 3: Products page
    - Deliverable 4: Contact us page

And here is an example of a WBS for the same project:

- Project goal: Develop a new website for a small business
    - Task 1: Define project scope
    - Task 2: Design website layout
    - Task 3: Develop website content
        - Subtask 3.1: Write copy for home page
        - Subtask 3.2: Write copy for about us page
        - Subtask 3.3: Write copy for products page

Q.4 (a) Briefly discuss the necessity for software maintenance. **03**

Software maintenance is the process of making changes or updates to a software system after it has been deployed. It is an important aspect of software development because it helps to ensure that the system continues to function effectively and meet the needs of users over time.

There are several reasons why software maintenance is necessary:

- To fix bugs and defects
- To improve performance
- To add new features
- To maintain security
- To comply with regulatory requirements

Indetails

- To fix bugs and defects: As users interact with a software system, they may encounter problems or errors that need to be fixed. Maintenance can help to identify and resolve these issues.
- To improve performance: As a software system is used over time, it may become slower or less efficient. Maintenance can help to optimize the system and improve its performance.
- To add new features: As users' needs and expectations change, it may be necessary to add new features or functionality to a software system. Maintenance can help to incorporate these changes.
- To maintain security: As new security threats emerge, it may be necessary to update a software system to protect it from these threats. Maintenance can help to ensure that the system remains secure.
- To comply with regulatory requirements: In some cases, software systems may need to be updated to comply with new regulatory requirements or industry standards. Maintenance can help to ensure that the system meets these requirements.

(b) Briefly explain important SQA activities. **04**

Software quality assurance (SQA) is the process of verifying that a software system meets specified quality standards and requirements. It involves a set of activities that are designed to ensure the quality of the software throughout the development lifecycle. Some important SQA activities include:

- Reviewing requirements: SQA team members review the functional and non-functional requirements for the software to ensure that they are clear, complete, and testable.
- Developing a test plan: SQA team members create a detailed plan for testing the

software, including the types of tests that will be performed, the criteria for evaluating the results, and the resources needed to perform the tests.
- Creating test cases: SQA team members develop detailed test cases that outline the steps to be followed to test the software. These test cases may include manual test cases, as well as automated tests.
- Executing tests: SQA team members execute the tests defined in the test plan, either manually or using automated testing tools. They document the results of the tests and report any defects or issues that are found.
- Reviewing test results: SQA team members review the results of the tests to determine if the software meets the specified quality standards. They may also perform additional testing as needed to confirm the results.
- Reporting defects: If defects or issues are found during testing, SQA team members report them to the development team for resolution. They may also work with the development team to ensure that the defects are properly fixed.

Benefits of Software Quality Assurance (SQA):

1. SQA produces high quality software.
2. High quality application saves time and cost.
3. SQA is beneficial for better reliability.
4. SQA is beneficial in the condition of no maintenance for a long time.
5. High quality commercial software increase market share of company.
6. Improving the process of creating software.
7. Improves the quality of the software.

(c) Explain SEI CMM with all its level. **07**

The SEI Capability Maturity Model (CMM) is a framework for evaluating the maturity of an organization's software development processes. It is a way to assess the organization's capabilities and identify areas for improvement.

The CMM is divided into five levels, each representing a different level of maturity:

1. Initial (level 1): At this level, the organization's software development processes are ad hoc and informal. There is little planning or documentation, and there is a high risk of project failure.
2. Repeatable (level 2): At this level, the organization has established some basic processes for software development. There is more planning and documentation, and the organization is able to replicate successful projects.
3. Defined (level 3): At this level, the organization has a well-defined software development process that is documented and followed consistently. All projects follow the same process, and there is a focus on quality assurance.
4. Managed (level 4): At this level, the organization has a mature software development process that is closely monitored and controlled. The organization is able to accurately predict the results of software development projects and proactively manage any issues that arise.
5. Optimizing (level 5): At this level, the organization continuously improves its software development processes based on data and experience. The organization is able to adapt to changing requirements and environments, and there is a focus on innovation.

By evaluating its processes against the CMM levels, an organization can determine its current level of maturity and identify areas for improvement. It can then take steps to move to higher levels of maturity and improve the quality and predictability of its software development projects.

OR

Q.4 (a) What is Earned Value Analysis? **03**

Earned Value Analysis (EVA) is a project management technique that combines the actual cost of a project with the planned value of the work completed to determine the project's progress and performance. It is used to compare the planned cost and schedule of a project with the actual cost and schedule to determine if the project is on track and within budget.

Some key features of EVA include:

- It combines cost and schedule information: EVA combines the actual cost of the project with the planned value of the work completed, allowing project managers to track both cost and schedule performance.
- It provides a snapshot of project performance: By comparing the planned and actual cost and schedule, EVA provides a snapshot of the project's performance at a given point in time.
- It allows for proactive management: By identifying problems early on, EVA can help project managers to take corrective action to get the project back on track.
- It can be used to forecast future performance: By analyzing past performance, EVA can help project managers to forecast future performance and make necessary adjustments to the project plan.

The need for EVA arises because of the complexity and uncertainty involved in most software development projects. By providing a systematic way to track and assess the project's progress and performance, EVA can help project managers to identify problems early on and take corrective action to keep the project on track. It can also help to ensure that the project stays within budget and is completed on time.

(b) What is Software Re-Engineering? Discuss the Software Re-Engineering Process Model. **04**

Software re-engineering is the process of improving the design and structure of existing software systems. It involves making changes to the software to improve its maintainability, reliability, and performance, without changing its functionality.

The software re-engineering process model outlines the steps involved in the re-engineering process. These steps include:

1. Planning: The first step in the re-engineering process is to develop a plan for the re-engineering project. This may involve identifying the goals of the re-engineering effort, assessing the current state of the software system, and identifying the resources and constraints of the project.
2. Analysis: In this step, the software system is analyzed to identify areas for improvement and to develop a detailed understanding of its structure and functionality. This may involve reverse engineering the software to create detailed documentation or models of its design.
3. Design: Based on the results of the analysis, a new design for the software system is developed. This design should improve the maintainability, reliability, and performance of the software without changing its functionality.
4. Implementation: The new design is implemented by making the necessary changes to the software system. This may involve re-writing or re-organizing the code, or integrating new tools or technologies.
5. Testing: The re-engineered software is thoroughly tested to ensure that it meets the desired quality standards and requirements.
6. Deployment: Once the re-engineering process is complete, the software is deployed

to its intended users.

(c) Why are standards important for software? Briefly explain ISO standards for software organization. **07**

Standards are important for software for several reasons:

- Standards provide a common language and framework for software development, which helps to ensure that different software systems are compatible and can work together.
- Standards help to ensure that software is of high quality and meets certain minimum requirements for functionality and performance.
- Standards can help to reduce the risk of errors and defects in software, which can save time and money by reducing the need for debugging and maintenance.
- Standards can help to promote innovation and interoperability in the software industry by providing a clear set of guidelines for software development.

The ISO 9000 series of standards is based on the assumption that if a proper stage is followed for production, then good quality products are bound to follow automatically. The types of industries to which the various ISO standards apply are as follows.

1. ISO 9001: This standard applies to the organizations engaged in design, development, production, and servicing of goods. This is the standard that applies to most software development organizations.
2. ISO 9002: This standard applies to those organizations which do not design products but are only involved in the production. Examples of these category industries contain steel and car manufacturing industries that buy the product and plants designs from external sources and are engaged in only manufacturing those products. Therefore, ISO 9002 does not apply to software development organizations.
3. ISO 9003: This standard applies to organizations that are involved only in the installation and testing of the products. For example, Gas companies.

Q.5 (a) Describe various Software Configuration Items. **03**

Software configuration items (SCIs) are the elements of a software system that need to be managed and controlled throughout the software development life cycle. SCIs may include:

- Source code: This is the code that makes up the software system. It may include both the code that is written by developers and any third-party code that is used in the system.
- Documentation: This includes any documentation that is associated with the software, such as user manuals, technical guides, and design documents.
- Executable code: This is the compiled version of the source code that is ready to be run on a computer.
- Test cases: This includes the test cases that are used to test the software to ensure that it meets the required quality standards.
- Test data: This is the data that is used in the test cases to test the functionality of the software.
- Libraries and frameworks: This includes any third-party libraries or frameworks that are used in the software.
- Tools: This includes any tools or software applications that are used to develop, test, or maintain the software.

(b) Differentiate between planned project closure and unplanned project closure. **04**

Planned Project Closure	Unplanned Project Closure
The project is closed according to the project plan	The project is closed unexpectedly due to unforeseen circumstances
The project team is prepared for the project to end	The project team is not prepared for the project to end
The project is closed when all objectives have been achieved	The project is closed before all objectives have been achieved
The project is closed in an orderly manner	The project is closed in an abrupt or chaotic manner

(c) Define Baseline also discuss the Standard Change Process of Baseline. **07**

A baseline is a fixed point of reference against which the progress or performance of a project can be measured. In software development, a baseline may refer to a specific version of the software that has been formally reviewed and accepted by stakeholders. Once a baseline has been established, any changes to the software must go through a standard change process before they can be implemented.

The standard change process for a baseline typically includes the following steps:

1. Identify the change: The first step in the change process is to identify the change that is being proposed. This may involve identifying the problem or issue that the change is intended to address, as well as the proposed solution.
2. Assess the impact: The next step is to assess the impact of the change on the software and the project. This may involve analyzing the potential risks and benefits of the change, as well as the resources and time required to implement it.
3. Review and approval: The change proposal is then reviewed by the relevant stakeholders, including the project team, management, and any other affected parties. If the change is approved, it moves on to the next step in the process. If it is rejected, the change process ends and the software remains unchanged.
4. Implementation: If the change is approved, it is implemented in the software. This may involve modifying the source code, updating documentation, and performing testing to ensure that the change does not introduce any new problems or defects.
5. Review and acceptance: Once the change has been implemented, it is reviewed and accepted by the relevant stakeholders. If the change meets the required quality standards, it is formally accepted and becomes part of the baseline. If it does not meet the standards, it may need to be revised or rejected.

OR

Q.5 **(a)** Discuss reasons for premature project closure. **03**

A project may be closed prematurely for a variety of reasons, including:

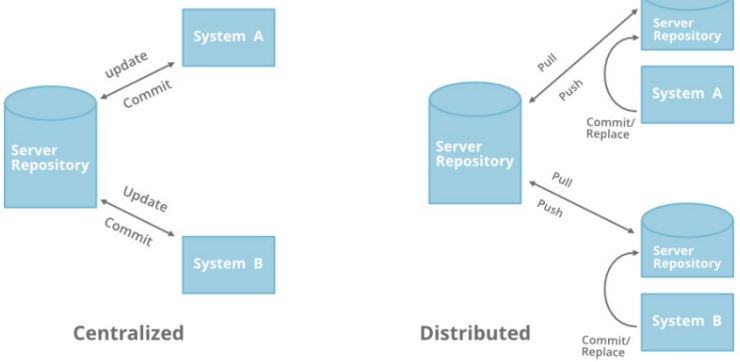
- Lack of resources: If a project lacks the necessary resources (e.g. budget, personnel, equipment) to continue, it may be closed prematurely.
- Change in priorities: If the priorities of the organization change, a project that was previously deemed important may no longer be a priority and may be closed prematurely.

- Poor performance: If a project is not meeting its goals or performance targets, it may be closed prematurely in order to redirect resources to more successful projects.
- Insufficient progress: If a project is not making sufficient progress towards its objectives, it may be closed prematurely.
- Political or external factors: Projects may also be closed prematurely due to external factors, such as changes in government policies, changes in funding sources, or changes in market conditions.

(b) Categorize version control models with diagrams. **04**

There are several version control models that are used to manage changes to software over time. Some common version control models include:

1. Centralized version control model: In this model, a central repository is used to store all versions of the software. Developers check out copies of the software from the repository and make changes, and then check the changes back in to the repository. This model is illustrated in the following diagram:



2. Distributed version control model: In this model, each developer has a local repository that contains a complete copy of the software and its history. Developers can make changes to their local repositories and then push their changes to a central repository or other developers' repositories.
3. Layered version control model: In this model, multiple layers of version control are used to manage the development of software. The innermost layer is used to manage changes to the software within a development team, while the outer layers are used to manage the integration of changes from multiple teams.
4. Local Version Control model: It is one of the simplest forms and has a database that kept all the changes to files under revision control. RCS is one of the most common VCS tools. It keeps patch sets (differences between files) in a special format on disk. By adding up all the patches it can then re-create what any file looked like at any point in time.

(c) Discuss project closure analysis report in detail with sample example. **07**

A project closure analysis report is a document that summarizes the results of a project and assesses its overall performance. It is typically prepared at the end of a project and is used to identify any lessons learned and best practices that can be applied to future projects.

The contents of a project closure analysis report may include:

- Executive summary: This is a brief overview of the report, outlining the key findings and recommendations.
- Project overview: This section provides a brief overview of the project, including its objectives, scope, and deliverables.
- Performance analysis: This section analyzes the performance of the project in terms of cost, schedule, and quality. It may include metrics such as budget and schedule

variance, as well as performance indicators such as customer satisfaction and defect rates.
- Lessons learned: This section identifies any key lessons learned during the project and recommends best practices for future projects.
- Recommendations: This section provides recommendations for improving the project process and addressing any issues that were identified during the project.
- Conclusion: This is a summary of the key findings and recommendations of the report.

Here is an example of a project closure analysis report:

EXECUTIVE SUMMARY

The XYZ project was completed on schedule and within budget, and the deliverables met the required quality standards. However, the project encountered several challenges that could be addressed in future projects.

PROJECT OVERVIEW

The XYZ project was a software development project with the following objectives:

- Develop a new software application for managing customer orders
- Integrate the new application with the company's existing systems
- Test and deploy the new application

The project was completed on schedule and within budget, and the final deliverables met the required quality standards.

PERFORMANCE ANALYSIS

Overall, the project performed well in terms of cost and schedule. The budget was $500,000 and the final cost was $495,000, resulting in a budget variance of -1%. The project was completed on schedule, with no significant delays.

In terms of quality, the project met the required standards. The final software application was tested and demonstrated to meet the functional and performance requirements. Customer satisfaction with the new application was also high, with a satisfaction rating of 95%.

LESSONS LEARNED

During the project, the following lessons were learned:

- It is important to clearly define the scope and objectives of the project at the outset to avoid scope creep.
- The project team should be closely monitored to ensure that they are meeting their commitments and making progress.
- Regular communication with stakeholders is critical to

ensure that their needs and concerns are addressed.

RECOMMENDATIONS

Based on the lessons learned during the project, the following recommendations are made:

- Establish a clear scope management process to ensure that the scope of the project is defined and controlled.
- Implement a project management tool to track the progress and performance of the project team.
- Increase communication with stakeholders to ensure that their needs and concerns are addressed in a timely manner.

CONCLUSION

Overall, the XYZ project was a success. The project was completed on schedule and within budget, and the deliverables met the required quality standards. However, there were several challenges that could be addressed in future projects. By implementing the recommendations outlined in this report, future projects can be more successful and efficient.

GUJARAT TECHNOLOGICAL UNIVERSITY
BE - SEMESTER–VII (NEW) EXAMINATION – SUMMER 2022

MARKS

Q.1 (a) What are the characteristic that make software project different from other project? **03**

Software projects: The projects that are used to engineer or develop a new software is referred to as software projects.

The software projects are different from other type of projects in many ways.

Difference:

- Software projects require logic and logical works.
- The complexity of a software project is unknown at the beginning.
- The progress is not visible unless one of the project module is completed at least.
- Software projects are more flexible than other projects.
- The resources required and the cost is also low when compared to the other types of project.

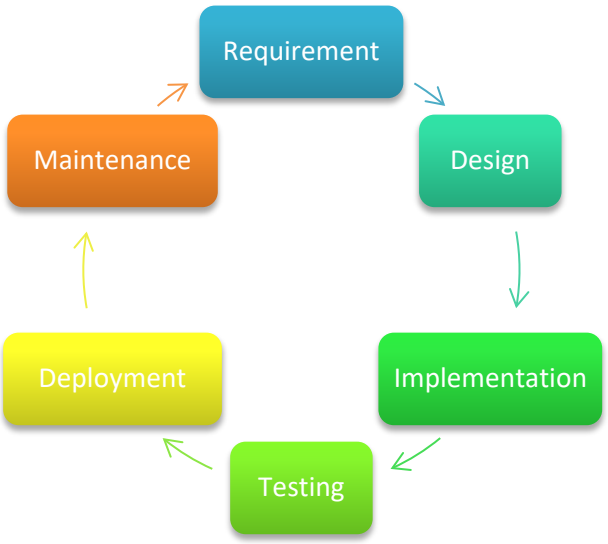
(b) Explain the major activities carried out by Software Project Management? **04**

Activities
Software Project Management consists of many activities, that includes planning of the project, deciding the scope of product, estimation of cost in different terms, scheduling of tasks, etc.
The list of activities are as follows:

1. Project planning and Tracking
2. Project Resource Management
3. Scope Management
4. Estimation Management
5. Project Risk Management
6. Scheduling Management
7. Project Communication Management
8. Configuration Management

(c) Describe major phases of Software Development life cycle. **07**

Software Development Life Cycle (SDLC) is a framework that defines activities that are performed during the software development process. There are 6 phases in SDLC model as given below.



1. Requirement: In this phase, all the requirements are collected from the customer/client. They are provided in a document called Businessmen requirement specification (BRS) and System requirement specification (SRS). All the details are discussed with the customer/client in detail.

2. Design: It has two steps:

- **High-level design (HLD):** It gives the architecture of software products.
- **Low-level design (LLD):** It describes how each and every feature in the product should work and every component.

3. Implementation:

- This is the longest phase.
- This phase consists of Front end + Middleware + Back-end.
- **In front-end:** Development of coding is done even SEO settings are done.
- **In Middleware:** They connect both the front end and back end.
- **In the back-end:** A database is created.

4. Testing: Testing is carried out to verify the entire system. The aim of the tester is to find out the gaps and defects within the system and also to check whether the system is running according to the requirement of the customer/client.

5. Deployment: After successful testing, the product is delivered/deployed to the client, and even clients are trained on how to use the product.

6. Maintenance: Once the product has been delivered to the client a task of maintenance starts as when the client will come up with an error the issue should be fixed from time to time.

Q.2 (a) Which factor decides the success of a project? **03**

1. Clear goals and objectives
2. Good planning
3. Effective communication

4. Strong project management
5. Quality work
6. Collaboration and teamwork
7. Adequate resources
8. Flexibility and adaptability
9. Risk management
10. Stakeholder engagement and buy-in
11. User experience
12. Time management
13. Cost management
14. Meeting deliverables and deadlines
15. Maintaining momentum and momentum

(b) Explain major activities carry out by a software manager. **04**

Responsibilities of Project Manager

Managing people
• Act as project leader
• Liaison with stakeholders
• Managing human resources
• reporting hierarchy etc.

Managing Project

• Defining and setting up project scope
• Managing project management activities
• Monitoring progress and performance
• Risk analysis at every phase
• Take necessary step to avoid or come out of problems
• Act as project spokesperson.

(c) Discuss the role of cost estimation in software development project. Briefly explain COCOMO model for cost estimation for all categories of project. **07**

Cost estimation is an important aspect of software development project management. It involves predicting the resources and costs that will be required to complete the project, and is used to help plan and budget for the project. There are a number of different techniques that can be used for cost estimation, including:

1. Expert judgment: This involves using the expertise and experience of project team members or external experts to estimate the costs and resources required for the project.
2. Analogous estimation: This involves using the costs and resources of similar projects as a basis for estimating the costs of the current project.
3. Parametric estimation: This involves using statistical models and algorithms to estimate the costs of a project based on certain variables, such as the size of the project or the complexity of the work.

One commonly used parametric estimation model is the COCOMO (Constructive Cost Model) model, which was developed by Dr. Barry Boehm in the 1980s. The COCOMO model is used to estimate the cost, schedule, and effort required to develop software. It is based on the premise that the cost and effort required for a software project is proportional to the size of the project and the complexity of the work.

The COCOMO model is divided into three categories: Basic COCOMO, Intermediate

COCOMO, and Detailed COCOMO. Each category provides a different level of detail and accuracy for cost estimation.

Basic COCOMO: This is the simplest and most basic version of the COCOMO model. It estimates the cost of a project based on the size of the project, measured in lines of code.

Intermediate COCOMO: This version of the COCOMO model includes additional factors that can affect the cost and effort of a project, such as the complexity of the project, the skills and experience of the project team, and the development environment.

Detailed COCOMO: This is the most detailed and accurate version of the COCOMO model. It includes all of the factors included in the intermediate model, as well as additional factors such as the reuse of existing code, the use of modern programming languages and tools, and the level of support required from external organizations.

Overall, the COCOMO model is a useful tool for estimating the costs and resources required for software development projects, and can help project managers to plan and budget effectively. However, it is important to remember that cost estimation is always subject to some level of uncertainty, and the actual costs of a project may differ from the estimates.

OR

(c) Discuss Capability Maturity Model (CMM) in detail. **07**

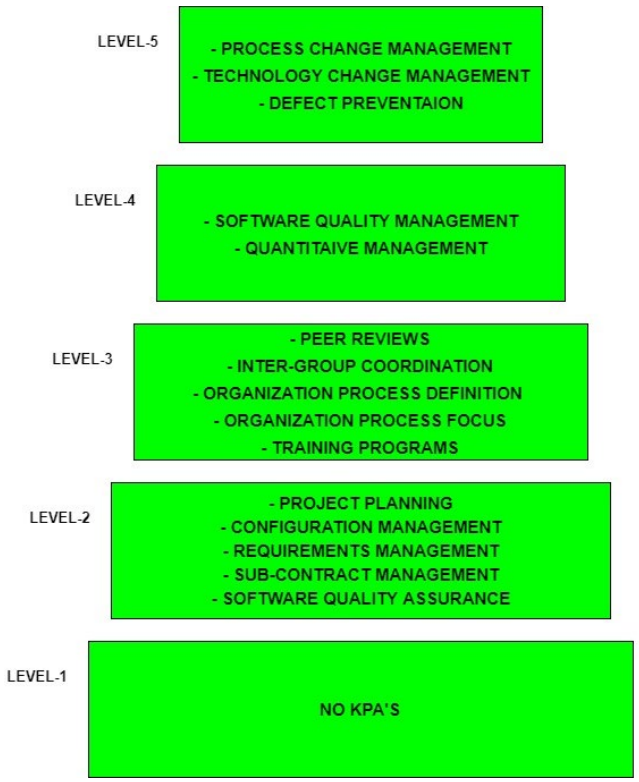
The Capability Maturity Model (CMM) is a framework that was developed in the 1980s by the Software Engineering Institute (SEI) to evaluate and improve the processes and practices used in software development organizations. The CMM is based on the premise that organizations can improve their software development capabilities by progressing through a series of levels, from an initial, ad hoc and reactive level, to a level of advanced process maturity that is characterized by continuous process improvement and the ability to produce high-quality software consistently.

The CMM consists of five levels:

1. Level 1: Initial (or "ad hoc") At this level, software development processes are ad hoc and reactive, and there is little or no formal planning or documentation.
2. Level 2: Repeatable At this level, processes are documented and are repeatable, but they are not yet fully effective or efficient. There is some formal planning, but it is not yet fully integrated into the software development process.
3. Level 3: Defined At this level, processes are defined and standardized, and they are managed using a defined set of software engineering policies and procedures. There is a strong emphasis on continuous process improvement, and there is a defined process for identifying and addressing problems.
4. Level 4: Managed At this level, processes are measured and controlled, and there is a strong focus on quality. There is a defined process for measuring and analyzing the effectiveness of processes, and for identifying and implementing improvements.
5. Level 5: Optimizing At this level, processes are continually improved and optimized based on data and experience. There is a strong focus on innovation and process excellence, and there is a defined process for integrating new technologies and practices into the organization.

The CMM is widely used in the software industry as a tool for evaluating and improving software development processes. It provides a structured and systematic approach to process improvement, and can help organizations to identify and address problems, and to continually improve the quality and efficiency of their software development practices.

OR



1. Level 1: Initial (or "ad hoc")

- Ad hoc and reactive processes
- Little or no formal planning or documentation

2. Level 2: Repeatable

- Documented processes that are repeatable, but not yet fully effective or efficient
- Some formal planning, but not fully integrated into the software development process

3. Level 3: Defined

- Standardized and defined processes that are managed using a defined set of software engineering policies and procedures
- Strong emphasis on continuous process improvement
- Defined process for identifying and addressing problems

4. Level 4: Managed

- Measured and controlled processes with a strong focus on quality
- Defined process for measuring and analyzing the effectiveness of processes, and for identifying and implementing improvements

5. Level 5: Optimizing

- Continually improved and optimized processes based on data and experience
- Strong focus on innovation and process excellence
- Defined process for integrating new technologies and practices into the organization

Q.3 (a) Difference between ISO9000 and SEI-CMM. **03**

	ISO 9000	SEI-CMM
Definition	ISO 9000 is a family of international standards that provide guidelines for quality management systems.	The SEI-CMM is a framework for evaluating and improving the processes and practices used in software development organizations.
Focus	ISO 9000 is focused on quality	The SEI-CMM is specifically

	management in general, and can be applied to any type of organization.	focused on software development processes and practices.
Evaluation	ISO 9000 uses a process-based approach to evaluating an organization's quality management system. It requires an external audit by a third party.	The SEI-CMM uses a capability-based approach to evaluating an organization's software development processes and practices. It does not require an external audit, but rather relies on self-assessment.
Levels	ISO 9000 has three levels: basic, intermediate, and advanced.	The SEI-CMM has five levels: initial, repeatable, defined, managed, and optimizing.
Certification	Organizations can seek certification to ISO 9000 if they meet the standards set out in the ISO 9000 series of guidelines.	The SEI-CMM is not a certification process, but rather a framework for evaluating and improving processes. However, organizations can seek recognition or validation for achieving a certain level of maturity.
Continuous improvement	ISO 9000 emphasizes continuous improvement as a key principle.	The SEI-CMM also emphasizes continuous improvement as a key principle, and includes a focus on process excellence and innovation at the highest level.

Overall, the ISO 9000 and SEI-CMM are both focused on improving processes and practices, but they differ in their specific focus and approach. ISO 9000 is a general quality management standard that can be applied to any type of organization, while the SEI-CMM is specifically focused on software development processes and practices. ISO 9000 uses a process-based approach and requires external audit, while the SEI-CMM uses a capability-based approach and relies on self-assessment. The ISO 9000 has three levels, while the SEI-CMM has five levels. ISO 9000 offers certification to

(b) Explain basic steps of Function Point Analysis. **04**

Function Point Analysis (FPA) is a method for measuring the size and complexity of software systems. It is based on the idea that the functionality provided by a software system can be measured in terms of the number and type of "functions" that it performs.

The basic steps of Function Point Analysis are as follows:

1. Identify the functions performed by the software system: The first step in FPA is to identify the functions that the software system performs. These functions may include input, output, inquiries, and data manipulation.
2. Determine the number and type of function points: Once the functions have been identified, the next step is to determine the number and type of function points. This is done by assigning a weight to each function based on its complexity and the amount of resources required to implement it.
3. Calculate the unadjusted function point count: The unadjusted function point count is calculated by summing the weights of all the functions in the software system.
4. Adjust the function point count for the complexity of the software system: The unadjusted function point count is then adjusted based on the complexity of the software system. This may involve adding or subtracting points based on factors such as the number of external interfaces, the level of data complexity, and the level of security.
5. Determine the function point size of the software system: The final step in FPA is to determine the function point size of the software system by applying a valuation

formula to the adjusted function point count. This formula takes into account the relative difficulty of implementing different types of functions.

(c) Explain PERT & CPM techniques in brief **07**

1. Project Evaluation and Review Technique (PERT) **:**
PERT is appropriate technique which is used for the projects where the time required or needed to complete different activities are not known. PERT is majorly applied for scheduling, organization and integration of different tasks within a project. It provides the blueprint of project and is efficient technique for project evaluation .

PERT (Program Evaluation Review Technique) is a method that is used to plan and control complex projects that involve a high level of uncertainty. It is based on the idea of breaking a project down into smaller tasks and defining the dependencies between those tasks, and then using that information to create a network diagram that shows the order in which the tasks must be completed. PERT allows project managers to visualize the tasks that need to be completed and the dependencies between those tasks, and to identify the critical path and allocate resources accordingly.

One of the key features of PERT is that it allows for the incorporation of uncertainty in the form of "optimistic," "most likely," and "pessimistic" time estimates for each task. These estimates are used to calculate a "weighted average" time estimate for each task, which is then used to determine the overall duration of the project. This allows project managers to account for the uncertainty and variability that is inherent in complex projects, and to plan accordingly.

2. Critical Path Method (CPM) **:**
   CPM is a technique which is used for the projects where the time needed for completion of project is already known. It is majorly used for determining the approximate time within which a project can be completed. Critical path is the largest path in project management which always provide minimum time taken for completion of project

CPM (Critical Path Method) is a similar method that is used to plan and schedule projects. It involves creating a network diagram that shows the tasks that need to be completed and the dependencies between those tasks, and then estimating the time required to complete each task. The critical path is the longest path through the network diagram, and it represents the minimum amount of time that is required to complete the project. CPM allows project managers to identify the critical tasks in a project and to allocate resources appropriately

OR

Q.3 (a) How Activity Planning done? Explain How to Identifying Critical Activities? **03**

Activity planning is the process of identifying and organizing the tasks that need to be completed in order to achieve the goals of a project. It involves breaking a project down into smaller, more manageable tasks, and defining the dependencies between those tasks.

There are a number of steps involved in activity planning, including:

1. Define the project scope
2. Break the project down into smaller tasks
3. Define the dependencies between tasks

4. Estimate the time and resources required for each task
5. Create a project schedule

Identifying critical activities is an important part of project management, as it allows project managers to focus their efforts and resources on the tasks that are most critical to the success of the project. There are a number of steps that project managers can take to identify critical activities, including:

1. Identify tasks with significant impact on project schedule
2. Identify tasks dependent on other tasks
3. Analyze impact of delays on project schedule
4. Monitor progress and adjust project schedule as needed

(b) What is Different between methods and Methodologies? **04**

Method	**Methodology**
A specific approach or technique used to complete a task	A set of related methods and practices that are used to guide the development and management of a project
Focuses on how to do something	Focuses on why and when to do something
Specific to a task or activity	Broad and covers multiple activities and tasks
Can be used as part of a methodology	Cannot be used as a standalone approach

(c) What is Process Model? Explain how to choose process model. **07**

A process model is a representation of the steps, activities, or tasks that are involved in a business process, project, or workflow. It is a tool that helps to understand, analyze, and optimize the process by visualizing it in a structured way.

There are several types of process models that can be used, depending on the specific needs and goals of the organization. Some common process models include:

1. Linear or sequential process model: This model involves a series of steps that are followed in a specific order. It is often used for simple processes that have a clear start and end point.
2. Iterative process model: This model involves repeating a series of steps until a desired result is achieved. It is often used for complex processes that require multiple iterations to optimize and improve the outcome.
3. Adaptive process model: This model involves adjusting the process as needed based on changing circumstances or requirements. It is often used in dynamic environments where the process needs to be flexible and responsive to change.
4. Agile process model: This model is based on the Agile software development method and emphasizes flexibility, collaboration, and rapid iteration. It is often used in projects that require rapid delivery and frequent changes.

To choose the appropriate process model for a given situation, consider the following factors:

1. The complexity and scope of the process: For simple processes, a linear or sequential model may be sufficient. For more complex processes, an iterative or adaptive model may be more appropriate.
2. The needs and goals of the organization: Consider the specific needs and goals of the organization, as well as any constraints or limitations that may impact the process.

3. The available resources: Consider the resources that are available, including time, budget, and personnel.
4. The level of uncertainty and change: For processes that are subject to frequent change or uncertainty, an adaptive or agile model may be more appropriate.

Ultimately, the right process model will depend on the specific needs and goals of the organization and the nature of the process itself.

Q.4 (a) What is SRS? What is the use of SRS? **03**

SRS stands for Software Requirements Specification. It is a detailed document that outlines the requirements for a software system, including the functional and non-functional requirements, as well as any constraints or limitations.

The purpose of an SRS is to provide a clear and concise description of the software system that is being developed, so that all stakeholders (including the development team, the customers, and the users) have a common understanding of the system's capabilities and limitations. It serves as a blueprint for the development process, helping to ensure that the final product meets the needs and expectations of all parties involved.

Here is a list of some common uses for an SRS (Software Requirements Specification) document:

1. Providing a common understanding of the software system
2. Guiding the development process
3. Defining the scope of the project
4. Communicating requirements to the development team
5. Facilitating testing and validation
6. Providing a reference for future updates or modifications
7. Assisting with project planning and estimation
8. Facilitating stakeholder communication and collaboration
9. Assisting with quality assurance and risk management.

(b) Briefly Discuss Work Breakdown Structure. **04**

A Work Breakdown Structure (WBS) is a hierarchical decomposition of a project's scope into smaller, more manageable components. It is a visual representation of the project that helps to organize and structure the work, and is typically used to identify the deliverables, milestones, and tasks required to complete the project.

The WBS is usually organized into three levels:

1. Level 1: The top level of the WBS represents the major deliverables or outcomes of the project.
2. Level 2: The second level of the WBS represents the major components or sub-deliverables that are required to complete the top-level deliverables.
3. Level 3: The third level of the WBS represents the individual tasks or activities that are required to complete the sub-deliverables.

The WBS helps to provide a clear understanding of the project's scope and helps to identify the resources and dependencies required to complete the work.

The WBS helps to break down a project into smaller, more manageable components, making it easier to understand and manage the work. It also helps to identify the resources and dependencies required to complete the work, which is important for project planning

and estimation.

(c) What is Risk? List Categories of Risk and Explain How to Evaluating Risk **07**

Risk is the potential for an event or situation to cause harm or loss. In the context of project management, risk refers to the potential for something to go wrong or to have a negative impact on the project.

There are several categories of risk that can affect a project, including:

1. Technical risk: This type of risk refers to the potential for problems or failures to occur due to technical issues, such as inadequate technology, software bugs, or hardware failures.
2. Operational risk: This type of risk refers to the potential for problems to occur due to inadequate processes, procedures, or controls.
3. Financial risk: This type of risk refers to the potential for financial loss due to factors such as cost overruns, budget constraints, or changes in market conditions.
4. Legal risk: This type of risk refers to the potential for legal issues or liabilities to arise due to factors such as contracts, regulations, or compliance requirements.
5. Reputational risk: This type of risk refers to the potential for damage to an organization's reputation due to negative events or circumstances.

To evaluate risk, it is important to identify and assess the likelihood and impact of potential risks. This can be done using a variety of techniques, such as:

1. Risk identification: This involves identifying all of the potential risks that could affect the project. This can be done through brainstorming sessions, stakeholder interviews, or review of past projects.
2. Risk assessment: This involves evaluating the likelihood and impact of each identified risk. The likelihood of a risk occurring can be estimated using probability scales, while the impact of a risk can be measured in terms of its potential consequences (e.g., cost, time, quality).
3. Risk prioritization: This involves ranking the identified risks based on their likelihood and impact. Risks with a high likelihood and high impact should be given the highest priority.
4. Risk management: This involves implementing strategies to mitigate, transfer, or accept the identified risks. This may include developing contingency plans, purchasing insurance, or allocating additional resources to address the risk.

By identifying and evaluating risks, organizations can better understand and prepare for potential challenges, and can take proactive steps to minimize their impact on the project.

OR

Q.4 (a) What are the features of EVA? **03**

Earned Value Analysis (EVA) is a project management technique that is used to measure the progress and performance of a project. It involves comparing the planned value (PV) of the work that has been completed to the actual cost (AC) of the work, and is calculated as the ratio of the planned value to the actual cost (PV/AC).

Here are some key features of EVA:

1. Measuring progress
2. Determining cost and schedule variance

3. Forecasting future performance
4. Identifying areas for improvement
5. Integration with other project management tools

(b) Discuss Redevelopment vs. Reengineering. **04**

Feature	Redevelopment	Reengineering
Scope	Focused on making incremental improvements to an existing product or process	Involves redesigning or restructuring an existing product or process
Goals	Enhancing the performance or functionality of the existing system	Improving efficiency, effectiveness, or competitiveness
Approach	Making incremental changes to the existing system	Redesigning or restructuring the system from the ground up
Level of change	Involves less change than reengineering	Involves more significant changes, such as reorganizing workflows, automating processes, or outsourcing work
Examples of activities	Updating software, improving product design, optimizing processes	Redesigning business processes, automating workflows, outsourcing non-core activities
Examples of tools	Lean Six Sigma, agile methodologies, continuous improvement approaches	Business process management, lean principles, total quality management

(c) Define Software Quality? Explain How Techniques Help Enhance Software Quality? **07**

Software quality refers to the degree to which a software product meets the needs of its users and the expectations of its stakeholders. It is a measure of the reliability, usability, performance, and maintainability of the software.

Here are some benefits of using techniques to enhance software quality:

1. Improved reliability: By using techniques such as testing and code review, organizations can identify and fix defects in the software, which can improve its reliability and reduce the risk of failures.
2. Enhanced usability: By clearly defining the requirements for a software product and testing it with users, organizations can improve its usability and ensure that it meets the needs of its users.
3. Improved performance: By using techniques such as continuous integration and delivery, organizations can ensure that code changes are integrated and tested frequently, which can improve the overall performance of the software.
4. Increased maintainability: By using agile methodologies and focusing on continuous improvement, organizations can enhance the maintainability of their software by making it easier to update and modify over time.
5. Increased customer satisfaction: By using techniques to enhance software quality, organizations can improve the overall user experience and increase customer satisfaction.

Q.5 (a) What is Software Re-Engineering? List out Various Activites of Software Re-Engineering. **03**

Software reengineering, also known as software reverse engineering, is the process of analyzing, modifying, and improving the design and structure of existing software. It is often used to improve the efficiency, maintainability, or compatibility of the software, or to add new features or capabilities.

Here is a list of activities involved in software reengineering:

1. Code analysis
2. Reverse engineering
3. Design improvement
4. Code transformation
5. Integration and testing

(b) Briefly explain Elements in Closure Analysis Report. **04**

A closure analysis report is a document that is used to evaluate the performance of a project during the closure phase. It is designed to identify any lessons learned or areas for improvement, and to document the final status of the project.

Here are some common elements that may be included in a closure analysis report:

1. Introduction: This section provides an overview of the purpose of the report and the project it covers.
2. Project summary: This section provides a summary of the project, including its goals, objectives, and deliverables.
3. Key accomplishments: This section highlights the key accomplishments of the project, including any milestones that were achieved or challenges that were overcome.
4. Lessons learned: This section identifies any lessons learned during the project, such as areas where the project was successful or where improvements could be made in the future.
5. Recommendations for future projects: This section provides recommendations for future projects, based on the lessons learned during the current project.
6. Project closure summary: This section provides a summary of the project closure process, including the final status of the project and any outstanding issues that need to be addressed.
7. Appendices: This section may include any additional supporting materials, such as project documents or presentations.

By including these elements in the closure analysis report, organizations can use the report to evaluate the performance of the project and to identify opportunities for improvement. This can help to increase the chances of success for future projects and improve overall project outcomes.

(c) Discuss step wise project planning with an example. **07**

Project planning is the process of defining the scope, goals, and resources needed to complete a project. It involves establishing a timeline, identifying the tasks that need to be completed, and allocating resources to those tasks.

Here are the steps involved in project planning, with an example:

1. Define the project scope: This involves identifying the specific goals and objectives of the project, as well as the boundaries of the work that will be done. For example, if the project is to design a new website for a company, the scope might include creating a user-friendly interface, integrating with social media, and optimizing for search engines.
2. Create a project plan: This involves establishing a timeline for the project and breaking down the work into smaller, manageable tasks. For example, the project plan for the website design project might include tasks such as researching design trends, creating wireframes, and testing the final design.
3. Allocate resources: This involves identifying the resources (such as personnel, equipment, and materials) that will be needed to complete the project, and assigning those resources to the tasks in the project plan. For example, the website design project might require a team of designers, developers, and content writers, as well as access to design software and hosting services.
4. Monitor progress: This involves regularly tracking the progress of the project to ensure that it is on track and that any issues or challenges are addressed in a timely manner. For example, the project manager for the website design project might use project management software to track the progress of tasks and identify any issues that need to be addressed.
5. Review and adjust the plan: As the project progresses, it may be necessary to review and adjust the project plan. This could involve adding new tasks, revising the timeline, or reallocating resources as needed. For example, if the website design project encounters unexpected challenges, the project manager might need to adjust the timeline or add additional resources to ensure that the project stays on track.

OR

Q.5 (a) Explain Advantage and disadvantage of Software Re-Engineering. **03**

Advantages of software re-engineering:

1. Improved maintainability
2. Enhanced functionality
3. Better performance
4. Increased reliability

Disadvantages of software re-engineering:

1. High cost
2. Risk of failure
3. Time-consuming
4. Loss of existing functionality

(b) Explain Stress and its significance in IT projects. **04**

Stress is a common occurrence in IT projects and can have significant impacts on the project's success and the well-being of the project team. Stress is a normal response to challenging or demanding situations and can be triggered by a variety of factors, such as tight deadlines, heavy workloads, conflicting priorities, or lack of resources.

In IT projects, stress can be caused by the complexity of the technology being used, the pressure to meet project milestones and deadlines, or the need to work long hours to complete tasks. It can also be caused by personal factors, such as work-life balance, interpersonal conflicts, or lack of job satisfaction.

Stress can have significant impacts on the project team, including reduced productivity, increased absenteeism, and higher turnover rates. It can also lead to burnout, which is a state of

physical, emotional, and mental exhaustion caused by prolonged stress.

Managing stress is important for the success of IT projects, as it can help to ensure that project team members are able to work effectively and deliver high-quality results. Some strategies for managing stress in IT projects include:

1. Establishing clear communication channels and expectations
2. Providing adequate resources and support
3. Encouraging a healthy work-life balance
4. Managing workloads and priorities effectively
5. Providing opportunities for team building and team support

(c) Write short note on Software Configuration Management. **07**

Software configuration management (SCM) is the process of controlling, coordinating, and tracking changes to a software system or application throughout its development and maintenance. It involves the use of tools and processes to ensure that the software is developed and maintained in a consistent and controlled manner.

SCM includes activities such as version control, building and releasing software, and managing dependencies between different versions of the software. It also involves the tracking of changes to the software, such as bug fixes, new features, and other modifications.

SCM is an important aspect of software development as it helps to ensure the integrity and reliability of the software, and makes it easier to maintain and update over time. It also helps to reduce the risk of errors and defects in the software by providing a systematic way to manage and track changes to the codebase.

- SCM involves the use of a configuration management tool, which is a software application that helps to automate and streamline the SCM process.
- SCM typically includes a version control system, which is a tool that tracks and manages changes to the software codebase.
- SCM is typically used in large and complex software development projects, where it is important to track and control changes to the software to ensure that it is developed and maintained in a consistent and reliable manner.
- SCM can help to improve the efficiency and productivity of the software development process by providing a structured and controlled approach to managing changes to the software.
- SCM is an important aspect of software quality assurance, as it helps to ensure that the software is tested and released in a consistent and reliable manner.

differentiate between planned project closure and unplanned project closure

Feature	Planned Project Closure	Unplanned Project Closure
Purpose	To complete the project as planned	To terminate the project early
Timing	Scheduled in advance	Unforeseen circumstances
Impact on budget	Minimal or none	Negative
Impact on stakeholders	Minimal or none	Negative

Documentation	Detailed and complete	Limited or incomplete

****