Name Kalan Joshi
Class: IT-B
RollNo. 057

ASSIGNMENT-2

Ques.1 What are the different activities during Software Project Planning?

Answer. Software project planning is the process of defining the scope, objectives, and approach for a software development project. It involves the creation of a detailed plan that outlines the project's tasks, timelines, resource allocation, and budget. Effective project planning is crucial for delivering software projects on time, within budget, and meeting stakeholder expectations.

Here are the key aspects and steps involved in software project planning:-

1. Project Initiation: Define the project's purpose, objectives, and scope. Identify the key stakeholders, projects team members, and their roles. Determine the feasibility of the project and its alignment with organizational goals.
2. Requirements Analysis: Gather and analyze the project requirements, including functional and non-functional requirements. Ensure a clear understanding of what the software needs.
3. Project Scope Definition: Clearly define the scope of the project, specifying what is included and excluded.
4. Task Estimation: Estimate the effort, time, and resources required for each task or activity.
5. Resource Allocation: Identify the required resources, including personnel, hardware, software, and tools. Allocate resources based on task requirements and availability.
6. Project Scheduling: Develop a project schedule that includes task sequences, dependencies, and durations.
7. Risk Assessment and Management: Identify potential risks that may impact the project's success. Develop a risk management plan to mitigate and manage these risks.

8. Quality Planning : Define the quality standards and processes that will be followed throughout the project to ensure the software meets quality requirements.

9. Budget Planning : Develop a budget that includes cost estimates for resources, tools, and other project-related expenses. Monitor and manage project expenditures.

10. Project Monitoring and Control : Develop methods and metrics for monitoring project progress. Use project management tools to track task completion, identify issues, and make necessary adjustments.

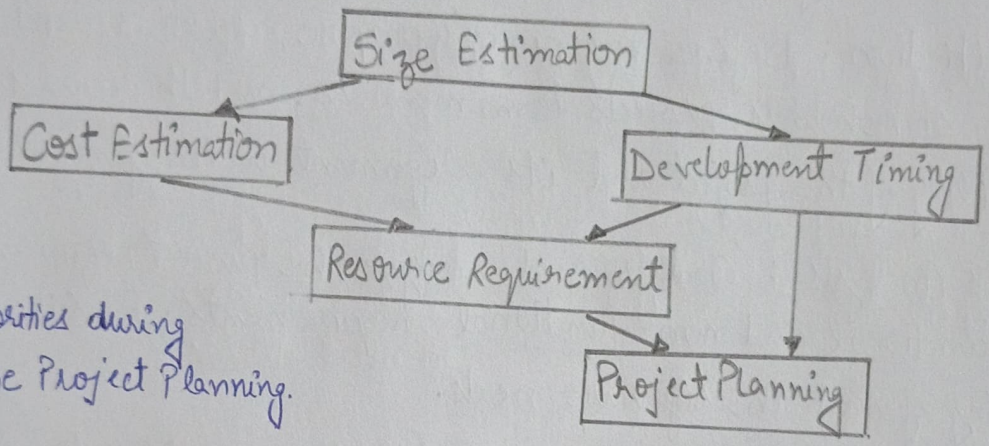

Figure. Authorities during Software Project Planning.

Ques-2 Differentiate between function oriented design and object oriented design.

Answer.

Function Oriented Design	Object Oriented Design
(i) The basic abstractions, which are given to the user, are real world functions.	(i) The basic abstractions are not the real world functions but are the data abstraction where the real world entities are represented.
(ii) Functions are grouped together by which a higher level function is obtained.	(ii) Function are grouped together on the basis of the data they operate since the classes are associated with their methods.
(iii) They are carried out using structured analysis and structured design i.e., data flow diagram	(iii) They are carried out using UML

(iv) In this approach the state information is often represented in a centralized shared memory.	(iv) In this approach the state information is not represented in a centralized memory but is implemented or distributed among the object of the system.
(v) It is a top-down approach	(v) It is a bottom up approach
(vi) Begins by considering the use case diagrams and the scenarios.	(vi) Begins by identifying objects and classes.
(vii) In function oriented design we decompose in function / procedure level.	(vii) We decompose in class level.
(viii) This approach is mainly used for computation sensitive application.	(viii) This approach is mainly used for evolving system which mimics a business or business case.

Ques.3 Explain risk management activities? Give top five risk in case of software development.

Answer. Risk management in software development involves identifying, assessing, mitigation, and monitoring potential risks that could negatively impact the success of a project. It's essential to ensure that project stays on track, within budget, and meets its objectives. Risk management activities typically include:

1. Risk Identification: Recognizing potential risks early in the project. These risks can be related to technology, budget, scheduling, or resources.
2. Risk Analysis: Assessing the identified risks to determine their likelihood and potential impact on the project.
3. Risk Prioritization: Ranking risks based on their severity and the probability of occurrence.
4. Risk Mitigation Planning: Developing strategies to reduce the impact of high priority risks. This may involve contigency planning, resource reallocation, or adopting safer technologies.
5. Risk Monitoring and Control: Continuously tracking risks throughout the project lifecycle and adjusting mitigation strategies as needed.

Name Karan Jotwani
Class IT-B
Roll No. 057

Top five risks in software development :-

1. Requirement changes / Scope Creep : Frequent changes in project requirements, leading to increased workload, extended timelines, and higher costs.

2. Unrealistic Timelines : Setting deadlines that are too aggressive, resulting in rushed development, poor quality, and burnout

3. Technical Debt : Accumulating shortcuts in code or infrastructure that may cause maintenance challenges later.

4. Inadequate Testing : Lack of thorough testing can result in the release of a product with critical bugs.

5. Team Skill Gaps: Insufficient technical expertise within the team to meet project demands, which can lead to delays or poor-quality deliverables.

Ques. 4 Define modularity. Explain
a) Stamp Coupling
b) Content Coupling
c) Functional Cohesion
d) Procedural Cohesion

Answer. Modularity in software development refers to the design principle of breaking down a system into smaller, independent, and manageable units called modules. Each module peforms a specific part of the system's functionality and can be developed, tested, and maintained independently. Modularity improves code readability, maintainability, and reusability. Well-designed modular systems allows changes to be made to one part of the system without impacting others.

a) Stamp Coupling : Stamp Coupling (also called data-structure coupling) occurs when modules share a data structure, and not just the elements of the data structure that they need. For example, if Module A

passes an entire record to Module B, but Module B only uses a part of that record, stamp coupling is present.

↳ Example → Passing a whole object to a function when only one attribute of that object is needed by the function.

↳ Disadvantages ⇒ This can lead to unnecessary dependencies and makes the code less flexible. If the shared data structure changes, all modules dependent on it may need to be updated.

b) Content Coupling : Content coupling occurs when one module directly modifies or relies on the internal workings (e.g., variables or control flow) of another module. This is the highest (most undesirable) level of coupling because it makes modules highly independent.

↳ Example ⇒ Module A directly accesses or alters the internal variables of Module B.

↳ Disadvantages ⇒ This makes the system difficult to maintain and modify because change in one module may lead to unpredictable behaviour in another.

c) Functional Cohesion : Functional cohesion occurs when all the elements of a module are grouped together because they all contribute to a single, well-defined task. This is the highest level of cohesion and is the most desirable.

↳ Example ⇒ A module that handles the calculation of payroll, with all elements focused on this specific task (e.g. gathering employee data, calculating taxes, and generating reports).

↳ Advantages ⇒ Modules with functional cohesion are easier to understand, test, and maintain since they focus on a single function or responsibility.

d) Procedural Cohesion : Procedural cohesion occurs when the elements of a module are grouped together because they follow a sequence of steps to achieve a task, even if the elements are unrelated to each other in functionality.

↳ Examples ⇒ A module that first validates user input & then calculates & displays them, even though these tasks are not functionally

Name Karan Joshi
Class IT-B
RollNo. 057

related but must occur sequentially.

↳ Disadvantages ⇒ Procedural cohesion is less desirable than functional cohesion because the tasks within the module might not be conceptually related, leading to reduced clarity and maintainability.

Ques. 5. a) Describe the trade-off between time versus cost in Putnam Resource Allocation model.

b) Differentiate between Cohesion and Coupling.

Answer a) The Putnam Resource Allocation Model, also known as the Putnam-Norden-Rayleigh Curve, is a software project management model that helps in estimating time and effort for a software project. It is based on the observation that the effort (manpower) and time needed to complete a project are related in a way that can be plotted on a Rayleigh curve.

The Putnam model demonstrates that there is a trade-off between the time required to complete a project and the cost (in terms of resources or efforts). The key trade-off is as follows:

1. Decreasing Time Increases Cost: When a project is required to be completed in less time, more resources (manpower) need to be added. This leads to an increase in overall cost because more people working in parallel can cause inefficiencies, such as increased communication overhead or difficulty in dividing the work.

2. Increasing Time Reduces Cost: On the other hand, if more time is allocated to a project, the same work can be done with fewer resources. The team can work at a steady pace, which often results in lower cost due to less overhead and more efficient use of resources.

The Putnam model suggests that there is an optimal point where the time and resource allocation are balanced for minimal cost. Beyond this point, compressing the project duration too much leads to exponentially increasing costs, while extending the project duration unnecessarily might reduce

Name Karan Joshi
Class IT-B
Roll No. 057

efficiency without significant cost savings.

b)

Cohesion

(i) Cohesion refers to how closely related and focused the responsibilities of a single module are.

(ii) High cohesion is desirable as it indicates that a module perform a single, well-defined task.

(iii) Concerns the internal relationship of elements within a module

(iv) High cohesion improves understability, reusability, and maintability of the module.

(v) Types of Cohesion :-
- Functional Cohesion (most desirable)
- Sequential Cohesion
- Communicational Cohesion
- Procedural Cohesion
- Temporal Cohesion
- Logical Cohesion
- Coincidental Cohesion (least desirable)

(vi) Example :- A module that handles all users authentication tasks (login, logout, and session management) has high cohesion.

Coupling

(i) Coupling refers to the degree of inter-dependence between different modules.

(ii) Low coupling is desirable as it reduces dependencies between modules, making the system more modular and maintainable.

(iii) Concerns the external relationship between different modules.

(iv) Low coupling makes the system more flexible and easier to maintain, as changes in one module are less likely to affect others.

(v) Types of Coupling :-
- Data Coupling (least harmful)
- Stamp Coupling
- Control Coupling
- External Coupling
- Common Coupling
- Content Coupling (most harmful)

(vi) Example :- If two modules share global variables or directly manipulate each other's internal data, they have high coupling.