Question Bank

SPM

1. What do you understand by software project planning? What are the various planning objectives? Also discuss various types of project plans with suitable example.

2. Write short notes on the following with suitable example:

    1. Software project estimation models

    2. Structure of a software project management plan.

3. Describe the following:

    1. Vision and scope document

    2. Management Spectrum

    3. SPM framework

    4. Decision Process

4. What you understand by work break structure (WBS)? What are the various types of WBS? What is the role of WBS directory and what the contents of it? Explain.

5. 1) What is the difference between the project life cycle and product life cycle? Discuss.

    2) Write a short note on organizational behaviour.

6. What do you mean by project scheduling? What are the scheduling objectives? How to build the project schedule?

7. What is the significance of project monitoring and control? What are the various dimensions of project monitoring and control? Does monitoring and control affect the project schedule? Discuss.

8. 1) what do you understand by schedule performances index (SPI)? Discuss.

    2) What do you mean by software review? What is the significance of software reviews in software project management?

9. Write short notes on the following with examples:

    1) Code Review

2) Error Tracking

10. 1) what is software testing? What are software testing objectives? Discuss various types of testing in detail.

2) Write a short note on testing automation and testing tools.

11. 1) what is the difference between testing principles and testing strategies? Discuss.

2) What is the difference between program verification and program validation? Explain the life cycle verification approach with suitable diagram.

12. 1) Write a short note on SEI capability maturity model (CMM).

2) Explain the term statistical quality assurance and clean room process.

13. What do you understand by software configuration management (SCM)? What are various software configuration items and tasks? Discuss with suitable example.

14. Describe the following with example:

1) Risk breakdown structure (RBS)

2) Cost benefits analysis.

15. What is the role of a software project management tool? Describe any software project management tool in detail.

16. Explain Software Project Planning, giving its various objectives. Also discuss the structure of a software project Management plan in detail.

17. 1) Explain why the intangibility of software system possess special problems for Software Project management?

2) Discuss the various responsibilities of a software project manager.

18. What do you mean by the software project estimation? Give various estimation models. Describe any one of the estimation model using suitable examples.

19. 1) What do you understand by the activity network and the Gantt chart? Draw the activity, network and Gantt chart representations for the following table that indicates the various tasks involved in completing a software project, the corresponding activities, and the estimated effort for each task in person-months:

Tasks	Activity	Efforts in person-months
T1	Requirements specification	1
T2	Design	2
T3	Code actuator interface module	2
T4	Code sensor interface module	5

T5	Code user interface part	3
T6	Code control processing part	1
T7	Integrate and test	6
T8	Write user manual	1

2) What is the difference between a macroscopic schedule and a detailed schedule? Is it possible to manage a project if only a microscopic schedule is developed? Discuss with suitable example.

3) Write short notes on the following:

a) WBS

b) CPM

20. What do you mean by earned value analysis and earned value indicators? Discuss various earned value indicators with examples.

21. Discuss error tracking with examples. Does it affect the SPM schedule? Explain.

22. Describe the difference between verification and validation. Do both make use of test case design methods and testing strategies?

23. 1) why is a highly coupled module difficult to unit test? Explain.

2) How can project scheduling affect integration testing? Discuss.

24. What is the difference between a software configuration management audit and a formal technical review? Can their functions be folded into one review? What are the pros and cons?

25. Differentiate between the following with example:

1) Known risks and predictable risks.

2) Change control and version control

26. Write short notes on the following:

1) CASE Tools

2) Risk Monitoring

27. Draw hierarchical organization of various project elements. Discuss each element in brief giving examples.

28 write down various components and their content in the vision and scope document of a project.

28. Planning is the most important activity in the overall Software Project Management. Comment on this statement.

29. What is cost benefit analysis? In context to cost benefit analysis, define the following term precisely.

- Net Profit (NP)
- Payback Period (PP)
- Return on Investment (ROI)
- Net Present Value (NPV)

30. The status of cash flow for four projects is given in the following table. (-ve figures at the end of year 0 represent initial investment).

Cash flow for four projects (figure are end of year totals in rupees)				
0	-100,000	-1,000,000	-100,000	-120,000
1	10,000	200,000	30,000	30,000
2	10,000	200,000	30,000	30,000
3	10,000	200,000	30,000	30,000
4	20,000	200,000	30,000	30,000
5	100,000	300,000	30,000	75,000

On the basis of data, calculate various terms (Q.29) above. You may assume discount rate to be as 10%.

31. 1) what is cash flow forecasting? Draw cash flow for a typical product life cycle.

2) Explain why discounted cash flow technique provides better criteria for project selection than net profit or return on investment.

32. List various methods of estimation. Discuss the alb retch function point count method for estimation of function points. Show that the complexity adjustment factor (CAF) adjusts the unadjusted value of function point(UFP) to +- 35%.

33. Discuss the COCOMO hierarchy of estimation models in details. How these model differ from the dynamic estimation models.

34. 1). Discuss SEI capability maturity model.

2). “software Quality Assurance is an umbrella activity” justify this statement.

35. Statistical quality assurance is done by carrying out a sequence of steps involving collection and classification of errors during all phases of development of the software and following Pareto’s principle. using this methodology derive expression for Error Index which acts as an indicator of the quality.

36. Consider the following information about a one year project.

1) Budgeted cost of work schedule (BCWS) = Rs. 23,000

2) Budgeted cost of work performed (BCWP) = Rs. 20,000

3) Actual cost of work performed (ACWP) = Rs. 25,000

4) Budget at completion (BAC) = Rs. 120000

Answer the following questions:

I) what is the cost variance, schedule variance, Cost Performance Index (CPI), and Schedule Performance Index (SPI) for the project?

II) How is the project doing? Is it ahead of schedule or behind the schedule? Is it under budget or over budget?

III) Use the CPI to calculate the estimate at completion (EAC) for this project. Is the project performing better or worse than planned??

IV) Use the schedule performance index to estimate how long it will take to finish the project.

Solution of Question Bank

Q) What do you understand by software configuration management (SCM)? What are various software configuration items and tasks? Discuss with suitable example?

Answer.

A system can be defined as a collection of components organized to accomplish a specific function or set of functions configuration management, then is the discipline of identifying the configuration of a system at distinct points in time for the purpose of systematically controlling changes to the configuration and maintaining the integrity and traceablilty of the configuration throughout the system life cycle.

A discipline applying technical and administrative direction and surveillance to: identify and document the functional and physical characterstics of a configuration item, control changes to those record an report and report change processing and implementation status and verify compliance with specified requirements. Software configuration item are defined as information that is created as a part of the software engineering process. In the extreme a SCI could be considered to be a single section of a large classification or one test case in a large suite of tests.

If a change were made to the source code object, the interrelationships enable a software engineer to determine what other object and (SCIs) might be affected.

SCI TASKS: There are four procedure that must be defined for each software project to ensure that a second SCM process is implemented. They are:

1) Configuration indentification
2) Configuration control.

Answer) A formal technical review is a software quality assurance acitivity performed by software engineers.

The objectives of the FTR are:

- To uncover errors in functions, logic or implementation for any representation of the software.
- To verify that the software under review meets its requirements.
- To ensure that the software has been represented according to predefined standards.
- To achieve software that is developed in a uniform manner, and
- To mank projects more manageable.

An audit often is called a technical code walkthrough or review. The typical scenario finds a developer inviting his technical lead, a database administrator and one or more peers to a meeting to review a set of source modules prior to production implementation.

Code walkthrough is an informal code analysis technique. Each member selects some test cases and simulates execution of the by hand.

The main objectives of the audit is to discover the algorithm and logical errors in the code.

Benefits of Audits:

- Improved code quality.
- Improved application performance.
- Improved developer performance.

Disadvantages of audit:

- Volumes of data.
- Deleted code.
- Distribution/ logistics.
- Manual Efforts.
- Lack of consistency.
- Developers are reluctant to be involved.

Q. What is software testing? What are software testing objectives? Discuss various types of testing in detail.

Answer.

SOFTWARE TESTING

Software testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results. Although crucial to software quality and widely deployed by programmers and testers, software testing still remains an art, due to limited understanding of the principles of software. The difficulty in software testing stems from the complexity of software: we cannot completely test a program with moderate complexity. Testing is more than just debugging. The purpose of testing can be quality assurance, verification and validation, or reliability estimation. Testing can be used as a generic metric as well. Correctness testing and reliability testing are two major areas of testing. Software testing is a trade-off between budget, time and quality.

Objectives of Software Testing:

Regression testing

Regression testing focuses on finding defects after a major code change has occurred. Specifically, it seeks to uncover software regression, or old bugs that have come back. Such regressions occur whenever software functionality that was previously working correctly

stops working as intended. Typically, regressions occur as an unintended consequence of program changes, when the newly developed part of the software collides with the previously existing code. Common methods of regression testing include re-running previously run tests and checking whether previously fixed faults have re-emerged. The depth of testing depends on the phase in the release process and the risk of the added features. They can either be complete, for changes added late in the release or deemed to be risky, to very shallow, consisting of positive tests on each feature, if the changes are early in the release or deemed to be of low risk.

Acceptance testing

Acceptance testing can mean one of two things:

1. A smoke test is used as an acceptance test prior to introducing a new build to the main testing process, i.e. before integration or regression.
2. Acceptance testing performed by the customer, often in their lab environment on their own hardware, is known as User Acceptance Testing (UAT). Acceptance testing may be performed as part of the hand-off process between any two phases of development.

Alpha testing

Alpha testing is simulated or actual operational testing by potential users/customers or an independent test team at the developers' site. Alpha testing is often employed for off-the-shelf software as a form of internal acceptance testing, before the software goes to beta testing.

Beta testing

Beta testing comes after alpha testing and can be considered a form of external User Acceptance Testing Versions of the software, known as beta versions are released to a limited audience outside of the programming team. The software is released to groups of people so that further testing can ensure the product has few faults or bugs. Sometimes, beta versions are made available to the open public to increase the feedback field to a maximal number of future users.

Various Types of Testing:

Correctness testing

Correctness is the minimum requirement of software, the essential purpose of testing. Correctness testing will need some type of oracle, to tell the right behavior from the wrong one. The tester may or may not know the inside details of the software module under test, e.g. control flow, data flow, etc. Therefore, either a white-box point of view or black-box

point of view can be taken in testing software. We must note that the black-box and white-box ideas are not limited in correctness testing only.

- **Black-box testing**

  - The black-box approach is a testing method in which test data are derived from the specified functional requirements without regard to the final program structure. It is also termed data-driven, input/output driven or requirements-based testing. Because only the functionality of the software module is of concern, black-box testing also mainly refers to functional testing -- a testing method emphasized on executing the functions and examination of their input and output data. The tester treats the software under test as a black box -- only the inputs, outputs and specification are visible, and the functionality is determined by observing the outputs to corresponding inputs. In testing, various inputs are exercised and the outputs are compared against specification to validate the correctness. All test cases are derived from the specification. No implementation details of the code are considered.

- **White-box testing**

Contrary to black-box testing, software is viewed as a white-box, or glass-box in white-box testing, as the structure and flow of the software under test are visible to the tester. Testing plans are made according to the details of the software implementation, such as programming language, logic, and styles. Test cases are derived from the program structure. White-box testing is also called glass-box testing, logic-driven testing or design-based testing.

There are many techniques available in white-box testing, because the problem of intractability is eased by specific knowledge and attention on the structure of the software under test. The intention of exhausting some aspect of the software is still strong in white-box testing, and some degree of exhaustion can be achieved, such as executing each line of code at least once (statement coverage), traverse every branch statements (branch coverage), or cover all the possible combinations of true and false condition predicates.

Reliability testing

Software reliability refers to the probability of failure-free operation of a system. It is related to many aspects of software, including the testing process. Directly estimating software reliability by quantifying its related factors can be difficult. Testing is an effective sampling method to measure software reliability. Guided by the operational profile, software testing (usually black-box testing) can be used to obtain failure data, and an estimation model can be further used to analyze the data to estimate the present reliability and predict future reliability. Therefore, based on the estimation, the developers can decide whether to release the software, and the users can decide whether to adopt and use the software. Risk of using software can also be assessed based on reliability information

advocates that the primary goal of testing should be to measure the dependability of tested software.
Stress testing, or load testing, is often used to test the whole system rather than the software alone. In such tests the software or system are exercised with or beyond the specified limits. Typical stress includes resource exhaustion, bursts of activities, and sustained high loads.

Security testing

Software quality, reliability and security are tightly coupled. Flaws in software can be exploited by intruders to open security holes. With the development of the Internet, software security problems are becoming even more severe.

Many critical software applications and services have integrated security measures against malicious attacks. The purpose of security testing of these systems include identifying and removing software flaws that may potentially lead to security violations, and validating the effectiveness of security measures. Simulated security attacks can be performed to find vulnerabilities.

Q.10) 2) write a short note on testing automation and testing tools.

Answer.

Testing automation

Software testing can be very costly. Automation is a good way to cut down time and cost. Software testing tools and techniques usually suffer from a lack of generic applicability and scalability. The reason is straight-forward. In order to automate the process, we have to have some ways to generate oracles from the specification, and generate test cases to test the target software against the oracles to decide their correctness. Today we still don't have a full-scale system that has achieved this goal. In general, significant amount of human intervention is still needed in testing. The degree of automation remains at the automated test script level.

The problem is lessened in reliability testing and performance testing. In robustness testing, the simple specification and oracle: doesn't crash, doesn't hang suffices. Similar simple metrics can also be used in stress testing.

When to stop testing?

Testing is potentially endless. We can not test till all the defects are unearthed and removed -- it is simply impossible. At some point, we have to stop testing and ship the software. The question is when.

Realistically, testing is a trade-off between budget, time and quality. It is driven by profit models. The pessimistic and unfortunately most often used approach is to stop testing whenever some or any of the allocated resources -- time, budget, or test cases -- are exhausted. The optimistic stopping rule is to stop testing when either reliability meets the requirement, or the benefit from continuing testing cannot justify the testing cost. This will usually require the use of reliability models to evaluate and predict reliability of the software under test. Each evaluation requires repeated running of the following cycle: failure data gathering -- modeling -- prediction. This method does not fit well for ultra-dependable systems, however, because the real field failure data will take too long to accumulate.

Available tools, techniques, and metrics

There are an abundance of software testing tools exist. The correctness testing tools are often specialized to certain systems and have limited ability and generality. Robustness and stress testing tools are more likely to be made generic.

Mothora is an automated mutation testing tool-set developed at Purdue University. Using Mothora, the tester can create and execute test cases, measure test case adequacy, determine input-output correctness, locate and remove faults or bugs, and control and document the test.

NuMega's Boundschecker Rational's Purify. They are run-time checking and debugging aids. They can both check and protect against memory leaks and pointer problems.

Ballista COTS Software Robustness Testing Harness .The Ballista testing harness is an full-scale automated robustness testing tool. The first version supports testing up to 233 POSIX function calls in UNIX operating systems. The second version also supports testing of user functions provided that the data types are recognized by the testing server. The Ballista testing harness gives quantitative measures of robustness comparisons across operating systems. The goal is to automatically test and harden Commercial Off-The-Shelf (COTS) software against robustness failures.

Question: Define Software Project Planning. What is Project Planning?

Answer.

This article explores the various aspects of Software Project Planning and Scheduling. Project planning is an aspect of Project Management, which comprises of various processes. The aim of theses processes is to ensure that various Project tasks are well coordinated and they meet the various project objectives including timely completion of the project

Project Planning is an aspect of Project Management that focuses a lot on Project Integration. The project plan reflects the current status of all project activities and is used to monitor and control the project.

The Project Planning tasks ensure that various elements of the Project are coordinated and therefore guide the project execution.

Project Planning helps in
- Facilitating communication
- Monitoring/measuring the project progress, and
- Provides overall documentation of assumptions/planning decisions

The Project Planning Phases can be broadly classified as follows:
- Development of the Project Plan
- Execution of the Project Plan
- Change Control and Corrective Actions

Project Planning is an ongoing effort throughout the Project Lifecycle.

Objectives:Why is it important?

"If you fail to plan, you plan to fail."
Project planning is crucial to the success of the Project.
Careful planning right from the beginning of the project can help to avoid costly mistakes.
It provides an assurance that the project execution will accomplish its goals on schedule and within budget.

What are the steps in Project Planning?

Project Planning spans across the various aspects of the Project. Generally Project Planning is considered to be a process of estimating, scheduling and assigning the projects resources in order to deliver an end product of suitable quality. However it is much more as it can assume a very strategic role, which can determine the very success of the project. A Project Plan is one of the crucial steps in Project Planning in General!

Typically Project Planning can include the following **types of project Planning**:
1) Project Scope Definition and Scope Planning
2) Project Activity Definition and Activity Sequencing
3) Time, Effort and Resource Estimation
4) Risk Factors Identification
5) Cost Estimation and Budgeting
6) Organizational and Resource Planning
7) Schedule Development
8) Quality Planning
9) Risk Management Planning
10) Project Plan Development and Execution
11) Performance Reporting

12) Planning Change Management
13) Project Rollout Planning

We now briefly examine each of the above steps:

1) Project Scope Definition and Scope Planning:
In this step we document the project work that would help us achieve the project goal. We document the assumptions, constraints, user expectations, Business Requirements, Technical requirements, project deliverables, project objectives and everything that defines the final product requirements. This is the foundation for a successful project completion.

2) Quality Planning:
The relevant quality standards are determined for the project. This is an important aspect of Project Planning. Based on the inputs captured in the previous steps such as the Project Scope, Requirements, deliverables, etc. various factors influencing the quality of the final product are determined. The processes required to deliver the Product as promised and as per the standards are defined.

3) Project Activity Definition and Activity Sequencing:
In this step we define all the specific activities that must be performed to deliver the product by producing the various product deliverables. The Project Activity sequencing identifies the interdependence of all the activities defined.

4) Time, Effort and Resource Estimation:
Once the Scope, Activities and Activity interdependence is clearly defined and documented, the next crucial step is to determine the effort required to complete each of the activities. See the article on "Software Cost Estimation" for more details. The Effort can be calculated using one of the many techniques available such as Function Points, Lines of Code, Complexity of Code, Benchmarks, etc.
This step clearly estimates and documents the time, effort and resource required for each activity.

5) Risk Factors Identification:
"Expecting the unexpected and facing it"
It is important to identify and document the risk factors associated with the project based on the assumptions, constraints, user expectations, specific circumstances, etc.

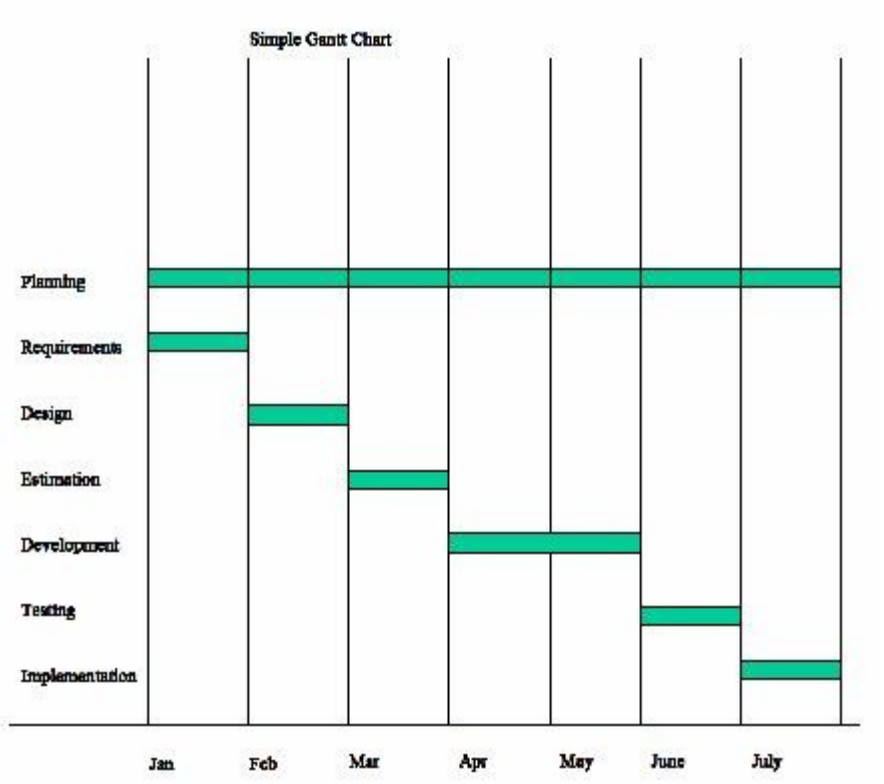
6) Schedule Development:
The time schedule for the project can be arrived at based on the activities, interdependence and effort required for each of them. The schedule may influence the cost estimates, the cost benefit analysis and so on.

Project Scheduling is one of the most important task of Project Planning and also the most difficult tasks. In very large projects it is possible that several teams work on developing the project. They may work on it in parallel. However their work may be interdependent.

Again various factors may impact in successfully scheduling a project

- o Teams not directly under our control
- o Resources with not enough experience

Popular Tools can be used for creating and reporting the schedules such as Gantt Charts



7) Cost Estimation and Budgeting:

Based on the information collected in all the previous steps it is possible to estimate the cost involved in executing and implementing the project. See the article on "Software Cost Estimation" for more details. A Cost Benefit Analysis can be arrived at for the project. Based on the Cost Estimates Budget allocation is done for the project.

8) Organizational and Resource Planning

Based on the activities identified, schedule and budget allocation resource types and resources are identified. One of the primary goals of Resource planning is to ensure that the project is run efficiently. This can only be achieved by keeping all the project resources fully utilized as possible. The success depends on the accuracy in predicting the resource demands that will be placed on the project. Resource planning is an iterative process and necessary to optimize the use of resources throughout the project life cycle thus making the

project execution more efficient. There are various types of resources – Equipment, Personnel, Facilities, Money, etc.

9) Risk Management Planning:
Risk Management is a process of identifying, analyzing and responding to a risk. Based on the Risk factors Identified a Risk resolution Plan is created. The plan analyses each of the risk factors and their impact on the project. The possible responses for each of them can be planned. Throughout the lifetime of the project these risk factors are monitored and acted upon as necessary.

10) Project Plan Development and Execution:
Project Plan Development uses the inputs gathered from all the other planning processes such as Scope definition, Activity identification, Activity sequencing, Quality Management Planning, etc. A detailed Work Break down structure comprising of all the activities identified is used. The tasks are scheduled based on the inputs captured in the steps previously described. The Project Plan documents all the assumptions, activities, schedule, timelines and drives the project.

Each of the Project tasks and activities are periodically monitored. The team and the stakeholders are informed of the progress. This serves as an excellent communication mechanism. Any delays are analyzed and the project plan may be adjusted accordingly

11) Performance Reporting:
As described above the progress of each of the tasks/activities described in the Project plan is monitored. The progress is compared with the schedule and timelines documented in the Project Plan. Various techniques are used to measure and report the project performance such as EVM (Earned Value Management) A wide variety of tools can be used to report the performance of the project such as PERT Charts, GANTT charts, Logical Bar Charts, Histograms, Pie Charts, etc.

12) Planning Change Management:
Analysis of project performance can necessitate that certain aspects of the project be changed. The Requests for Changes need to be analyzed carefully and its impact on the project should be studied. Considering all these aspects the Project Plan may be modified to accommodate this request for Change

Change Management is also necessary to accommodate the implementation of the project currently under development in the production environment. When the new product is implemented in the production environment it should not negatively impact the environment or the performance of other applications sharing the same hosting environment.

13) Project Rollout Planning:

In Enterprise environments, the success of the Project depends a great deal on the success of its rollout and implementations. Whenever a Project is rolled out it may affect the technical systems, business systems and sometimes even the way business is run. For an application to be successfully implemented not only the technical environment should be ready but the users should accept it and use it effectively. For this to happen the users may need to be trained on the new system. All this requires planning.

Q. What do you understand by software configuration management (SCM)? What are various software configuration items and tasks? Discuss with suitable example.

Answer.

In software engineering, **software configuration management (SCM)** is the task of tracking and controlling changes in the software. Configuration management practices include revision control and the establishment of baselines.

SCM concerns itself with answering the question "Somebody did something, how can one reproduce it?" Often the problem involves not reproducing "it" identically, but with controlled, incremental changes. Answering the question thus becomes a matter of comparing different results and of analysing their differences. Traditional configuration management typically focused on controlled creation of relatively simple products. Now, implementers of SCM face the challenge of dealing with relatively minor increments under their own control, in the context of the complex system being developed. According to another simple definition: Software Configuration Management is how you control the evolution of a software project.

Software CM involves four key functions:

1. *identification* of work products and baselines that are subject to configuration control
2. *control* (i.e., approval/rejection) of proposed changes to configuration items
3. *status accounting* of configuration data and changes
4. *Verification* – this task involves reviews and audits to ensure the information contained in the CMDB is accurate.

Configuration Item (CI) - an aggregation of work products (e.g., hardware, software, firmware, or documentation) that is designated for configuration management and treated as a single entity in the configuration management process

Baseline – a set of specifications or work products that has been formally reviewed and agreed on, which thereafter serves as the basis for further development, and which can be changed only through change control procedures. *Typically* defined for each project life-cycle phase

Q. What is the difference between a software configuration management audit and a formal technical review? Can their functions be folded into one review?

Answer.
To ensure that the change has been properly implemented following steps are carried out
(1) formal technical reviews and
(2) the software configuration audit

The formal technical review focuses on the technical correctness of the configuration object that has been modified. The reviewers assess the SCI to determine consistency with other SCIs, omissions, or potential side effects. A formal technical review should be conducted for all but the most trivial changes.
A software configuration audit complements the formal technical review by assessing a configuration object for characteristics that are generally not considered during review.
The audit asks and answers the following questions:

1. Has the change specified in the ECO been made? Have any additional modifications been incorporated?
2. Has a formal technical review been conducted to assess technical correctness?
3. Has the software process been followed and have software engineering standards been properly applied?
4. Has the change been "highlighted" in the SCI? Have the change date and change author been specified? Do the attributes of the configuration object reflect the change?
5. Have SCM procedures for noting the change, recording it, and reporting it been followed?
6. Have all related SCIs been properly updated?

In some cases, the audit questions are asked as part of a formal technical review. However, when SCM is a formal activity, the SCM audit is conducted separately by the quality assurance
group.

Configuration status reporting (sometimes called *status accounting)* is an SCM task that answers the following questions: (1) what happened? (2) Who did it? (3) When did it happen? (4) What else will be affected?

Configuration status reporting plays a vital role in the success of a large software development project. When many people are involved, it is likely that "the left hand not knowing what the right hand is doing" syndrome will occur.

Two developers may attempt to modify the same SCI with different and conflicting intents. A software engineering team may spend months of effort building software to an obsolete hardware specification. The person who would recognize serious side effects for a proposed change is not aware that the change is being made. CSR helps to eliminate the problems by improving communication among all people involved.

Q. What is the difference b/w Product Life Cycle and Project Life Cycle?

Answer.

Product Life Cycle

The product life cycle represents the amount of revenue a product generates over time, from its inception to the point where it is discontinued.

The five stages of a product's life are

1) development
2) introduction
3) growth,
4) maturity
5) decline

In the development stage, the product isn't yet being sold, so there is no revenue. During introduction, sales are small as people begin to try the product. Sales will increase during the growth phase, peak during maturity, and eventually decline as the market shifts or better alternatives become available. There is no set time span for a given stage; the entire cycle may last only months or a product like the refrigerator may remain in the maturity phase for decades.

Project Life Cycle

A project life cycle measures the work that goes into a project from beginning to end. The phases in product life cycle are

1) initiation
2) planning
3) Execution
4) Closure

During initiation, a business case and goals are created, and resources are assigned. During planning, the team researches solutions to reach the project goals and creates a plan and timeline to complete the project. Execution involves following each step on the project plan and adjusting as necessary along the way. Finally, in the closure phase, the project's final details are wrapped up and deliverable items like final reports are given to the appropriate parties.

Q. Differences between the Two

Project is the one which is executed to create a unique product or services and Product is the outcome of a Project.

Answer.

A product life cycle is a conceptual map of where a product's sales are and where they may be headed. However, it has no comment on what to do with the product. If a company believes its product is entering the decline phase, it will probably create a plan to either rejuvenate the product or cease production, but that is not inherent in the product life cycle. By contrast, a project life cycle is all about action. A project life cycle maps out the steps needed to complete a project with specific targeted results.

A product life cycle talks about the various stages a product goes through before it actually declines on the other hand the project life cycle is the set of activities that you need to complete to finish the project. This means that the stages in the product life cycle are controlled by external factors like entry of competitors, the cost structure, customer response etc but project life cycle is controlled by the one who is carrying it out. All products go through the same stages, some progress quickly while others do so slowly. The steps in the project life cycle can be delayed or skipped but this is not possible in the product life cycle. Generally people try to speed up the project and complete it as early as possible while they try to slow down the product life cycle stages.

Strategies

Remember that the product life cycle concept has limitations. Not every product follows a smooth, predictable bell curve from introduction to decline. A product may appear to be in the decline phase and enjoy a return to the maturity phase due to a competitor exiting the market or a successful project rejuvenation strategy. With regards to project life cycle management, things tend to be much more clearly defined, but watch out for "scope creep." This is the tendency for projects to continually grow in breadth to the point where they never actually get completed.

Generally, a project life cycle is contained within one or more product life cycles.

Q Write a short note on organizational behavior?

Answer.

The study of Organizational Behaviour (OB) is very interesting and challenging too. It is related to individuals, group of people working together in teams. The study becomes more challenging when situational factors interact. The study of organizational behaviour re-lates to the expected behaviour of an individual in the organization. No two individuals are likely to behave in the same manner in a particular work situation. It is the predict- ability of a

manager about the expected behaviour of an individual. There are no absolutes in human behaviour. It is the human factor that is contributory to the productivity hence the study of human behaviour is important. Great importance therefore must be attached to the study. Researchers, management practitioners, psychologists, and social scientists must understand the very credentials of an individual, his background, social framework, educational update, impact of social groups and other situational factors on behaviour. Managers under whom an individual is working should be able to explain, predict, evalu- ate and modify human behaviour that will largely depend upon knowledge, skill and experience of the manager in handling large group of people in diverse situations. Pre- emptive actions need to be taken for human behaviour forecasting. The value system, emotional intelligence, organizational culture, job design and the work environment are important causal agents in determining human behaviour.

The scope of the organizational behaviour is as under:

(a) Impact of personality on performance
(b) Employee motivation
(c) Leadership
(d) How to create effective teams and group

(e) Study of different organizational structures
(f) Individual behaviour, attitude and learning
(g) Perception
(h) Design and development of effective organization
(i) Job design
(j) Impact of culture on organizational behaviour
(k) Management of change
(l) Management of conflict and stress
(m) Organizational development
(n) Organizational culture
(o) Transactional analysis
(p) Group behaviour, power and politics
(q) Job design
(r) Study of emotions

"Organizational behaviour is a field of study that investigates the impact that individuals, groups and organizational structure have on behaviour within the organization, for the purpose of applying such knowledge towards improving an

organizational effectiveness". The above definition has three main elements; **first** organizational behaviour is an investigative study of individuals and groups, **second**, the impact of organizational structure on human behaviour and the **third**, the application of knowledge to achieve organizational effectiveness. These factors are interactive in nature and the impact of such behaviour is applied to various systems so that the goals are achieved. The nature of study of organizational behaviour is investigative to establish cause and effect relationship.

OB involves integration of studies undertaken relating to behavioural sciences like psychology, sociology, anthropology, economics, social psychology and political science. Therefore, organizational behaviour is a comprehensive field of study in which individual, group and organizational structure is studied in relation to organizational growth and organizational culture, in an environment where impact of modern technology is great. The aim of the study is to ensure that the human behaviour contributes towards growth
of the organization and greater efficiency is achieved.

Organizational behaviour can be defined as – **"the study and application of knowledge about human behaviour related to other elements of an organization such as structure, technology and social systems** (LM Prasad**)**. Stephen P Robins defines **"Organizational behaviour as a systematic study of the actions and attitudes that people exhibit within organizations."**

Q. List various methods of estimation. Discuss the albretch function point count method for estimation of function points. Show that the complexity adjustment factor (CAF) adjusts the unadjusted value of function point(UFP) to +- 35%.

Answer.

Popular estimation processes for software projects include:

- Cocomo
- Cosysmo
- Event chain methodology
- Function points
- Program Evaluation and Review Technique (PERT)

- Proxy Based Estimation (PROBE) (from the Personal Software Process)
- The Planning Game (from Extreme Programming)
- Weighted Micro Function Points (WMFP)
- Wideband Delphi

ALBRECHT'S FUNCTION POINT METHOD

Function Point Analysis was developed first by Allan J. Albrecht in the mid 1970s. It was an attempt to overcome difficulties associated with lines of code as a measure of software size, and to assist in developing a mechanism to predict effort associated with software development.

Characteristic of Quality Function Point Analysis

Function Point Analysis should be performed by trained and experienced personnel. If Function Point Analysis is conducted by untrained personnel, it is reasonable to assume the analysis will be done incorrectly. The personnel counting function points should utilize the most current version of the Function Point Counting Practices Manual,Current application documentation should be utilized to complete a function point count. For example, screen formats, report layouts, listing of interfaces with other systems and between systems, logical and/or preliminary physical data models will all assist in Function Points Analysis.

The Five Major Components

Since it is common for computer systems to interact with other computer systems, a boundary must be drawn around each system to be measured prior to classifying components. This boundary must be drawn according to the user's point of view. In short, the boundary indicates the border between the project or application being measured and the external applications or user domain. Once the border has been established, components can be classified, ranked and tallied.

External Inputs (EI) - is an elementary process in which data crosses the boundary from outside to inside. This data may come from a data input screen or another application. The data may be used to maintain one or more internal logical files. The data can be either control information or business information. If the data is control information it does not have to update an internal logical file. The graphic represents a simple EI that updates 2 ILF's (FTR's).

External Outputs (EO) - an elementary process in which derived data passes across the boundary from inside to outside. Additionally, an EO may update an ILF. The data creates reports or output files sent to other applications. These reports and files are created from one or more internal logical files and external interface file. The following graphic represents on EO with 2 FTR's there is derived information (green) that has been derived from the ILF's

External Inquiry (EQ) - an elementary process with both input and output components that result in data retrieval from one or more internal logical files and external interface

files. The input process does not update any Internal Logical Files, and the output side does not contain derived data. The graphic below represents an EQ with two ILF's and no derived data.

Internal Logical Files (ILF's) - a user identifiable group of logically related data that resides entirely within the applications boundary and is maintained through external inputs.

External Interface Files (EIF's) - a user identifiable group of logically related data that is used for reference purposes only. The data resides entirely outside the application and is maintained by another application. The external interface file is an internal logical file for another application.

After the components have been classified as one of the five major components (EI's, EO's, EQ's, ILF's or EIF's), a ranking of low, average or high is assigned. For transactions (EI's, EO's, EQ's) the ranking is based upon the number of files updated or referenced (FTR's) and the number of data element types (DET's). For both ILF's and EIF's files the ranking is based upon record element types (RET's) and data element types (DET's). A record element type is a user recognizable subgroup of data elements within an ILF or EIF. A data element type is a unique user recognizable, non recursive, field.

Each of the following tables assists in the ranking process (the numerical rating is in parentheses). For example, an EI that references or updates 2 File Types Referenced (FTR's) and has 7 data elements would be assigned a ranking of average and associated rating of 4. Where FTR's are the combined number of Internal Logical Files (ILF's) referenced or updated and External Interface Files referenced.

EI Table

FTR's	DATA ELEMENTS		
	1-4	5-15	> 15
0-1	Low	Low	Ave
2	Low	Ave	High
3 or more	Ave	High	High

Shared EO and EQ Table

FTR's	DATA ELEMENTS		
	1-5	6-19	> 19
0-1	Low	Low	Ave
2-3	Low	Ave	High
> 3	Ave	High	High

Values for transactions

Rating	VALUES		
	EO	EQ	EI
Low	4	3	3
Average	5	4	4
High	7	6	6

Like all components, EQâ€™s are rated and scored. Basically, an EQ is rated (Low, Average or High) like an EO, but assigned a value like and EI. The rating is based upon the total number of unique (combined unique input and out sides) data elements (DETâ€™s) and the file types referenced (FTRâ€™s) (combined unique input and output sides). If the same FTR is used on both the input and output side, then it is counted only one time. If the same DET is used on both the input and output side, then it is only counted one time.

For both ILF's and EIF's the number of record element types and the number of data elements types are used to determine a ranking of low, average or high. A Record Element Type is a user recognizable subgroup of data elements within an ILF or EIF. A Data Element Type (DET) is a unique user recognizable, non recursive field on an ILF or EIF.

RET's	DATA ELEMENTS		
	1-19	20 - 50	> 50
1	Low	Low	Ave
2-5	Low	Ave	High
> 5	Ave	High	High

Rating	Values	
	ILF	EIF
Low	7	5
Average	10	7
High	15	10

The counts for each level of complexity for each type of component can be entered into a table such as the following one. Each count is multiplied by the numerical rating shown to determine the rated value. The rated values on each row are summed across the table, giving a total value for each type of component. These totals are then summed across the table, giving a total value for each type of component. These totals are then summoned down to arrive at the Total Number of Unadjusted Function Points.

The value adjustment factor (VAF) is based on 14 general system characteristics (GSC's) that rate the general functionality of the application being counted. Each characteristic has associated descriptions that help determine the degrees of influence of the characteristics. The degrees of influence range on a scale of zero to five, from no influence to strong influence. The IFPUG Counting Practices Manual provides detailed evaluation criteria for each of the GSC'S, the table below is intended to provide an overview of each GSC.

General System Characteristic	Brief Description

1.	Data communications	How many communication facilities are there to aid in the transfer or exchange of information with the application or system?
2.	Distributed data processing	How are distributed data and processing functions handled?
3.	Performance	Was response time or throughput required by the user?
4.	Heavily used configuration	How heavily used is the current hardware platform where the application will be executed?
5.	Transaction rate	How frequently are transactions executed daily, weekly, monthly, etc.?
6.	On-Line data entry	What percentage of the information is entered On-Line?
7.	End-user efficiency	Was the application designed for end-user efficiency?
8.	On-Line update	How many ILF's are updated by On-Line transaction?
9.	Complex processing	Does the application have extensive logical or mathematical processing?
10.	Reusability	Was the application developed to meet one or many user's needs?
11.	Installation ease	How difficult is conversion and installation?
12.	Operational ease	How effective and/or automated are start-up, back-up, and recovery procedures?
13.	Multiple sites	Was the application specifically designed, developed, and supported to be installed at multiple sites for multiple organizations?
14.	Facilitate change	Was the application specifically designed, developed, and supported to facilitate change?

Once all the 14 GSC's have been answered, they should be tabulated using the IFPUG Value Adjustment Equation (VAF) --

14 where: Ci = degree of influence for each General System Characteristic

VAF = 0.65 + [(Ci) / 100] .i = is from 1 to 14 representing each GSC.

i =1 Ã¥ = is summation of all 14 GSC’s.

The final Function Point Count is obtained by multiplying the VAF times the Unadjusted Function Point (UAF).

FP = UAF * VAF

Q. Describe the following:-

Answer.

1-VISION AND SCOPE DOCUMENT-

PROBLEM STATEMENT-

a) Project background-This Section contains a summary of the problem that the project wills solve.
b) Stakeholders-this is the bulleted list of stakeholders.each stakeholder may be refferd to by name or by title.
c) Users- this is the bulleted list of users.each stakeholder may be refferd to by name or by title.
d) Risks-this section list any potential risk to the project.
e) Assumptions-This the list of assumptions that the stakeholders ,usersor project team have made.
Vision of the solution-
a)vision statement
b)list of features
C)scope of phased released

2-MANAGEMENT SPECTRUM

4P's in Project Management Spectrum

- **People**
- **Product**
- **Process**

- **Project**

PEOPLE

- the most important factor in success of software project.
- “Companies That sensibly manage their investment in people will prosper in the long run” .
- Cultivation of motivated and highly skilled software people has always been important for software organizations.

The “people-factor” is so important that has developed People Management Capability Maturity Model (PM-CMM

PRODUCT

- The product and the problem it is intended to solve must be examined at very beginning of the software project.
- The scope of product must be established and bounded.
    - Bounded scope means
        - establishing quantitative data like no. of simultaneous users, max. allowable response time. etc.

PROCESS

- These characterize a software process and are applicable to all software projects
    - Communication
    - Planning
    - Modeling

- Construction
- Deployment

- These are applied to software engineering work tasks (e.g., different product functions)

PROJECT

- Software people don't understand customer needs
- Product scope is poorly defined
- Changes are managed poorly
- The chosen technology changes
- Business needs change
- Deadlines are unrealistic

3-DECISION PROCESS

Decision making can be regarded as the mental processes resulting in the selection of a course of action among several alternative scenarios. Every decision making process produces a final choice. The output can be an action or an opinion of choice.

Objectives must first be established

- Objectives must be classified and placed in order of importance
- Alternative actions must be developed
- The alternative must be evaluated against all the objectives
- The alternative that is able to achieve all the objectives is the tentative decision

- The tentative decision is evaluated for more possible consequences
- The decisive actions are taken, and additional actions are taken to prevent any adverse consequences from becoming problems and starting both systems (problem analysis and decision making) all over again
- There are steps that are generally followed that result in a decision model that can be used to determine an optimal production plan.[5]
- In a situation featuring conflict, role-playing is helpful for predicting decisions to be made by involved parties.[6]

Q. Discuss capability maturity model

Answer.

Introduction

The **Capability Maturity Model (CMM)** is a theoretical *process* capability maturity model.
The CMM was originally developed as a tool for objectively assessing the ability of government contractors'*processes* to perform a contracted software project. For this reason, it has been used extensively for avionics software and government projects around the world. The 5-Level structure of the CMM can be illustrated by the diagram below (Figure 1).

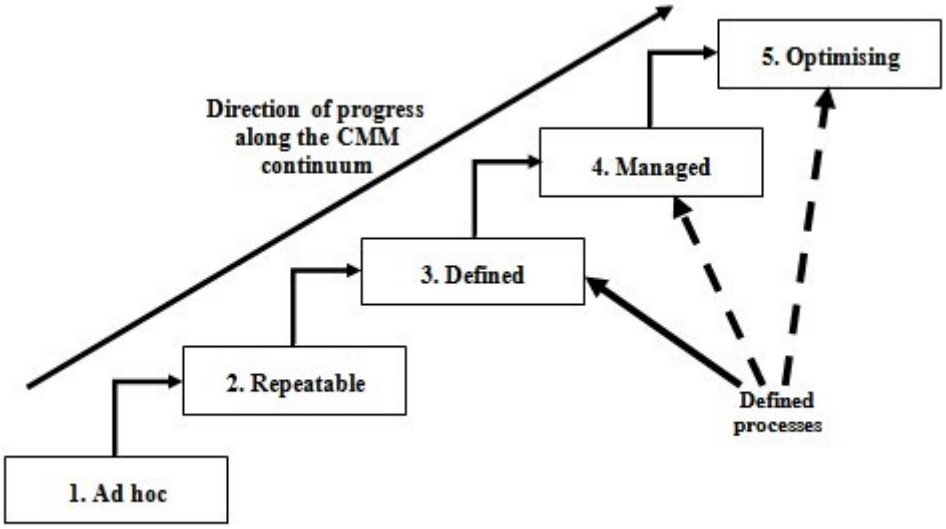


Figure 1: Diagram of the CMM

Although the CMM comes from the area of software development, it can be (and has been and still is being) applied as a generally applicable model to assist in understanding the process capability maturity of organization in areas as diverse as, for example: software engineering, system engineering, project management, software maintenance, risk management, system acquisition, information technology (IT), and personnel management.

Structure of the CMM

The CMM involves the following aspects:

- ***Maturity Levels****:* A 5-Level process maturity continuum - where the uppermost (5th) level is a notional ideal state where processes would be systematically managed by a combination of process optimization and continuous process improvement.
- ***Key Process Areas****:* Within each of these maturity levels are Key Process Areas (KPAs) which characterise that level, and for each KPA there are five definitions identified:
  - Goals
  - Commitment
  - Ability

- Measurement
- Verification

The KPAs are not necessarily unique to CMM, representing - as they do - the stages that organisations' processes will need to pass through as they progress up the CMM continuum.

- ***Goals:*** The goals of a key process area summarize the states that must exist for that key process area to have been implemented in an effective and lasting way. The extent to which the goals have been accomplished is an indicator of how much capability the organisation has established at that maturity level. The goals signify the scope, boundaries, and intent of each key process area.
- ***Common Features****:* Common features include practices that implement and institutionalize a key process area. There are five types of common features: Commitment to Perform, Ability to Perform, Activities Performed, Measurement and Analysis, and Verifying Implementation.
- ***Key Practices****:* The key practices describe the elements of infrastructure and practice that contribute most effectively to the implementation and institutionalization of the KPAs.

Levels of the CMM

- At the ***initial*** level, processes are disorganized, even chaotic. Success is likely to depend on individual efforts, and is not considered to be repeatable, because processes would not be sufficiently defined and documented to allow them to be replicated.
- At the ***repeatable*** level, basic project management techniques are established, and successes could be repeated, because the requisite processes would have been made established, defined, and documented.
- At the ***defined*** level, an organization has developed its own standard software process through greater attention to documentation, standardization, and integration.

- At the ***managed*** level, an organization monitors and controls its own processes through data collection and analysis.
- At the ***optimizing*** level, processes are constantly being improved through monitoring feedback from current processes and introducing innovative processes to better serve the organization's particular needs.

Q. SOFTWARE QUALITY ASSURANCE IS AN UMBRELLA ACTIVITY

Answer

quality assurance is an umbrella activity that is applied throughout the software process.

SQA encompasses:

(1) a quality management approach
(2) effective software engineering technology
(3) formal technical reviews
(4) a multi-tiered testing strategy
(5) document change control
(6) software development standard and its control procedure
(7) measurement and reporting mechanism

Quality --> refers to measurable characteristics of a software.
These items can be compared based on a given standard

Two types of quality control:

- Quality design -> the characteristics that designers specify for an item.

--> includes: requirements, specifications, and the design of the system.

- Quality of conformance -> the degree to which the design specification are followed. It focuses on implementation based on the design.

Q. What is the difference b/w Product Life Cycle and Project Life Cycle?

Answer.

Product Life Cycle

The product life cycle represents the amount of revenue a product generates over time, from its inception to the point where it is discontinued.

The five stages of a product's life are

1) development

2) introduction

3) growth,

4) maturity

5) decline

In the development stage, the product isn't yet being sold, so there is no revenue. During introduction, sales are small as people begin to try the product. Sales will increase during the growth phase, peak during maturity, and eventually decline as the market shifts or better alternatives become available. There is no set time span for a given stage; the entire cycle may last only months or a product like the refrigerator may remain in the maturity phase for decades.

Project Life Cycle

A project life cycle measures the work that goes into a project from beginning to end. The phases in product life cycle are

1) initiation

2) planning

3) Execution

4) Closure

During initiation, a business case and goals are created, and resources are assigned. During planning, the team researches solutions to reach the project goals and creates a plan and timeline to complete the project. Execution

involves following each step on the project plan and adjusting as necessary along the way. Finally, in the closure phase, the project's final details are wrapped up and deliverable items like final reports are given to the appropriate parties.