Answer Key –III Internal

Subject Code- 13MCA33 Subject Name: Software Engineering

Q1(a): Explain the concept of risk assessment and mention the techniques for managing them.

Ans: A risk is a probabilistic event—it may or may not occur. The goal of risk assessment is to prioritize the risks so that attention and resources can be focused on the more risky items.

Risk identification is the first step in risk assessment, which identifies all the different risks for a particular project. Methods that can aid risk identification include checklists of possible risks, surveys, meetings and brainstorming, and reviews of plans, processes, and work products. Checklists of frequently occurring risks are probably the most common tool for risk identification. Such a list can form the starting point for identifying risks for the current project.

One approach for prioritization the risk is through the concept of risk exposure (RE) , which is sometimes called risk impact.

RE = Prob(UO) Loss(UO),

Where Prob(UO) is the probability of the risk materializing (i.e., undesirable outcome) and Loss(UO) is the total loss incurred due to the unsatisfactory outcome. The RE is the expected value of the loss due to a particular risk. The higher the RE, the higher the priority of the risk item.

The top-ranked risk item is personnel shortfalls. This involves just having fewer people than necessary or not having people with specific skills that a project might require. To manage this risk are to get the top talent possible and to match the needs of the project with the skills of the available personnel. Adequate training, along with having some key personnel for critical areas of the project, will also reduce this risk.

The second item, unrealistic schedules and budgets, happens very frequently due to business and other reasons. It is very common that high-level management imposes a schedule for a software project that is not based on the characteristics of the project and is unrealistic. Underestimation may also happen due to inexperience or optimism. Detailed cost and schedule estimation, Design to cost, Incremental development, Software reuse, Requirements scrubbing are few techniques to deal with these kind of risk.
Projects run the risk of developing the wrong software if the requirements analysis is not done properly and if development begins too early. Organization analysis, Machine analysis, User surveys, Prototyping, Early user's manuals are few techniques to deal with these kind of risk.

Similarly, often improper user interface may be developed. This requires extensive rework of the user interface later. Prototyping, Scenarios, Task analysis, User characterization are few techniques to deal with these kind of risk.

Gold plating refers to adding features in the software that are only marginally useful. This adds unnecessary risk to the project because gold plating consumes resources and time with little return. Requirements scrubbing, Prototyping, Cost benefit analysis, Design to cost are few techniques to deal with these kind of risk.

Q2(a) : What is Project Monitoring plan?

Ans: A project management plan is merely a document that can be used to guide the execution of a project. Even a good plan is useless unless it is properly executed. And execution cannot be properly driven by the plan unless it is monitored carefully and the actual performance is tracked against the plan. Monitoring requires measurements to be made to assess the situation of a project.

If measurements are to be taken during project execution, we must plan carefully regarding what to measure, when to measure, and how to measure. Hence, measurement planning is a key element in project planning. For monitoring the state of a project, size, effort, schedule, and defects are the basic measurements that are needed. Schedule is one of the most important metrics because most projects are driven by schedules and deadlines. It is easy to measure because calendar time is usually used in all plans.

Effort is the main resource consumed in a software project. Some type of timesheet system is needed where each person working on the project enters the amount of time spent on the project. The effort spent on various tasks should be logged separately. Effort is recorded through some on-line system which allows a person to record the amount of time spent against a particular activity in a project. At any point, total effort on an activity can be aggregated.

Q2(b) : Explain the Risk management planning approach.

Ans: Following are the steps involved in Risk Management and Planning Approach:

1. For each risk, rate the probability of its happening as low, medium, or high.
2. For each risk, assess its impact on the project as low, medium, or high.
3. Rank the risks based on the probability and effects on the project; for example, a high-probability, high-impact item will have higher rank than a risk item with a medium probability and high impact. In case of conflict,use judgment.

4. Select the top few risk items for mitigation and tracking.

Q3(a) Explain why there are fundamental ideas of software engineering that apply to all types of software systems. What are the challenges facing software engineering?

Ans: Software engineering is an engineering discipline that is concerned with all aspects of software production from the early stages of system specification through to maintaining the system after it has gone into use. *It is important for two reasons:*

- ✧ More and more, individuals and society rely on advanced software systems. We need to be able to produce reliable and trustworthy systems economically and quickly.
- ✧ It is usually cheaper, in the long run, to use software engineering methods and techniques for software systems rather than just write the programs as if it was a personal programming project. For most types of system, the majority of costs are the costs of changing the software after it has gone into use.

Challenges facing software engineering:

- ✧ Heterogeneity
  - ✧ Increasingly, systems are required to operate as distributed systems across networks that include different types of computer and mobile devices.
- ✧ Business and social change
  - ✧ Business and society are changing incredibly quickly as emerging economies develop and new technologies become available. They need to be able to change their existing software and to rapidly develop new software.
- ✧ Security and trust
  - ✧ As software is intertwined with all aspects of our lives, it is essential that we can trust that software.

Q3(b) : What are the attributes of good software?

Ans: **Attributes of good Software**: Good software should deliver the required functionality and performance to the user and should be maintainable, dependable and usable.

Product characteristic	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilisation, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable and compatible with other systems that they use.

Q4(a) : Explain the Professional & Ethical responsibilities of a software engineer.

- Ans: Ans: Software engineering involves wider responsibilities than simply the application of technical skills. Software engineers must behave in an honest and ethically responsible way if they are to be respected as professionals. Ethical behaviour is more than simply upholding the law but involves following a set of principles that are morally correct.
- Confidentiality
    - Engineers should normally respect the confidentiality of their employers or clients irrespective of whether or not a formal confidentiality agreement has been signed.
- Competence
    - Engineers should not misrepresent their level of competence. They should not knowingly accept work which is outwith their competence.
- Intellectual property rights

  - Engineers should be aware of local laws governing the use of intellectual property such as patents, copyright, etc. They should be careful to ensure that the intellectual property of employers and clients is protected.

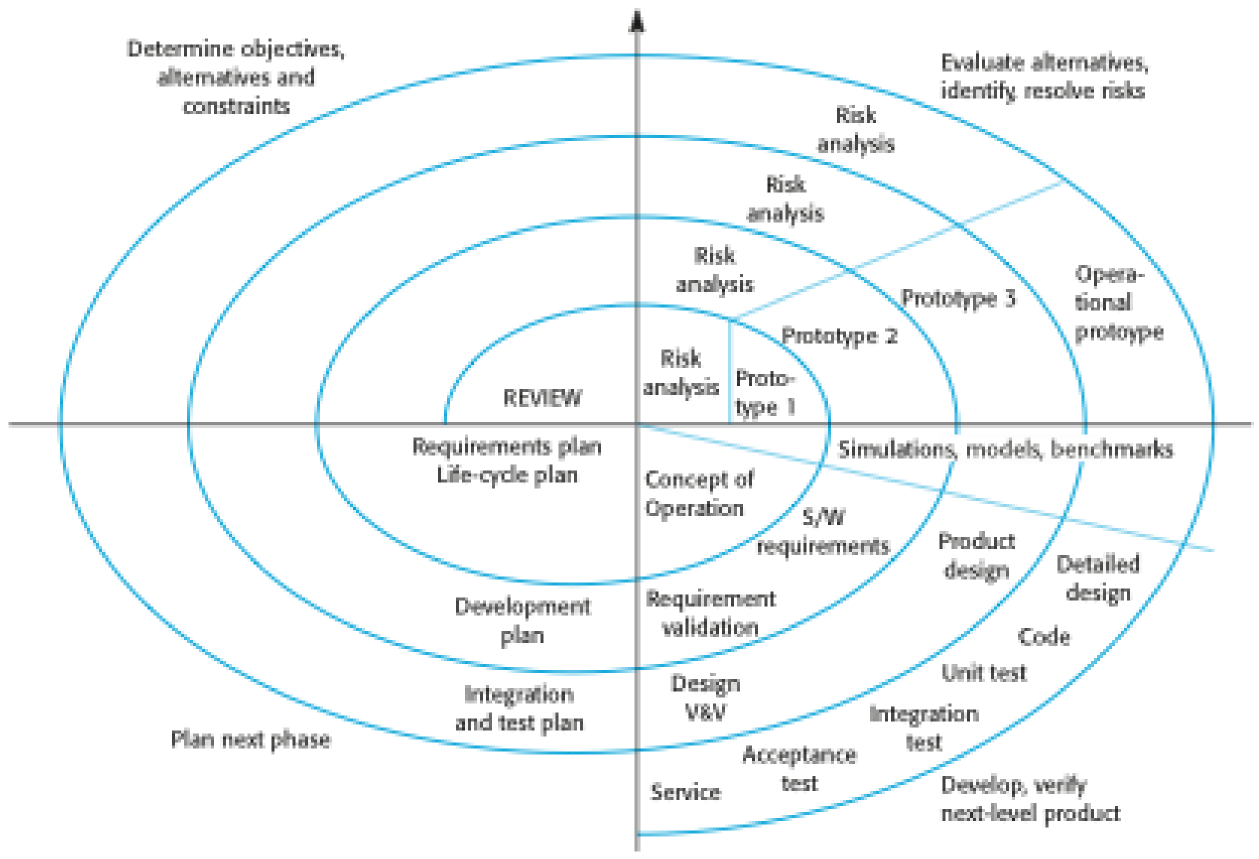
- ✧ Computer misuse
  - Software engineers should not use their technical skills to misuse other people's computers. Computer misuse ranges from relatively trivial (game playing on an employer's machine, say) to extremely serious (dissemination of viruses).

The professional societies in the US have cooperated to produce a code of ethical practice. The Code contains eight Principles related to the behaviour of and decisions made by professional software engineers, including practitioners, educators, managers, supervisors and policy makers, as well as trainees and students of the profession.

- ✧ PUBLIC - Software engineers shall act consistently with the public interest.
- ✧ 2. CLIENT AND EMPLOYER - Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
- ✧ 3. PRODUCT - Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
- ✧ 4. JUDGMENT - Software engineers shall maintain integrity and independence in their professional judgment.
- ✧ 5. MANAGEMENT - Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
- ✧ 6. PROFESSION - Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
- ✧ 7. COLLEAGUES - Software engineers shall be fair to and supportive of their colleagues.
- ✧ 8. SELF - Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Q5(a) : Discuss Boehm's spiral model with a neat diagram.

Ans: **Boehm's spiral model**



- ✧ Process is represented as a spiral rather than as a sequence of activities with backtracking.
- ✧ Each loop in the spiral represents a phase in the process.
- ✧ No fixed phases such as specification or design - loops in the spiral are chosen depending on what is required.
- ✧ Risks are explicitly assessed and resolved throughout the process.

Spiral model sectors:

- ✧ Objective setting
    - ✧ Specific objectives for the phase are identified.
- ✧ Risk assessment and reduction
    - ✧ Risks are assessed and activities put in place to reduce the key risks.
- ✧ Development and validation

    - A development model for the system is chosen which can be any of the generic models.
- Planning
    - The project is reviewed and the next phase of the spiral is planned.

Spiral model usage

- Spiral model has been very influential in helping people think about iteration in software processes and introducing the risk-driven approach to development.
- In practice, however, the model is rarely used as published for practical software development.

Q6(a): Explain types of Requirement.

- Ans: User requirements
    - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- System requirements
    - A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.

  Requirement can be categorised in following way also:

- Functional requirements
    - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
    - May state what the system should not do.
- Non-functional requirements
    - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
    - Often apply to the system as a whole rather than individual features or services.
- Domain requirements
    - Constraints on the system from the domain of operation

Q6(b) : Explain the IEEE structure of SRS document.

Ans:

Chapter	**Description**
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.
Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.

User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

Q7(a) : Briefly explain types of effort estimation approaches & project scheduling and staffing.

Ans: **Types of effort estimation approaches**

- Top- Down Estimation Approach
- Bottom-Up Estimation Approach

Top- Down Estimation Approach

a. Consider effort as a function of project size.
b. First determine the nature of the function.
c. Then estimate the size of the function.

Past productivity on similar projects can be used as the estimation function.If productivity is P KLOC/PM, then

effort estimate = SIZE/P person-months

More general function

$$EFFORT= a*SIZE^{b}$$

Where a and b are constant and determined through regression analysis.

Bottom-Up Estimation Approach

a. Project is first divided into tasks
b. And then estimates for the different tasks of the project are obtained.
c. From the estimates of the different tasks, the overall estimate is determined.
d. This type of approach is also called activity-based estimation.

Project scheduling and staffing

For a project with some estimated effort, multiple schedules (or project duration) are indeed possible. Once the effort is fixed, there is some flexibility in setting the schedule by appropriately staffing the project, but this flexibility is not unlimited. The overall schedule can be determined as a function of effort. Such function can be determined from data from completed projects using statistical techniques like fitting a regression curve

M, in calendar months can also be estimated by $M = 4.1E^{.36}$.

In COCOMO, the equation for schedule for an organic type of software is $M = 2.5E^{.38}$

Another method for medium-sized projects is the rule of thumb called the square root check .

The proposed schedule can be around the square root of the total effort in person months.

For example, if the effort estimate is 50 person-months, a schedule of about 7 to 8 months will be suitable.

Q8(a) : Explain different types of architectural styles for component and connector view.

Ans: Architecture Styles for C&C View

Pipe and Filter

Pipe-and-filter style of architecture is well suited for systems that primarily do data transformation whereby some input data is received and the goal of the system is to produce some output data by suitably transforming the input data. The pipe-and-filter style has only one component type called the filter. It also has only one connector type, called the pipe. A filter performs a data transformation, and sends the transformed data to other filters for further processing using the pipe connector.

Shared-Data Style

In this style, there are two types of components—data repositories and data accessors. Components of data repository type are where the system stores shared data—these could be file systems or databases. These components provide a reliable and permanent storage, take care of any synchronization needs for concurrent access, and provide data access support. Components of data accessors type access data from the repositories, perform computation on the data obtained, and if they want to share the results with other components, put the results back in the depository.
There are two variations of this style possible. In the blackboard style, if some data is posted on the data repository, all the accessor components that need to know about it are informed
The other is the repository style, in which the data repository is just

a passive repository which provides permanent storage and related controls for data accessing. The components access the repository as and when they want.

Client-Server Style

In this style, there are two component types—clients and servers. A constraint of this style is that a client can only communicate with the server, and cannot communicate with other clients. The communication between a client component and a server component is initiated by the client when the client sends a request for some service that the server supports. The server receives the request at its defined port, performs the service, and then returns the results of the computation to the client who requested the service.
There is one connector type in this style—the request/reply type. A connector connects a client to a server.

Some Other Styles

Publish-Subscribe Style
In this style, there are two types of components. One type of component subscribes to a set of defined events. Other types of components generate or publish events. In response to these events, the components that have published their intent to process the event, are invoked. This type of style is most natural in user interface frameworks, where many events are defined (like mouse click) and components are assigned to these events. When that event occurs, the associated component is executed.

Peer-to-peer style, or object-oriented style If we take a client-server style, and generalize each component to be a client as well as a server, then we have this style. In this style, components are peers and any component can request a service from any other component.

Communicating processes style
The components in this model are processes or threads, which communicate with each other either with message passing or through shared memory. This style is used in some form in many complex systems which use multiple threads or processes.