Software Engineering & Project Management (SEPM)

Module-Wise Descriptive Answers (Part 1)

Prepared from the uploaded SEPM Question Bank.

MODULE 1 – Introduction to Software Engineering & Process Models

---

Q1. Define Software Engineering and explain different Umbrella Activities.

Definition of Software Engineering

Software Engineering is the systematic, disciplined, and quantifiable approach for the development, operation, maintenance, and management of software systems.

According to IEEE:

> "Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software."

Software engineering helps in developing reliable, efficient, maintainable, and cost-effective software within a specified time.

Need for Software Engineering

Software Engineering is required because:

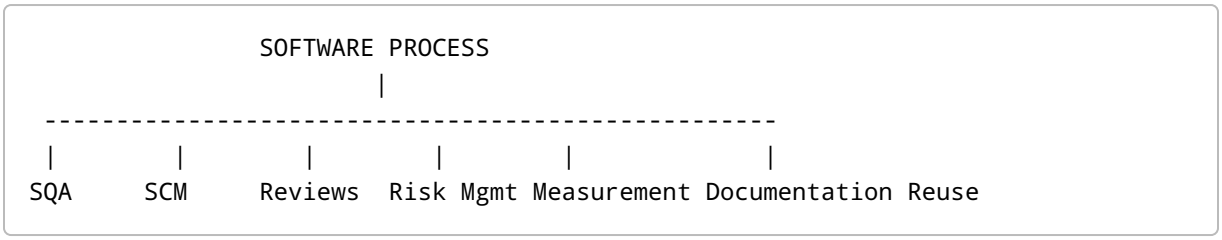
1. Software systems are becoming increasingly complex.
2. Large software projects require proper planning and management.
3. It improves software quality and reliability.
4. It reduces development cost and time.
5. It ensures maintainability and scalability.
6. It helps manage risks effectively.

Umbrella Activities in Software Engineering

Umbrella activities are activities that are carried out throughout the software process irrespective of the development model used.

These activities support and improve the software development process.

Diagram



---

1. Software Project Tracking and Control

This activity ensures that the project is progressing according to the plan.

Functions:

- Monitor schedule and cost.
- Compare actual progress with planned progress.
- Take corrective actions when deviations occur.

Example:

If coding is delayed by 1 week, additional resources may be allocated.

---

2. Risk Management

Risk management identifies possible problems before they occur.

Steps:

1. Risk Identification
2. Risk Analysis
3. Risk Prioritization
4. Risk Mitigation
5. Risk Monitoring

Example Risks:

- Requirement changes
- Budget issues
- Technical failures

---

3. Software Quality Assurance (SQA)

SQA ensures that software quality standards are maintained.

Activities:

- Audits
- Reviews
- Testing
- Verification and Validation

Benefits:

- Reduced defects
- Improved reliability
- Better customer satisfaction

---

4. Formal Technical Reviews (FTR)

FTR is a review process conducted to detect errors before testing.

Objectives:

- Detect defects
- Ensure standards compliance
- Improve software quality

Participants:

- Review leader
- Author
- Reviewers
- Recorder

---

5. Measurement (Metrics)

Metrics are used to measure software quality and productivity.

Examples:

- Lines of Code (LOC)
- Function Points
- Defect Density
- Productivity Metrics

Importance:

- Helps estimate cost and effort
- Improves process quality

---

6. Software Configuration Management (SCM)

SCM manages changes made to software.

Activities:

- Version control
- Change management
- Baseline management
- Release management

Example:

GitHub is commonly used for SCM.

---

7. Reusability Management

It focuses on reusing existing software components.

Advantages:

- Faster development
- Reduced cost
- Improved reliability

Example:

Using reusable libraries or APIs.

---

8. Work Product Preparation and Production

This activity includes preparation of documents such as:

- SRS
- Design documents
- Test cases
- User manuals

Proper documentation improves communication and maintenance.

Conclusion

Umbrella activities support the entire software development lifecycle. They help improve software quality, manage risks, maintain documentation, and ensure smooth project execution.

Q2. Explain Agile Process Model. Compare Agile and Traditional/Waterfall Models.

Agile Process Model

Agile is an iterative and incremental software development model where software is developed in small cycles called iterations.

The Agile model focuses on:

- Customer collaboration
- Continuous feedback
- Rapid delivery
- Flexibility
- Team communication

Agile Manifesto Principles

1. Customer satisfaction through continuous delivery.
2. Welcome changing requirements.
3. Deliver working software frequently.
4. Developers and customers must work together.

5. Working software is the main measure of progress.

Agile Development Cycle

```
Requirement → Planning → Design → Coding → Testing → Release → Feedback
                                  ↑_____________________________________|
```

Features of Agile Model

1. Incremental Development
2. Customer involvement
3. Small releases
4. Adaptive planning
5. Continuous testing
6. Faster delivery

Advantages of Agile Model

1. Faster delivery of software.
2. Easy to handle requirement changes.
3. Better customer satisfaction.
4. Reduced development risk.
5. Continuous feedback improves quality.

Disadvantages of Agile Model

1. Difficult for large projects.
2. Requires experienced developers.
3. Documentation may be limited.
4. Frequent changes may increase scope.

Applications of Agile

- Web applications
- Mobile applications

- Startups
- AI-based applications
- Cloud software

Comparison Between Agile and Waterfall Model

Parameter	Agile Model	Waterfall Model
Development Style	Iterative	Sequential
Requirement Changes	Easily accepted	Difficult
Customer Involvement	High	Low
Testing	Continuous	After development
Delivery	Small increments	Final product at end
Flexibility	High	Low
Documentation	Less	Extensive
Risk Handling	Better	Difficult
Suitable For	Dynamic projects	Stable projects
Feedback	Continuous	Limited

Waterfall Model

Waterfall model is a linear sequential software development model.

Phases of Waterfall Model

```
Requirement Analysis
        ↓
System Design
        ↓
Implementation
        ↓
Testing
        ↓
Deployment
```

```
        ↓
Maintenance
```

Advantages of Waterfall Model

1. Simple and easy to understand.
2. Proper documentation.
3. Easy management.
4. Suitable for fixed requirements.

Disadvantages of Waterfall Model

1. Difficult to change requirements.
2. Late testing.
3. Higher risk.
4. Customer feedback comes very late.

Conclusion

Agile is more flexible and customer-oriented compared to Waterfall. Agile is suitable for modern dynamic software projects where requirements change frequently.

Q3. Describe Waterfall Model and Incremental Process Model.

Waterfall Model

The Waterfall model is the earliest software process model where development flows sequentially through various phases.

Phases

1. Requirement Analysis
2. System Design
3. Coding

4. Testing
5. Deployment
6. Maintenance

Diagram

```
Requirements
     ↓
Design
     ↓
Coding
     ↓
Testing
     ↓
Deployment
     ↓
Maintenance
```

Advantages

- Simple structure
- Proper documentation
- Easy to manage
- Suitable for small projects

Disadvantages

- Difficult to change requirements
- Testing occurs late
- High risk for complex projects

Incremental Process Model

In Incremental model, software is developed and delivered in multiple increments.

Each increment adds new functionality.

Incremental Model Diagram

```
Increment 1 → Basic Features
Increment 2 → Additional Features
Increment 3 → Final Features
```

Advantages

1. Early delivery of software.
2. Reduced risk.
3. Easy testing and debugging.
4. Customer feedback after each release.

Disadvantages

1. Requires proper planning.
2. Integration issues may arise.
3. Architecture must be strong.

Comparison Between Waterfall and Incremental Model

Waterfall	Incremental
Sequential approach	Iterative approach
Single delivery	Multiple deliveries
Requirement changes difficult	Requirement changes easier
High risk	Lower risk
Testing late	Testing after each increment

Conclusion

Incremental model is more flexible and suitable for modern software systems compared to the traditional Waterfall model.

Q4. Describe Waterfall Model and Evolutionary Process Model.

Evolutionary Process Model

Evolutionary model develops software gradually through repeated iterations.

The system evolves according to user feedback.

Types of Evolutionary Models

1. Prototyping Model
2. Spiral Model
3. Concurrent Development Model

Prototyping Model

A prototype is an early working model of software.

Steps:

1. Requirement gathering
2. Quick design
3. Prototype development
4. Customer evaluation
5. Refinement
6. Final product

Spiral Model

Spiral model combines iterative development with risk analysis.

Phases:

1. Planning

2. Risk Analysis
3. Engineering
4. Evaluation

Advantages of Evolutionary Model

1. Better customer involvement.
2. Reduced risk.
3. Flexible development.
4. Early software visibility.

Disadvantages

1. Expensive.
2. Complex management.
3. Requires expert developers.

Conclusion

Evolutionary models are suitable for large and complex projects where requirements are not clearly defined.

Q5. What are Agile Methodologies? Explain any one.

Agile Methodologies

Agile methodologies are software development approaches based on iterative development and customer collaboration.

Types of Agile Methodologies

1. Scrum
2. Kanban
3. Extreme Programming (XP)

4. Crystal Method
5. Lean Development
6. Feature Driven Development (FDD)

Scrum Methodology

Scrum is the most popular Agile methodology.

It divides work into fixed time periods called sprints.

Important Terms in Scrum

1. Sprint

A sprint is a short development cycle usually lasting 2–4 weeks.

2. Product Backlog

List of all project requirements.

3. Sprint Backlog

Tasks selected for current sprint.

4. Scrum Master

Person who manages Scrum process.

5. Product Owner

Represents customer requirements.

6. Development Team

Develops the software.

Scrum Workflow

```
Product Backlog → Sprint Planning → Sprint → Testing → Review → Next Sprint
```

Advantages of Scrum

1. Faster delivery.
2. Better teamwork.
3. High customer satisfaction.
4. Flexible development.

Disadvantages

1. Requires experienced team.
2. Difficult for large teams.
3. Scope creep possible.

Conclusion

Scrum improves collaboration, flexibility, and software quality through iterative development.

Q6. Short Note on Scrum Model

Scrum is an Agile framework used for iterative software development.

Features of Scrum

• Incremental development
• Time-boxed sprints
• Daily Scrum meetings
• Continuous customer feedback
• Rapid delivery

Scrum Roles

Role	Responsibility
Product Owner	Manages requirements
Scrum Master	Facilitates Scrum
Development Team	Builds software

Scrum Artifacts

1. Product Backlog
2. Sprint Backlog
3. Increment

Scrum Events

1. Sprint Planning
2. Daily Scrum
3. Sprint Review
4. Sprint Retrospective

Advantages

- Improved productivity
- Better communication
- Faster delivery

Q7. Short Note on Kanban Model

Kanban is an Agile methodology focused on workflow visualization and continuous delivery.

Kanban Board

```
To Do → In Progress → Testing → Done
```

Features of Kanban

1. Visual workflow.
2. Continuous delivery.
3. Work-in-progress limits.
4. Flexible task handling.

Advantages

- Improved efficiency
- Reduced bottlenecks
- Better workflow visibility

Q8. Short Note on Extreme Programming (XP)

Extreme Programming is an Agile methodology focused on improving software quality.

Features of XP

1. Pair Programming
2. Continuous Integration
3. Test Driven Development
4. Small Releases
5. Customer Involvement

Advantages

- High software quality
- Fewer defects
- Better teamwork

Q9. Short Note on Software Design Patterns

Software Design Patterns are reusable solutions to common software design problems.

Types of Design Patterns

1. Creational Patterns

Used for object creation.

Example:

- Singleton
- Factory

2. Structural Patterns

Used for class and object composition.

Example:

- Adapter
- Bridge

3. Behavioral Patterns

Used for communication between objects.

Example:

- Observer
- Strategy

Advantages

1. Reusable solutions.
2. Reduced development time.
3. Better maintainability.
4. Improved software architecture.

Conclusion

Design patterns help developers build scalable and maintainable software systems.

MODULE 2 – Requirements Analysis & Cost Estimation

Q1. Elaborate the COCOMO Model for Cost Estimation.

Introduction

COCOMO (Constructive Cost Model) is a software cost estimation model proposed by Barry Boehm.

It estimates:

- Development effort
- Project cost
- Development time

based on the size of software.

Software size is measured in KLOC (Kilo Lines of Code).

Types of COCOMO Models

1. Basic COCOMO
2. Intermediate COCOMO
3. Detailed COCOMO

Software Project Categories

Category	Description
Organic	Small and simple projects
Semi-Detached	Medium complexity projects
Embedded	Complex and hardware-dependent projects

Basic COCOMO Formula

Effort Equation

```
Effort = a × (KLOC)^b
```

Development Time Equation

```
Time = c × (Effort)^d
```

Where:

- Effort = Person-Months
- Time = Months
- KLOC = Thousand Lines of Code

Constants for Basic COCOMO

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

Example Problem

Assume:

- Project Size = 10 KLOC
- Organic Mode

Step 1: Effort Calculation

```
Effort = 2.4 × (10)^1.05
       ≈ 26.9 Person-Months
```

Step 2: Development Time

```
Time = 2.5 × (26.9)^0.38
     ≈ 8.7 Months
```

Advantages of COCOMO

1. Easy estimation.
2. Accurate for traditional projects.
3. Helps in project planning.
4. Useful for budgeting.

Disadvantages

1. Depends on LOC estimation.
2. Less suitable for Agile projects.
3. Accuracy depends on input data.

Conclusion

COCOMO is one of the most widely used software estimation models for calculating project effort, time, and cost.

Q2. Explain Functional and Non-Functional Requirements.

Requirements

Requirements describe what a software system should do and how it should perform.

Requirements are classified into: 1. Functional Requirements 2. Non-Functional Requirements

---

Functional Requirements

Functional requirements specify the functions performed by the system.

These describe system behavior.

Examples

- User login
- Payment processing
- Report generation
- Search functionality

Characteristics

1. Describe system operations.
2. Define inputs and outputs.
3. Describe interactions.

Non-Functional Requirements

Non-functional requirements specify quality attributes of the system.

They describe how the system performs.

Examples

- Security
- Performance
- Reliability
- Scalability
- Maintainability
- Portability

Types of Non-Functional Requirements

Type	Description
Performance	Response time and speed
Security	Protection from unauthorized access
Reliability	Continuous operation
Usability	Ease of use
Scalability	Handling increased load

Difference Between Functional and Non-Functional Requirements

Functional	Non-Functional
Describes what system does	Describes how system works
Feature-oriented	Quality-oriented
Mandatory functions	Performance constraints
Example: Login	Example: Fast login response

Conclusion

Both functional and non-functional requirements are essential for developing reliable and high-quality software systems.
