

# Subject Name Solutions

4331604 – Summer 2025

Semester 1 Study Material

*Detailed Solutions and Explanations*

## Question 1(a) [3 marks]

Give IEEE definition of software. Write one example of each for application and system software.

### Solution

**IEEE Definition:** Software is a collection of computer programs, procedures, rules, and associated documentation and data.

**Examples:**

Software Type	Example	Purpose
Application Software	Microsoft Word	Word processing and document creation
System Software	Windows 10	Operating system managing hardware resources

- **Application software:** Programs designed for end-users to accomplish specific tasks
- **System software:** Programs that manage and operate computer hardware

### Mnemonic

“Apps help Users, Systems help Hardware”

## Question 1(b) [4 marks]

Write a short note on data dictionary.

### Solution

Data dictionary is a centralized repository containing definitions and characteristics of data elements used in a system.

**Components Table:**

Component	Description
<b>Data Name</b>	Unique identifier for data element
<b>Aliases</b>	Alternative names used
<b>Description</b>	Purpose and meaning
<b>Data Type</b>	Format (integer, string, etc.)
<b>Length</b>	Size constraints
<b>Values</b>	Valid range or set

- **Purpose:** Ensures consistency in data usage across development team
- **Benefits:** Reduces ambiguity, improves communication, standardizes data definitions
- **Usage:** Referenced during system design and database creation

### Mnemonic

“Dictionary Defines Data Clearly”

### Question 1(c) [7 marks]

Explain prototype model with figure.

#### Solution

Prototype model is an iterative approach where a working model is built early to understand requirements better.

**Diagram:**

#### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Requirement Gathering] --> B[Quick Design]
    B --> C[Build Prototype]
    C --> D[User Evaluation]
    D --> E{User Satisfied?}
    E -- No --> F[Refine Requirements]
    F --> B
    E -- Yes --> G[Final System Development]
    G --> H[Testing & Maintenance]
{Highlighting}
{Shaded}
```

**Characteristics:**

Phase	Activity	Output
<b>Quick Design</b>	Basic architecture	Initial design
<b>Prototype Build</b>	Working model	Testable system
<b>User Evaluation</b>	Feedback collection	Requirements refinement

- **Advantages:** Early user feedback, reduced development risk, better requirement understanding
- **Disadvantages:** May lead to inadequate analysis, customer expects prototype as final product
- **Best for:** Projects with unclear requirements

#### Mnemonic

“Prototype Proves Possibilities”

### Question 1(c) OR [7 marks]

Explain RAD model with advantages and disadvantages.

#### Solution

RAD (Rapid Application Development) emphasizes quick development through prototyping and iterative development.

**RAD Phases:**

#### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Business Modeling] --> B[Data Modeling]
    B --> C[Process Modeling]
    C --> D[Application Generation]
    D --> E[Testing & Turnover]
{Highlighting}
```

{Shaded}

#### Advantages vs Disadvantages:

Advantages	Disadvantages
Faster development	Requires skilled developers
Early user involvement	Not suitable for large projects
Reduced costs	Requires user commitment
Better quality	Technical risks if not managed

- **Key feature:** Uses automated tools and 4GL programming
- **Timeline:** Typically 60-90 days for development
- **Team:** Small, experienced development teams

#### Mnemonic

“RAD Rapidly Accelerates Development”

### Question 2(a) [3 marks]

Give the full form of following: SQA, FTR, RAD, BVA, GUI, DFD

#### Solution

Abbreviation	Full Form
<b>SQA</b>	Software Quality Assurance
<b>FTR</b>	Formal Technical Review
<b>RAD</b>	Rapid Application Development
<b>BVA</b>	Boundary Value Analysis
<b>GUI</b>	Graphical User Interface
<b>DFD</b>	Data Flow Diagram

#### Mnemonic

“Software Quality And Formal Technical Reviews Rapidly Analyze Development, Boundary Value Analysis Guides User Interface, Data Flow Diagrams”

### Question 2(b) [4 marks]

Define agile methodology. Discuss agile principles.

#### Solution

**Definition:** Agile is an iterative software development approach emphasizing collaboration, flexibility, and rapid delivery of working software.

#### Core Agile Principles:

Principle	Description
<b>Individuals over processes</b>	People and communication are priority
<b>Working software over documentation</b>	Functional software is primary measure
<b>Customer collaboration</b>	Continuous customer involvement
<b>Responding to change</b>	Adaptability over rigid plans

- **Iteration length:** Typically 2-4 weeks (sprints)
- **Delivery:** Frequent working software releases
- **Team structure:** Cross-functional, self-organizing teams

#### Mnemonic

“Agile Adapts And Advances”

### Question 2(c) [7 marks]

Explain XP model with its advantages and disadvantages.

#### Solution

XP (Extreme Programming) is an agile methodology emphasizing engineering practices and customer satisfaction.

##### XP Practices:

mindmap

```

root((XP Practices))
  Planning Game
  Small Releases
  Pair Programming
  Test{-Driven Development}
  Continuous Integration
  Refactoring
  Simple Design
  Collective Code Ownership
  
```

##### Advantages and Disadvantages:

Advantages	Disadvantages
High code quality	Requires experienced programmers
Rapid feedback	Customer must be available
Reduced bugs	Code-focused, less documentation
Flexibility	Difficult to estimate costs

- **Key practice:** Pair programming ensures code quality
- **Testing:** Test-first approach with automated testing
- **Customer role:** On-site customer provides continuous feedback

#### Mnemonic

“eXtreme Programming eXcels through Practices”

### Question 2(a) OR [3 marks]

Define black box testing. Give at least two names of black box testing method.

#### Solution

**Definition:** Black box testing examines software functionality without knowledge of internal code structure, focusing on input-output behavior.

##### Black Box Testing Methods:

Method	Description
<b>Equivalence Partitioning</b>	Divides input into valid/invalid classes
<b>Boundary Value Analysis</b>	Tests values at input boundaries

- **Approach:** Tests based on requirements and specifications
- **Tester knowledge:** No internal code knowledge required
- **Focus:** External behavior and functionality

#### Mnemonic

“Black Box Behavior Based”

### Question 2(b) OR [4 marks]

Give the full form of CLI. Explain CLI in brief.

#### Solution

**CLI:** Command Line Interface  
**CLI Characteristics:**

Aspect	Description
<b>Input method</b>	Text commands typed by user
<b>Output</b>	Text-based responses
<b>Navigation</b>	Commands for file/directory operations
<b>Efficiency</b>	Faster for experienced users

- **Advantages:** Fast execution, less memory usage, scriptable
- **Disadvantages:** Requires learning commands, not user-friendly for beginners
- **Examples:** Windows Command Prompt, Linux Terminal, DOS

#### Mnemonic

“Commands Lead Interaction”

### Question 2(c) OR [7 marks]

Explain waterfall model with neat figure.

#### Solution

Waterfall model is a linear sequential approach where each phase must be completed before moving to the next.

**Waterfall Model Diagram:**

#### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Requirement Analysis] --> B[System Design]
    B --> C[Implementation]
    C --> D[Integration & Testing]
    D --> E[Deployment]
    E --> F[Maintenance]
```

```

style A fill:\#e1f5fe
style B fill:\#f3e5f5
style C fill:\#fff3e0
style D fill:\#f1f8e9
style E fill:\#fce4ec
style F fill:\#fff8e1
{Highlighting}
{Shaded}

```

#### Phase Details:

Phase	Activities	Deliverables
<b>Requirements</b>	Gather and document needs	SRS document
<b>Design</b>	System architecture	Design documents
<b>Implementation</b>	Code development	Source code
<b>Testing</b>	Verify functionality	Test reports
<b>Deployment</b>	System installation	Working system
<b>Maintenance</b>	Bug fixes, updates	Updated system

- **Advantages:** Simple, easy to manage, well-documented
- **Disadvantages:** Inflexible, late testing, difficult to accommodate changes

#### Mnemonic

“Water Always Flows Downward”

### Question 3(a) [3 marks]

Give one word answer:

#### Solution

Question	Answer
<b>Lowest cohesion is</b>	Coincidental
<b>Highest coupling is</b>	Content
<b>Slack time of critical activity is</b>	Zero

#### Mnemonic

“Coincidental Cohesion, Content Coupling, Critical Zero”

### Question 3(b) [4 marks]

Explain classification of coupling.

#### Solution

Coupling measures interdependence between modules. Lower coupling is better for maintainability.

**Coupling Types (Best to Worst):**

Type	Description	Example
<b>Data</b>	Parameters passed	Method calls with parameters
<b>Stamp</b>	Data structure passed	Passing objects/records

<b>Control</b>	Control information passed	Flags/switches passed
<b>External</b>	External data reference	Global variables
<b>Common</b>	Shared data area	Common memory blocks
<b>Content</b>	Direct access to internals	Modifying another module's data

- **Best practice:** Aim for data coupling
- **Avoid:** Content and common coupling
- **Design goal:** Minimize dependencies between modules

#### Mnemonic

“Data Stamps Control External Common Content”

### Question 3(c) [7 marks]

Define following terms (don't just give the full form):

#### Solution

Term	Definition
<b>UI</b>	User Interface - the means by which users interact with software systems
<b>SE</b>	Software Engineering - systematic approach to software development using engineering principles
<b>PMC</b>	Project Management and Control - planning, monitoring, and controlling software projects
<b>SDLC</b>	Software Development Life Cycle - phases involved in software development from conception to maintenance
<b>Verification</b>	Process of checking if software meets specified requirements and design
<b>Validation</b>	Process of checking if software meets user needs and intended purpose
<b>SRS</b>	Software Requirements Specification - detailed document describing software functionality and constraints

- **Verification:** “Are we building the product right?”
- **Validation:** “Are we building the right product?”
- **Key difference:** Verification checks specifications, Validation checks user satisfaction

#### Mnemonic

“Users Interact, Software Engineers Plan, Managing Cycles, Specifications Define, Verification checks Requirements, Validation checks Satisfaction, Requirements Specify Software”

### Question 3(a) OR [3 marks]

Explain menu based UI with advantages and disadvantages.

#### Solution

Menu-based UI presents options in hierarchical menus for user selection.

**Advantages vs Disadvantages:**

Advantages	Disadvantages
<b>Easy to learn</b>	<b>Slower for experts</b>

**Reduces errors**      **Limited flexibility**  
**Self-explanatory**   **Screen space consumption**

- **Structure:** Hierarchical organization of options
- **Navigation:** Point-and-click or keyboard shortcuts
- **Best for:** Applications with well-defined functions

#### Mnemonic

“Menus Make Choices Clear”

### Question 3(b) OR [4 marks]

Explain classification of cohesion.

#### Solution

Cohesion measures how closely related elements within a module are. Higher cohesion is better.

**Cohesion Types (Best to Worst):**

Type	Description
<b>Functional</b>	Single, well-defined task
<b>Sequential</b>	Output of one element feeds next
<b>Communicational</b>	Elements work on same data
<b>Procedural</b>	Elements follow execution sequence
<b>Temporal</b>	Elements executed at same time
<b>Logical</b>	Elements perform similar functions
<b>Coincidental</b>	Elements randomly grouped

- **Goal:** Achieve functional cohesion
- **Design principle:** Each module should have single responsibility
- **Measurement:** Higher cohesion = better design

#### Mnemonic

“Functional Sequences Communicate Procedures Temporally through Logical Coincidence”

### Question 3(c) OR [7 marks]

Define risk. Explain risk management.

#### Solution

**Risk Definition:** Potential problem that may occur during software development, causing negative impact on project success.

**Risk Management Process:**

Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Risk Identification] --> B[Risk Assessment]
    B --> C[Risk Prioritization]
    C --> D[Risk Mitigation]
    D --> E[Risk Monitoring]
```

E {-{-}{ } A}  
{Highlighting}  
{Shaded}

#### Risk Management Activities:

Activity	Description	Output
<b>Identification</b>	Find potential problems	Risk list
<b>Assessment</b>	Analyze probability and impact	Risk analysis
<b>Prioritization</b>	Rank risks by importance	Priority matrix
<b>Mitigation</b>	Plan risk responses	Mitigation strategies
<b>Monitoring</b>	Track risk status	Updated risk status

- **Risk types:** Technical, Project, Business risks
- **Strategies:** Avoid, Transfer, Mitigate, Accept
- **Tools:** Risk matrices, probability-impact charts

#### Mnemonic

“Risk Requires Careful Planning”

### Question 4(a) [3 marks]

Define: Error, Failure, Test case

#### Solution

Term	Definition
<b>Error</b>	Human mistake made during software development process
<b>Failure</b>	Deviation of software behavior from expected results
<b>Test case</b>	Set of conditions to verify specific functionality or system requirement

- **Relationship:** Error leads to defect, defect causes failure
- **Error source:** Developer mistakes, misunderstanding requirements
- **Test case components:** Input, expected output, execution steps

#### Mnemonic

“Errors Cause Failures, Tests Catch Problems”

### Question 4(b) [4 marks]

Identify any six functional requirements of ATM system.

#### Solution

##### ATM System Functional Requirements:

Requirement	Description
<b>User Authentication</b>	PIN verification for account access
<b>Balance Inquiry</b>	Display current account balance
<b>Cash Withdrawal</b>	Dispense requested cash amount
<b>Fund Transfer</b>	Transfer money between accounts

**Transaction History** Show recent transaction records  
**PIN Change** Allow users to modify PIN

- **Security:** All transactions require authentication
- **Validation:** Check sufficient balance before withdrawal
- **Logging:** Record all transactions for audit

#### Mnemonic

“ATMs Authenticate, Balance, Cash, Transfer, History, PIN”

### Question 4(c) [7 marks]

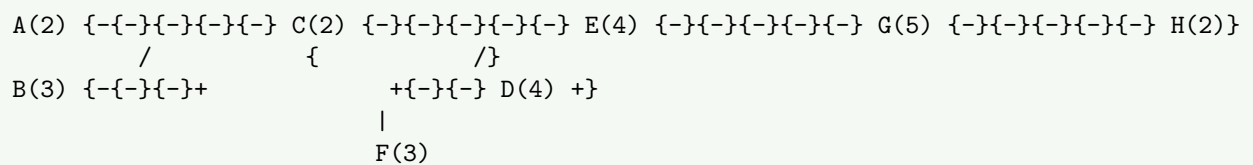
State the use of activity network diagram. Develop activity network diagram for the following system and find the critical path for the same.

#### Solution

**Activity Network Diagram Uses:**

- **Project scheduling:** Determine project timeline
- **Critical path identification:** Find longest path determining minimum project duration
- **Resource planning:** Optimize resource allocation

**Activity Network Diagram:**



**Critical Path Analysis:**

Path	Activities	Duration	Critical?
A-C-E-G-H	A	2+2+4+5+2 = 15	No
B-C-E-G-H	B	3+2+4+5+2 = 16	<b>Yes</b>
A-C-D-G-H	A	2+2+4+5+2 = 15	No

**Critical Path:** B(16 days) **Project Duration:** 16 days

#### Mnemonic

“Networks Navigate Project Paths”

### Question 4(a) OR [3 marks]

Explain any three requirement gathering activities.

#### Solution

**Requirement Gathering Activities:**

Activity	Description	Output
<b>Stakeholder Interviews</b>	Direct discussion with users and clients	Interview notes, requirements list

<b>Questionnaires</b>	Structured questions for large user groups	Survey responses, statistical data
<b>Document Analysis</b>	Review existing system documentation	Current system understanding
<ul style="list-style-type: none"> <li>• <b>Purpose:</b> Understand user needs and system expectations</li> <li>• <b>Participants:</b> Users, customers, domain experts, developers</li> <li>• <b>Documentation:</b> All findings recorded in SRS document</li> </ul>		

#### Mnemonic

“Interviews, Questions, Documents Gather Requirements”

### Question 4(b) OR [4 marks]

Develop use case diagram for Bank ATM system.

#### Solution

**ATM Use Case Diagram:**

```
graph TB
    Customer((Customer))
    Admin((Admin))
    Bank[Bank System]

    Customer -- UC1[Check Balance]
    Customer -- UC2[Withdraw Cash]
    Customer -- UC3[Transfer Funds]
    Customer -- UC4[Change PIN]
    Customer -- UC5[Print Receipt]

    Admin -- UC6[Load Cash]
    Admin -- UC7[View Logs]
    Admin -- UC8[Maintenance]

    UC1 -. Bank
    UC2 -. Bank
    UC3 -. Bank
    UC4 -. Bank
```

**Use Case Details:**

Actor	Use Cases
<b>Customer</b>	Check Balance, Withdraw Cash, Transfer Funds, Change PIN
<b>Admin</b>	Load Cash, View Logs, System Maintenance
<b>Bank System</b>	Validate accounts, Process transactions

#### Mnemonic

“Customers Use ATMs, Admins Maintain Systems”

### Question 4(c) OR [7 marks]

Draw the figure of spiral model. Explain it in brief.

## Solution

### Spiral Model Diagram:

```
graph TB
    subgraph "Spiral Model"
        A[Planning] --> B[Risk Analysis]
        B --> C[Engineering]
        C --> D[Customer Evaluation]
        D --> A

        A1[Plan 1] --> B1[Risk 1]
        B1 --> C1[Code 1]
        C1 --> D1[Test 1]
        D1 --> A2[Plan 2]
        A2 --> B2[Risk 2]
        B2 --> C2[Code 2]
        C2 --> D2[Test 2]
    end
```

### Spiral Model Characteristics:

Quadrant	Activity	Purpose
<b>Planning</b>	Define objectives, alternatives	Set goals for iteration
<b>Risk Analysis</b>	Identify and resolve risks	Minimize project risks
<b>Engineering</b>	Develop and test product	Create working software
<b>Evaluation</b>	Customer assessment	Get user feedback

- **Key feature:** Risk-driven approach with iterative development
- **Best for:** Large, complex, high-risk projects
- **Advantages:** Risk management, flexible, incremental development
- **Disadvantages:** Complex management, expensive, requires risk expertise

## Mnemonic

“Spirals Plan, Risk, Engineer, Evaluate”

## Question 5(a) [3 marks]

State TRUE or FALSE for the following.

## Solution

Statement	Answer	Explanation
Activity network diagram used to determine critical path	<b>TRUE</b>	Primary purpose of activity networks
In CPM, the shortest path is the critical path	<b>FALSE</b>	Longest path is critical path
Risk avoidance is the best technique to solve risks	<b>FALSE</b>	Best technique depends on risk type

- **Critical path:** Longest duration path in project network
- **CPM:** Critical Path Method identifies project bottlenecks
- **Risk strategies:** Avoid, Transfer, Mitigate, Accept (choice depends on context)

### Mnemonic

“True Networks, False Shortest, False Best”

### Question 5(b) [4 marks]

Identify the differences between traditional model approach and agile approach. (at least 4 differences)

#### Solution

##### Traditional vs Agile Comparison:

Aspect	Traditional	Agile
<b>Planning</b>	Extensive upfront planning	Adaptive planning
<b>Documentation</b>	Heavy documentation	Minimal documentation
<b>Customer involvement</b>	Limited to requirements phase	Continuous involvement
<b>Change handling</b>	Difficult and expensive	Embraces change
<b>Delivery</b>	Single final delivery	Frequent incremental delivery
<b>Process</b>	Process-driven	People-driven

- **Traditional:** Predictive, sequential approach
- **Agile:** Adaptive, iterative approach
- **Flexibility:** Agile more responsive to changing requirements

### Mnemonic

“Traditional Plans Heavy, Agile Adapts Light”

### Question 5(c) [7 marks]

Define unit testing. Draw the figure of it. Explain the process of unit testing.

#### Solution

**Unit Testing Definition:** Testing individual software components or modules in isolation to verify they function correctly according to design specifications.

**Unit Testing Process:**

##### Mermaid Diagram (Code)

```
{Shaded}
{Highlighting}[]
graph LR
    A[Select Unit] --> B[Design Test Cases]
    B --> C[Set Up Test Environment]
    C --> D[Execute Tests]
    D --> E[Record Results]
    E --> F{All Tests Pass?}
    F -- No --> G[Debug and Fix]
    G --> D
    F -- Yes --> H[Unit Approved]
{Highlighting}
{Shaded}
```

**Unit Testing Process Steps:**

Step	Activity	Purpose
<b>Test Planning</b>	Identify units to test	Define testing scope
<b>Test Design</b>	Create test cases	Cover all code paths
<b>Test Setup</b>	Prepare test environment	Isolate unit under test
<b>Test Execution</b>	Run test cases	Verify unit behavior
<b>Result Analysis</b>	Evaluate outcomes	Identify defects
<b>Defect Fixing</b>	Correct found issues	Ensure unit quality

- **Benefits:** Early defect detection, easier debugging, improved code quality
- **Tools:** JUnit, NUnit, automated testing frameworks
- **Coverage:** Aim for high code coverage (statements, branches, paths)

#### Mnemonic

“Units Test Individual Components Thoroughly”

### Question 5(a) OR [3 marks]

Give the full form of the following.

#### Solution

Abbreviation	Full Form
<b>AOA</b>	Activity On Arrow
<b>PERT</b>	Program Evaluation and Review Technique
<b>EVA</b>	Earned Value Analysis
<b>CPM</b>	Critical Path Method
<b>WBS</b>	Work Breakdown Structure
<b>PMC</b>	Project Management and Control

#### Mnemonic

“Activities On Arrows, Programs Evaluate Review Techniques, Earned Values Analyzed, Critical Paths Managed, Work Broken Structured, Projects Managed Controlled”

### Question 5(b) OR [4 marks]

Explain code inspection.

#### Solution

Code inspection is a systematic examination of source code by team members to identify defects and ensure quality standards.

**Code Inspection Process:**

Phase	Activity	Participants
<b>Planning</b>	Schedule inspection meeting	Moderator
<b>Preparation</b>	Review code individually	All inspectors
<b>Inspection Meeting</b>	Discuss findings	Team members
<b>Rework</b>	Fix identified issues	Author
<b>Follow-up</b>	Verify corrections	Moderator

- **Benefits:** Early defect detection, knowledge sharing, improved code quality
- **Roles:** Author, Moderator, Reviewers, Recorder
- **Focus areas:** Logic errors, coding standards, maintainability

### Mnemonic

“Inspections Improve Code Quality”

## Question 5(c) OR [7 marks]

Define white box testing method. Explain different white box testing methods.

### Solution

**White Box Testing Definition:** Testing method that examines internal code structure, logic paths, and implementation details to ensure thorough coverage.

**White Box Testing Methods:**

Method	Description	Coverage Focus
<b>Statement Coverage</b>	Execute every statement	All code lines
<b>Branch Coverage</b>	Test all decision outcomes	If-else conditions
<b>Path Coverage</b>	Execute all possible paths	Complete execution flows
<b>Condition Coverage</b>	Test all condition combinations	Boolean expressions

**Testing Techniques:**

```
mindmap
  root((White Box Testing))
    Statement Testing
      Line Coverage
      Code Execution
    Branch Testing
      Decision Points
      True/False Paths
    Path Testing
      All Routes
      Loop Testing
    Condition Testing
      Boolean Logic
      Multiple Conditions
```

**Coverage Analysis:**

Technique	Formula	Purpose
<b>Statement</b>	Executed statements / Total statements	Ensure all code runs
<b>Branch</b>	Tested branches / Total branches	Cover all decisions
<b>Path</b>	Tested paths / Total paths	Complete flow coverage

- **Tools:** Code coverage analyzers, debugging tools
- **Advantages:** Thorough testing, identifies dead code, ensures quality
- **Disadvantages:** Requires code knowledge, time-consuming, may miss requirement gaps

### Mnemonic

“White Box Sees Inside Code Structure”