K. K. Wagh Institute of Engineering Education and Research, Nashik

(An Autonomous Institute from A. Y. 2022-23)

Model Answer

End-Sem Examination-I, Winter 2025

Academic Year: 2025-2026	Semester: I
Class: FYMCA	Program: MCA
Branch Code:	Pattern: 2022
Name of Course: Software Engineering	Course Code: MCA221005

Q. No.	Answer Details	Max. Marks
Q.1.	a.) List the generic framework activities of a software process and state their purpose in brief. **Answer** A software process consists of a set of activities required to develop, deliver, and maintain software. The generic framework activities of a software process are: 1. Communication Purpose: To interact with stakeholders to understand and gather software requirements. 2. Planning Purpose: To estimate resources, schedule tasks, and plan project activities. 3. Modeling Purpose: To create representations of the software such as design models and specifications. 4. Construction Purpose: To develop the software through coding and testing. 5. Deployment Purpose: To deliver the software to the customer and obtain feedback. These activities are common to all software process models and provide a structured approach to software development.	[6]
Q.2.	a.) List and briefly explain the types of requirements represented in the Kano model. **Answer** The Kano model is a technique used in requirements engineering to classify customer requirements based on their impact on customer satisfaction. The types of requirements represented in the Kano model are: 1. Must-Be Requirements (Basic Needs) These are mandatory requirements. If they are not present, the customer will be dissatisfied, but their presence does not increase satisfaction. 2. Performance Requirements (One-Dimensional Needs)	[6]

	Customer satisfaction increases proportionally with the level of fulfillment of these requirements. 3. Attractive Requirements (Excitement Needs) These requirements are not expected by customers, but when provided, they greatly increase customer satisfaction. 4. Indifferent Requirements These requirements do not have any significant effect on customer satisfaction whether they are present or not. 5. Reverse Requirements These requirements cause dissatisfaction when present and satisfaction when absent. The Kano model helps in prioritizing requirements based on customer satisfaction.	
Q.3.	a) Apply the design model to a Student Information System and explain: Data design, Architectural design, Interface design, Component-level design. Show how these models interact during development. **Answer** A Student Information System (SIS) manages student data such as admissions, courses, attendance, and results. The design model is applied as follows: 1. Data Design Data design defines how data is structured and stored. In a Student Information System: • Entities such as Student, Course, Faculty, Enrollment, Attendance, and Results are identified. • Relationships like *Student enrolls in Course* and *Faculty teaches Course* are defined. • Data constraints such as primary keys, foreign keys, and data integrity rules are specified. This ensures consistent and reliable data storage. 2. Architectural Design Architectural design defines the overall structure of the system. For SIS: • A layered architecture is used: o Presentation Layer – User interface for students and staff o Application Layer – Business logic such as enrollment and grading o Data Layer – Database management • This separation improves scalability and maintainability. 3. Interface Design Interface design focuses on interaction between users and the system. In SIS: • Interfaces such as Student Dashboard, Admin Panel, and Faculty Portal are designed.	[8] [8]

- Clear navigation, consistent layout, and form validation are **provided.**
- Error messages and feedback are included for better usability.

This improves user experience and reduces errors.
4. Component-Level Design
Component-level design defines internal working of modules.
Examples:

- Student Management Component – handles student registration
- Course Management Component – manages course details
- Result Processing Component – calculates and stores grades

Each component has defined inputs, outputs, and processing logic.
5. Interaction among Design Models

- Data design supports component operations.
- Architectural design organizes components into layers.
- Interface design communicates with components through the architecture.
- Component-level design implements business logic using the data structure.

Together, these models work cohesively to ensure systematic software development.

OR

b) Given a Library Management System, apply software design concepts to:
Identify abstraction, modularity, information hiding, and refinement
Explain how each concept improves the system design.

Answer
A Library Management System (LMS) is used to manage books, members, issue/return operations, and fines. Software design concepts help in creating a clear, maintainable, and scalable system.
1. Abstraction
Application:
In LMS, abstraction is applied by identifying high-level entities such as Book, Member, Librarian, and Transaction without showing internal details like database queries.
Improvement:
Abstraction reduces complexity by focusing on essential features, making the system easier to understand and design.
2. Modularity
Application:
The system is divided into modules such as:

- Book Management Module
- Member Management Module
- Issue/Return Module
- Fine Calculation Module

Improvement:
Modularity allows independent development, testing, and maintenance of each module, improving system reliability.
3. Information Hiding

Application:
Internal data such as book availability logic, fine calculation rules, and database access are hidden inside modules and accessed only through defined interfaces.
Improvement:
Information hiding prevents unauthorized access and reduces the impact of changes, enhancing security and maintainability.
4. Refinement
Application:
High-level operations like *Issue Book* are refined into detailed steps such as:

- Validate member
- Check book availability
- Update records
- Generate due date

Improvement:
Refinement provides step-by-step clarity, helping developers implement functionality accurately and efficiently.

c.) Apply user interface design principles to design a UI for a Mobile Banking Application, focusing on: User familiarity, Consistency, Error prevention, Feedback mechanisms.

Answer

A Mobile Banking Application allows users to perform financial transactions such as balance enquiry, fund transfer, and bill payments. Applying user interface design principles ensures safety, usability, and user trust.

1. User Familiarity

Application:

- Use familiar banking terms such as Account Balance, Transfer Funds, Mini Statement.
- Follow common mobile app navigation like bottom tabs and hamburger menu.

Benefit:

User familiarity reduces learning time and helps users operate the app confidently.

2. Consistency

Application:

- Maintain consistent color schemes, fonts, icons, and button placement across all screens.

- Use the same flow for actions like login, OTP verification, and confirmation.

Benefit:

Consistency improves usability and avoids confusion during transactions.

3. Error Prevention

Application:

- Input validation for account numbers and IFSC codes.
- Disable the *Submit* button until all mandatory fields are correctly filled.
- Confirmation screen before final fund transfer.

Benefit:

Error prevention reduces transaction failures and prevents financial loss.

4. Feedback Mechanisms

Application:

- Display real-time messages such as “Processing…”, “Transaction Successful”, or “Transaction Failed”.
- Use notifications, vibration, or sound alerts for important actions.

Benefit:

Feedback assures users about system status and increases trust in the application.

OR

d.) Apply different architectural styles to the following scenarios:

- Real-time monitoring system
- Banking system
- Content management system

Select and justify appropriate styles for each case.

Answer

Different software systems require different architectural styles based on their functional and non-functional requirements.

1. Real-Time Monitoring System

	Selected Architectural Style: Event-Driven / Pipe-and-Filter Architecture Justification: • Handles continuous data streams from sensors. • Supports fast event processing and real-time response. • Allows parallel data processing and easy scalability. Examples include traffic monitoring and industrial control systems. 2. Banking System Selected Architectural Style: Layered Architecture (Multi-tier) Justification: • Separates presentation, business logic, and data layers. • Enhances security, reliability, and maintainability. • Supports transaction processing and regulatory compliance. Commonly used in core banking applications. 3. Content Management System (CMS) Selected Architectural Style: Model–View–Controller (MVC) Justification: • Separates content, user interface, and control logic. • Simplifies content updates and UI changes. • Improves maintainability and scalability. Widely used in web-based CMS platforms.	
Q.4.	a.) For a Student Management System, apply the following size metrics: Lines of Code (LOC) , Token Count, Function Count. Explain how each metric can be used for effort estimation. **Answer** A Student Management System (SMS) manages student records, courses, attendance, and results. Size metrics are used to estimate development effort based on system size. 1. Lines of Code (LOC) Metric Application: In SMS, LOC is estimated by calculating the total number of source code lines required to implement modules such as:	[8] [8]

- Student Registration
- Course Management
- Attendance Tracking
- Result Processing

Effort Estimation:

Higher LOC indicates more development effort. Effort can be estimated using historical data, for example:

Effort = LOC / Productivity (LOC per person-month)

2. Token Count Metric

Application:

Token count measures the number of operators, operands, keywords, and identifiers used in the source code of SMS modules.

Effort Estimation:

A higher token count reflects increased code complexity and development effort. It helps estimate effort early in coding by measuring logical size rather than physical lines.

3. Function Count Metric

Application:

Function count estimates size based on system functionality such as:

- Add Student
- Enroll Course
- Generate Reports
- Calculate Grades

Effort Estimation:

Effort is estimated by assigning weights to functions and using standard productivity rates, making it suitable during early development stages.

OR

b) Apply data structure metrics and information flow metrics to a Library Management System to Measure complexity and Identify modules with high information flow.

Answer

A Library Management System (LMS) manages books, members, transactions, and fines. Data structure metrics and information flow metrics are applied to assess system complexity and communication between modules.

1. Data Structure Metrics Application:
Data structure metrics measure complexity based on the number and type of data structures used in the system.
In LMS, major data structures include:

- Book records (arrays or lists)
- Member details (structures or objects)
- Transaction logs (queues or linked lists)
- Fine records (tables)

Modules such as Book Management and Transaction Processing use multiple complex data structures.
Complexity Measurement:
A higher number of complex data structures (linked lists, trees, hash tables) increases module complexity. Thus, modules handling book inventory and issue–return operations are considered more complex.

2. Information Flow Metrics
Application:
Information flow metrics measure complexity based on data flow between modules, considering:

- Fan-in (number of modules calling a module)
- Fan-out (number of modules called by a module)

In LMS:

- Transaction Module has high fan-in (called by Issue, Return, and Fine modules).
- Database Access Module has high fan-out (interacts with multiple modules).

Identification of High Information Flow Modules:
Modules with high fan-in and fan-out, such as Transaction Processing, Database Management, and Fine Calculation, have high information flow and require careful design and testing.

c) Apply use-case oriented metrics to an Online Exam Portal by identifying actors and use cases , estimating project size and effort based on use cases.

Answer
An Online Exam Portal allows institutions to conduct examinations digitally. Use-case oriented metrics estimate software size and effort based on user interactions with the system.

1. Identification of Actors
The main actors involved in an Online Exam Portal are:

- Student – appears for exams and views results
- Exam Administrator – manages exams and students

- Faculty/Examiner – creates questions and evaluates answers
- System Administrator – maintains system and users

2. Identification of Use Cases
Major use cases include:

- Student Registration and Login
- Exam Scheduling
- Question Paper Creation
- Attempt Online Exam
- Auto Evaluation and Result Generation
- Result Viewing and Report Generation

Each use case represents a functional requirement.

3. Project Size Estimation Using Use-Case Metrics
Use cases are classified as:

- Simple (e.g., Login, View Result)
- Average (e.g., Register Student, Generate Report)
- Complex (e.g., Conduct Online Exam, Auto Evaluation)

Weights are assigned to each category. The total number of weighted use cases gives an estimate of the project size.

4. Effort Estimation Based on Use Cases
Effort is estimated by multiplying the total use-case points with a productivity factor (e.g., person-hours per use-case point).
Effort = Use-Case Points × Productivity Factor
This helps estimate development time, cost, and manpower required for the Online Exam Portal.

OR

d.) For a content-driven website, apply web engineering project metrics to estimate development effort, measure productivity and quality.

Answer
A content-driven website (such as a news portal or blog platform) focuses on creating, managing, and publishing large volumes of content. Web engineering project metrics are used to estimate effort, productivity, and quality.

1. Effort Estimation Metrics
Application:
Effort is estimated using metrics such as:

- Number of web pages (static and dynamic)
- Content volume (articles, images, videos)
- Functional components (search, comments, CMS)

For example, a higher number of dynamic pages and multimedia content increases development effort.
Outcome:
These metrics help estimate development time, cost, and team size.

	2. Productivity Metrics Application: Productivity is measured using: • Pages developed per person-month • Content modules completed per sprint Outcome: Higher productivity indicates efficient use of resources and faster content deployment. 3. Quality Metrics Application: Quality is measured using: • Defect density per web page • Broken link count • Page load time and usability feedback Outcome: Lower defect density and faster response time indicate higher website quality. 4. Relevance to Content-Driven Website Since content-driven websites require frequent updates, these metrics help maintain performance, reliability, and user satisfaction over time.	
Q.5.	a.) Apply appropriate leadership styles and emotional intelligence in a software project team to motivate team members and manage stress and interpersonal issues **Answer** In a software project team, effective leadership and emotional intelligence are essential to maintain motivation, productivity, and team harmony, especially under tight deadlines. 1. Application of Leadership Styles • Democratic Leadership: The project leader involves team members in decision-making during sprint planning and design discussions. *Impact:* Team members feel valued, increasing motivation and ownership. • Transformational Leadership: The leader sets a clear project vision and encourages innovation while recognizing individual contributions. *Impact:* Inspires team members to perform beyond expectations. • Situational Leadership: The leader adapts style based on team experience—providing guidance to juniors and autonomy to seniors. *Impact:* Ensures efficient task execution and confidence building. 2. Application of Emotional Intelligence • Self-Awareness and Self-Control: The leader remains calm during high-pressure releases, preventing	[8] [8]

stress from spreading to the team.
- Empathy:
  Understanding personal challenges of team members and adjusting workloads accordingly.
- Social Skills:
  Resolving interpersonal conflicts through open communication and constructive feedback.

Impact: Reduces stress, improves collaboration, and maintains a positive work environment.

3. Relevance to Software Project Team
Applying suitable leadership styles along with emotional intelligence ensures high morale, reduced conflicts, and sustained productivity in software development projects.

OR

b) Apply software engineering codes of ethics to a real-world project scenario involving data privacy, user safety and professional responsibility.
Answer
Consider a mobile health-tracking application that collects user health data and provides fitness recommendations. Applying software engineering codes of ethics is critical to ensure responsible system development.

1. Data Privacy
According to the ACM/IEEE Code of Ethics, software engineers must respect user privacy and confidentiality.
Application:
- Collect only necessary user data with explicit user consent.
- Store data securely using encryption and access controls.
- Do not share personal data with third parties without authorization.

Ethical Impact:
This ensures trust and protects users from data misuse or breaches.

2. User Safety
The code emphasizes protecting the public interest and avoiding harm.
Application:
- Validate algorithms to avoid incorrect health recommendations.
- Clearly display disclaimers for medical advice.
- Thoroughly test the application to prevent crashes or misleading outputs.

Ethical Impact:
Ensures the software does not endanger users' health or well-being.

3. Professional Responsibility
Software engineers are ethically responsible for their work.
Application:
- Report known defects or security vulnerabilities to management.

- Avoid releasing software that fails safety or privacy standards.
- Follow professional practices and legal regulations.

4. Real-World Relevance
By following ethical codes, the project ensures legal compliance, user trust, and social responsibility.

c) A software engineer discovers a security flaw just before product release. Apply ethical decision-making steps and professional codes of ethics to decide the correct course of action

Answer

When a software engineer discovers a security flaw just before product release, ethical decision-making and adherence to professional codes of ethics are essential to protect users and public interest.

1. Ethical Decision-Making Steps
   1. Identify the Problem:
      The security flaw may expose user data or system functionality to unauthorized access.
   2. Assess Stakeholders and Impact:
      Stakeholders include users, the organization, and society. Releasing the product could cause financial loss, privacy violations, and legal consequences.
   3. Evaluate Alternatives:
      - Ignore the flaw and release on time
      - Inform management and delay release
      - Release with a warning (not ethical if risk is high)
   4. Choose the Ethical Action:
      The engineer should report the flaw immediately and recommend fixing it before release.

2. Application of Professional Codes of Ethics
According to the ACM/IEEE Software Engineering Code of Ethics:
- Public Interest: Engineers must act in the best interest of users and the public by preventing harm.
- Honesty: The engineer must truthfully disclose the flaw to management.
- Professional Responsibility: Engineers must not approve or release software that is unsafe or insecure.

Thus, releasing flawed software violates professional ethics.

3. Correct Course of Action
The correct course of action is to delay the release, fix the security flaw, and ensure the software meets safety and security standards before deployment.

OR

d.) Apply the SCM process and SCM repository to manage source code , documentation and test cases in a team-based development environment.

Answer

In a team-based software development environment, multiple developers work simultaneously on source code, documentation, and test cases. Software Configuration Management (SCM) ensures controlled and consistent development.

1. Application of SCM Process
   - Configuration Identification:
     Source code files, design documents, requirement documents, and test cases are identified as configuration items (CIs).
   - Version Control and Change Control:
     Changes to code, documents, or test cases are made only through approved change requests. Versions are assigned to track modifications.
   - Configuration Status Accounting:
     Records of versions, change history, and release status are maintained for all configuration items.
   - Configuration Audits:
     Regular audits ensure that correct versions of code, documents, and test cases are included in releases.

This process ensures consistency and traceability.

2. Application of SCM Repository

An SCM repository acts as a central storage for all project artifacts:
   - Source Code Management:
     Developers use branching and merging to work independently while avoiding conflicts.
   - Documentation Management:
     Requirement and design documents are versioned and updated in sync with code changes.
   - Test Case Management:
     Test cases and test reports are stored with version control to match corresponding software builds.

The repository supports collaboration and prevents data loss.

3. Relevance to Team-Based Environment

Using SCM processes and a shared repository enables parallel development, controlled changes, and reliable releases in a team environment.