Daffodil International University

Department of Software Engineering
Faculty of Science & Information Technology
Midterm Examination, Fall 2024
Course Code: SE 113; Course Title: Introduction to Software Engineering
Sections & Teachers: All & AP, DAKM, MKS, JIC, TH, SAN, MIM, ST

Time: 1 Hour 30 Mins **Marks: 25**

Answer ALL Questions

[The figures in the right margin indicate the full marks and corresponding course outcomes. All portions of each question must be answered sequentially.]

1. *a)* **Define** Software Engineering and **detail** the core responsibilities of software engineers. [Marks-3] CLO-1 *Level-2*

   b) **Describe** the work scopes of a Software Engineer and a Software Developer with real-life examples, highlighting their key responsibilities and differences. [Marks-3] CLO-1 *Level-2*

   c) **Define** functional requirements and non-functional requirements of software. Suppose, Daffodil International University is developing a Bus Seat Booking System where students can pre-book seats, view real-time availability, and get booking confirmations. The system aims to reduce overcrowding and improve bus service efficiency. **Classify** the key functional and non-functional requirements for this system. [Marks-3] CLO-1 *Level-2*

   d) **Distinguish** between High-Level Design and Low-Level Design. [Marks-2] CLO-1 *Level-2*

   e) **Contrast** Quality Assurance and Quality Control with an example. *[Marks-3]* *CLO-1* *Level-2*

   f) You are leading a project to develop a comprehensive e-learning platform for a university. The project involves various stakeholders, including educators, students, and administrators, each with specific needs and expectations. **Demonstrate the different phases of Software Project Management** you would implement to guide the project from start to finish. For each phase, including the different activities you would perform, how you would manage changing requirements, and the tools you would use to ensure the project is completed on time, within budget, and meets the quality standards required by the university. [Marks-4] CLO-2 *Level-3*

2 *a)* Imagine you are leading a team to develop a new banking system with features like secure transactions, user interfaces, and risk management. Given the need for ongoing risk analysis, prototype development, and continuous feedback. **Detect** a suitable Software Process model considering these points. **Describe** the activities of each phase for the model. [Marks-5] 5 CLO-2 *Level-3*

   b) Imagine you are managing a software application for inventory management in a retail company. After deployment, [Marks-2] CLO-2 *Level-3*

   i. Users report a bug that prevents the system from updating inventory levels accurately.
   ii. The company's IT infrastructure has been upgraded to a new operating system, and the software needs to be compatible with this new environment.
   iii. Users also request new features, such as advanced reporting and data analytics, to improve their experience.

Determine which type of maintenance to apply for above mentioned scenarios.

1. a) Definition and Responsibilities of Software Engineering

Software Engineering is the systematic application of engineering approaches to the development of software. It involves principles, methods, and tools to design, develop, maintain, test, and evaluate software systems.

Core Responsibilities of Software Engineers:

* Analyzing user requirements.

* Designing software solutions.

* Writing clean, efficient, and maintainable code.

* Testing and debugging software.

* Maintaining and upgrading existing systems.

* Collaborating with cross-functional teams.

* Ensuring software quality and performance.

1.b)Software Engineer

A Software Engineer focuses on applying engineering principles to the entire software development life cycle (SDLC), ensuring the system is reliable, scalable, and efficient.

Key Responsibilities

* System Design & Architecture: They design overall system architecture (e.g., microservices for a large application).

* Problem Analysis: Identify technical constraints and propose solutions before coding begins.

* Performance & Scalability: Ensure the software can handle large loads (e.g., an e-commerce site during Black Friday sales).

* Integration & Infrastructure: Work with hardware, databases, and networks to integrate software properly.

* Quality & Standards: Apply engineering principles, use design patterns, and follow strict coding standards.

Real-Life Example

* A Software Engineer at Google designing the architecture for Google Maps to handle billions of location queries daily.

* They decide on load balancing strategies, choose database models (e.g., NoSQL for geolocation data), and ensure APIs are secure and scalable.

Software Developer

A Software Developer primarily focuses on writing, implementing, and maintaining code to build applications based on the design provided by engineers or analysts.

Key Responsibilities

* Coding & Implementation: Write functional code according to requirements.

* Debugging & Testing: Fix bugs and ensure the application runs smoothly.

* Feature Development: Add new features to applications.

* Documentation: Maintain clear code documentation.

* Collaboration: Work with UI/UX designers, testers, and engineers to deliver the product.

Real-Life Example

* A Software Developer at a startup creating the mobile front-end for a food delivery app (like Pathao Food or Foodpanda).

* They focus on coding the app interface, implementing APIs for order tracking , and ensuring it runs well on Android and iOS.

Key Differences

Aspect	**Software Engineer**	**Software Developer**
Scope	System-level design & architecture	Application-level coding & implementation
Focus	Big-picture problem solving, scalability, security	Building features, writing clean code
Life Cycle	Involved in full SDLC	Mostly in implementation & maintenance
Example Task	Designing system architecture for cloud services	Building a login feature for an app

1.c)Definition

Functional Requirements

These are the specific features and functionalities that the software system must provide. They describe what the system should do to meet the user's needs.

Non-Functional Requirements (NFRs)

These define quality attributes, performance constraints, and system behavior under specific conditions. They describe how the system should perform, not what it does.

For Daffodil International University Bus Seat Booking System

✅ **Functional Requirements (What the system will do)**

1. User Registration & Login

    * Students must register and log in using their university ID.

2. Seat Booking Feature

    * Students can select a bus, choose available seats, and confirm booking.

3. Real-Time Seat Availability

    * The system should display available seats in real-time.

4. Booking Confirmation & Notification

    * After successful booking, the system sends a confirmation message (SMS or email).

5. Cancellation & Refund

* Students can cancel their bookings within a specified time.

6. Bus Schedule Display

    * Display bus routes, timings, and pick-up points.

7. Admin Dashboard

    * Admin can manage buses, schedules, and monitor seat occupancy.

✓ **Non-Functional Requirements (How the system will perform)**

1. Performance

    * The system should handle at least 500 concurrent users without delay.

2. Availability

    * The system must be available 24/7 during semester days.

3. Security

    * All student data must be encrypted, and login should require authentication.

4. Usability

    * The interface should be mobile-friendly and easy to use for students.

5. Reliability

    * The system should ensure 99.9% uptime to avoid booking failures.

6. Scalability

    * The system should support adding more buses and routes in the future.

7. Response Time

    * Seat availability should update within 2 seconds after a booking.

8. Data Integrity

    * No two users can book the same seat at the same time.

✓ **Summary Table**

Type	Requirement
Functional	Seat booking, real-time availability, notifications
Non-Functional	Security, performance, usability, scalability

1. d) High-Level Design vs Low-Level Design

High-Level Design (HLD):

* Architecture-level design.

* Overview of modules/components.

* Technologies and data flow.

* Example: Using a microservices architecture with React frontend and Node.js backend.

Low-Level Design (LLD):

* Detailed design of modules and functions.

* Database schema, class diagrams.

* Algorithm specifications.

* Example: Designing the login function logic and database query structures.

1. e) Quality Assurance vs Quality Control

Quality Assurance (QA):

* Proactive process.

* Focuses on improving the development process to prevent defects.

* Example: Conducting code reviews and following development guidelines.

Quality Control (QC):

* Reactive process.

* Focuses on identifying defects in the final product.

* Example: Testing a website for bugs before release.

1. f) Phases of Software Project Management for E-Learning Platform

1. Initiation Phase:

* Define objectives.

* Identify stakeholders (students, teachers, admins).

* Perform feasibility study.

* Tools: Stakeholder analysis matrix, project charter.

2. Planning Phase:

* Define scope, budget, timelines.

* Resource allocation.

* Risk management plan.

* Tools: Gantt Chart, WBS, Risk Register.

3. Execution Phase:

* Development of platform (frontend, backend).

* Regular team meetings.

* Stakeholder feedback sessions.

* Tools: JIRA, GitHub, Slack.

4. Monitoring & Controlling Phase:

* Track progress using KPIs.

* Manage changes via change requests.

* Ensure milestones are met.

* Tools: Burndown charts, Change Control Board.

5. Closing Phase:

* Final product delivery.

* Documentation and training.

* Post-mortem analysis.

Managing Changing Requirements:

* Use Agile methodology.

* Regular sprint reviews and client feedback.

Ensuring Project Success:

* Tools: Agile boards, Continuous Integration tools (e.g., Jenkins), Testing frameworks (e.g., Selenium).

* Maintain frequent communication with stakeholders.

* Prioritize tasks and manage scope creep.

2. a) Suitable Software Process Model: Spiral Model

Given the project requirements—secure transactions, user-friendly interfaces, risk management, prototype development, and continuous feedback—the Spiral Model is the most appropriate.

Phases of Spiral Model:

1. Planning Phase:

   * Define objectives and constraints.

   * Identify requirements from stakeholders.

   * Estimate time and cost.

   * Example: Determine banking features like account creation, secure login, fund transfer.

2. Risk Analysis Phase:

   * Identify potential project risks (e.g., data breach, failed integration).

   * Evaluate alternative solutions and mitigation strategies.

   * Example: Analyze risks of transaction failures under peak loads.

3. Engineering Phase (Development and Testing):

   * Design and develop prototypes for selected modules.

   * Perform coding and unit testing.

   * Example: Create a working model of the transaction system or login module.

4. Evaluation Phase (Customer Feedback):

   * Present the prototype to users and stakeholders.

   * Collect feedback for improvements.

   * Update requirements based on input.

5. Repeat the Spiral (Iterative Refinement):

   * Each loop around the spiral adds more detailed features and reduces risk.

   * The system is incrementally built with better quality and less uncertainty.

Why Spiral Model?

* Supports iterative development.

* Strong focus on **risk management**.

* Allows incorporation of **user feedback** in every iteration.

* Best suited for **large and complex** systems like banking applications.

2. b) Types of Software Maintenance**

i. Bug in Inventory Update

*Type: Corrective Maintenance

*Explanation: This bug affects the core functionality of the system (inventory updates). Corrective maintenance involves fixing bugs and resolving defects reported after deployment.

*Example Task: Fix the faulty inventory update logic and test it for accuracy.

ii. OS Compatibility After Upgrade

*Type: Adaptive Maintenance

*Explanation: The software must adapt to changes in the external environment (new operating system). Adaptive maintenance ensures the system continues to function in the new environment.

*Example Task: Modify system drivers, database paths, or compatibility layers.

iii. Request for New Features (Advanced Reporting and Analytics)

*Type: Perfective Maintenance

*Explanation: Enhancing the system with new capabilities to improve user experience and meet evolving business needs.

*Example Task: Add dashboards, graphs, and data export options for advanced reporting.

Daffodil International University

Faculty of Science & Information Technology

Department of Software Engineering

Midterm Examination, Spring 2025; Course Code: SE 113

Course Title: Intrôduction to Software Engineering

Sections & Teachers: A, B, C, D, E, F, G, H, I, J, K, L, M & SS, MKS, NS, KR, LA, SA

Time: 1:30 Hrs **Marks: 25**

Answer ALL Questions

[The figures in the right margin indicate the full marks and corresponding course outcomes. All portions of each question must be answered sequentially.]

1.	*a)*	**Explain** which characteristics are common in a high quality software with reasons.	[Marks-4]	**CLO-1** *Level-2*
	b)	Briefly **discuss** the basic responsibilities of a software engineer that are essential to develop software.	[Marks-4]	
	c)	**Predict** an appropriate career that you may like to have in near future with reasons.	[Marks-3]	
	d)	**Distinguish** between plan-driven and agile processes	[Marks-3]	
	e)	**Differentiate** between the waterfall model and incremental model with examples.	[Marks-4]	
2.	*a)*	The upcoming software project to serve the Non Residential Bangladeshi citizens would be very time critical as well as hard to be satisfactory because of high variety of scopes – some of which are still not clear. As the foreign ministry is really eager to complete the project with high satisfaction of all the stakes which is a bit risky, hence the ministry likes to provide flexibility to the development team with budget and working environment. **Determine** a related Software Development Life Cycle model for the above-mentioned issues.	[Marks-3]	**CLO-2** *Level-3*
	b)	**Derive** a SDLC model with each phase that relate the suitability for case given in scenario 2.a)	[Marks-4]	

1(a) Common characteristics of high-quality software with reasons

High-quality software generally has the following characteristics:

i)Correctness – The software meets all requirements and performs the intended functions without errors. Reason: Users can rely on it to get accurate results.

ii)Reliability – It consistently performs well under expected conditions. Reason: Reduces downtime and user frustration.

iii)Efficiency – Uses system resources (CPU, memory, storage) effectively. Reason: Improves speed and reduces operational costs.

iv)Usability – Easy to learn and operate. Reason: Users can work productively without extensive training.

v)Maintainability – Easy to update, fix, or improve. Reason: Saves time and cost in future modifications.

vi)Portability – Can run on different platforms or environments with minimal changes. Reason: Increases flexibility and market reach.

1(b) Basic responsibilities of a software engineer

A software engineer's main responsibilities include:

i)Requirement analysis – Understanding user needs and documenting them clearly.

ii)Design – Creating system architecture and module designs to meet requirements.

iii)Development – Writing clean, efficient, and maintainable code.

iv)Testing – Checking for bugs, errors, and ensuring the software meets standards.

v)Deployment – Installing and configuring the software for users.

vi)Maintenance – Updating and improving the software after release.

vii)Collaboration – Working with designers, testers, and clients to ensure success.

These responsibilities ensure the final product is functional, reliable, and user-friendly.

1(c) Predict an appropriate career I may like to have in the near future with reasons

I would like to have a career as a **Software Developer**.

Reasons:

i)I enjoy problem-solving and logical thinking, which are key skills in programming.

ii)The software industry is growing, offering many job opportunities worldwide.

iii)It allows me to work on creative projects like web apps, mobile apps, or AI solutions.

iv)There is flexibility to work remotely or in different industries.

v)The career offers continuous learning and good earning potential.

1(d) Difference between plan-driven and agile processes

Plan-driven process:

i)Follows a fixed, detailed plan from start to finish.

ii)Requirements are defined early and rarely change.

iii)Suitable for large, well-defined projects like government systems.

Example: Waterfall model.

Agile process:

i)Flexible and adapts to changes throughout development.

ii)Work is divided into small iterations with frequent feedback.

iii)Suitable for projects with changing requirements, like mobile app development.

Example: Scrum, Extreme Programming (XP).

1(e) Difference between Waterfall model and Incremental model with examples

Waterfall model:

i) Sequential process where each phase (requirement, design, coding, testing,deployment) is completed before the next begins.

ii) Changes are difficult once a phase is finished.

Example: Developing a fixed banking system where requirements are clear from the start.

Incremental model:

i)Development is done in small parts (increments). Each increment adds new functionality.

ii)Allows early delivery of working software and easier changes.

Example: Building an e-commerce site where basic shopping cart is launched first, then payment gateway, then order tracking in later releases.

2.a)Recommended SDLC Model: Spiral Model

Reason:

*The project has unclear and evolving requirements, making traditional models like Waterfall unsuitable.

*It is time-critical and risky, requiring risk analysis and stakeholder feedback at every stage.

*Spiral Model combines iterative development with risk management, enabling flexibility in budget and scope.

Key Features:

*Phases: Planning → Risk Analysis → Engineering → Evaluation (repeated in spirals).

*Allows prototyping, continuous stakeholder involvement, and progressive delivery.

Conclusion:

Spiral Model is best suited because it ensures flexibility, risk control, and stakeholder satisfaction while delivering the project on time.

2.b)Derived SDLC Model: Spiral Model

Suitability for the Scenario:

The project for Non-Residential Bangladeshi citizens has unclear requirements, high risks, and is time-sensitive. The Spiral Model fits as it supports risk management, iterative development, and continuous stakeholder involvement.

Phases of Spiral Model and Relevance to Scenario

i)Planning Phase:

*Define objectives, identify features for NRBs (e.g., passport renewal, attestation).

*Collect initial requirements and set constraints with the ministry.

ii)Risk Analysis Phase:

*Analyze technical, security, and compliance risks (e.g., identity verification, data protection.

*Develop prototypes to validate uncertain features.

iii)Engineering Phase:

*Develop system incrementally (e.g., appointment booking first, then other services).

*Test each module for functionality, security, and performance.

iv)Evaluation Phase:

*Review deliverables with stakeholders (Foreign Ministry, NRBs).

*Gather feedback, update requirements, and plan the next iteration.

Conclusion:

The Spiral Model is best because it handles evolving requirements, mitigates risks, and ensures stakeholder satisfaction while meeting time constraints.

Daffodil International University

Faculty of Science & Information Technology

Department of Software Engineering

Midterm Examination, Summer 2025; Course Code: SE 113;

Course Title: Introduction to Software Engineering

Sections & Teachers: 45 (A-N) & MKS, RM, MIF, KMH, ZNM, MMN, FMD, MTM

Time: 1:30 Hrs **Marks: 25**

Answer ALL Questions

[The figures in the right margin indicate the full marks and corresponding course outcomes. All portions of each question must be answered sequentially.]

1.	*a)*	A company is developing a new e-commerce platform but is facing issues like unclear requirements, frequent changes, tight deadlines, and third-party integration problems. Based on this scenario, briefly **discuss** the basic challenges commonly faced in software development.	[Marks-4]	**CLO-1** *Level-2*
	b)	A software company receives two projects from different clients. One project has well-defined, stable requirements and a fixed timeline, while the other has evolving requirements and needs continuous user feedback throughout the development process. Depending on the scenario, **distinguish** between Adaptive (Agile) and Predictive (Waterfall) software development approaches.	[Marks-5]	
	d)	**Describe** the important characteristics that are necessary for high-quality software with details.	[Marks-5]	
	e)	For any software project, **elaborate** the advantages and disadvantages of using the XP model in software development.	[Marks-4]	
2.	*a)*	A new system for farming is to be developed from the scratch (starting), therefore lots of activities - from collecting requirement from the farmers, acquiring team, planning, scheduling, cost calculating, managing people to motivate team members - would have to be carried out. This is a huge project which would be given enough flexibilities to develop your team and to acquire other resources (e.g. human or machines). However, the users of the system are expecting quick implementation of it within three months and the main stakeholders do not have enough idea about the final software and functions, few experts' think huge change of requirements would be unavoidable with time frame. You may invent other information as necessary. Based on this scenario, **determine** a suitable Software Development Life Cycle (SDLC) model that will be the best choice for this project with clear reasoning.	[Marks-4]	**CLO-2** *Level-3*
	b)	According to the answer of Question 2a, **Demonstrate** each phases that specifically addresses the needs and challenges of the SDLC model.	[Marks-3]	

Answers to Software Engineering Questions

Question 1 (a) Basic challenges commonly faced in software development

Common challenges in software development include: 1. Unclear requirements – Clients may not provide complete or stable requirements, causing scope creep.
2. Frequent changes – Requirements may evolve during development, disrupting schedules.
3. Tight deadlines – Limited time leads to pressure on developers, affecting quality.
4. Integration issues – Combining modules or third-party systems can cause compatibility problems.
5. Communication gaps – Misunderstanding between stakeholders, developers, and clients delays progress.
6. Resource constraints – Limited manpower, tools, or budget hinder progress.

Question 1 (b) Distinguish between Adaptive (Agile) and Predictive (Waterfall) approaches

AspectAdaptive (Agile)Predictive (Waterfall) RequirementsContinuously evolving; flexibleWell-defined and stable Development processIterative and incrementalSequential and linear Customer involvementContinuous feedback throughoutMainly at the beginning and end DeliveryPartial working software delivered frequentlyFinal product delivered at the end Flexibility to changeHighly flexibleDifficult and costly to change later Example: Adaptive is used when requirements evolve; Predictive is used when requirements are fixed.

Question 1 (d) Characteristics necessary for high-quality software

High-quality software should have the following characteristics: 1. Functionality – Performs required tasks accurately.
2. Reliability – Operates without failure for a specified time.
3. Usability – Easy to use and understand by end-users.
4. Efficiency – Uses system resources effectively.
5. Maintainability – Easy to update and modify.
6. Portability – Can be used across different platforms.
7. Security – Protects data and prevents unauthorized access.

Question 1 (e) Advantages and disadvantages of using XP (Extreme Programming) model

Advantages:
1. High customer satisfaction through continuous feedback.
2. Rapid delivery of functional software.
3. Improved quality through pair programming and testing.
4. Encourages teamwork and communication.

Disadvantages:
1. Requires constant customer involvement.
2. Difficult to apply for large teams.
3. Frequent changes can affect stability.
4. Less focus on formal documentation.

Question 2 (a) Suitable SDLC model for the farming system

Best Model: Agile (Scrum) Model
Reasoning:
- Requirements are unclear at the beginning.
- High uncertainty and changing requirements.
- Quick implementation needed within three months.

- Frequent feedback from users helps refine functionality.
- Encourages flexibility and adaptation.

Hence, the Agile model is best suited as it supports incremental development, accommodates frequent changes, and ensures stakeholder involvement.

Question 2 (b) Phases of Agile model that address needs and challenges

1. Requirement Gathering & Planning – Collects user stories and clarifies unclear needs.
2. Design & Iteration (Sprints) – Development in short cycles allowing quick feedback.
3. Implementation & Testing – Continuous testing ensures quality.
4. Review & Feedback – Stakeholders review outputs and suggest changes.
5. Deployment & Maintenance – Software deployed incrementally for early use and improvement.