Unit 1: Introduction to Software Development

Textbook Exercise

Multiple Choice Questions (MCQs)

1. The primary purpose of the Software Development Life Cycle (SDLC) is to:

a) Design websites
b) Deliver high-quality software within time and cost estimates
c) Manage database systems
d) Create hardware components

Correct option: b) Deliver high-quality software within time and cost estimates

Explanation: The SDLC gives teams a planned process for creating quality software on time and within the approved budget.

2. Which type of requirement specifies system performance?

a) Functional Requirements
b) Non-Functional Requirements
c) Technical Requirements
d) Operational Requirements

Correct option: b) Non-Functional Requirements

Explanation: Non-functional requirements describe how well a system works, including its speed, reliability, and security.

3. The role of a framework in the context of the SDLC is to:

a) Write code from scratch
b) Provide a structured foundation with pre-built components and architectures
c) Manage hardware
d) Perform manual testing

Correct option: b) Provide a structured foundation with pre-built components and architectures

Explanation: A framework supplies ready-made structures and components so developers can build software more quickly and consistently.

4. Which software development model involves short cycles or sprints?

a) Waterfall Model
b) Agile Methodology
c) Lean Software Development
d) Scrum

Correct option: b) Agile Methodology

Explanation: Agile develops software in short cycles called sprints and uses feedback to improve the product.

5. Which is a crucial part of comprehensive project planning?

a) Understanding the project scope and tasks
b) Deciding the project's colour scheme
c) Hiring a large development team
d) Ignoring possible risks

Correct option: a) Understanding the project scope and tasks

Explanation: Knowing the project's boundaries and required work helps the team plan time, people, and resources correctly.

6. Which factor does not normally influence the cost estimate of a software project?

a) Scope of the project
b) Technology stack
c) Number of meetings held
d) Operational costs

Correct option: c) Number of meetings held
Explanation: Project scope, technology, and operating costs directly affect the budget, while the number of meetings is not a main cost factor in this context.

7. The purpose of Use Case Diagrams is to:

a) Document the system's architecture
b) Identify and document the system's functional requirements
c) Illustrate the database schema
d) Define the system's user-interface design

Correct option: b) Identify and document the system's functional requirements
Explanation: A Use Case Diagram shows what users do with a system and therefore helps define its required functions.

Short Questions

Q.1 Differentiate between functional and non-functional requirements.

Answer: Functional requirements describe the specific actions or functions a system must perform. They explain what the system should do and are directly connected to user interactions and system tasks.

Non-functional requirements describe the quality and limits of a system. They explain how well the system should work, including performance, usability, reliability, and security.

Q.2 Explain why the testing phase is important in the SDLC, and give two reasons for its importance.

Answer: The testing phase checks whether the software works as planned and meets user needs.

1. Improves quality: Testing finds and fixes errors early, which makes the software more dependable.
2. Saves money and effort: Fixing problems during development is usually cheaper and easier than fixing them after release.

Q.3 Explain continuous integration in Agile methodology and discuss its importance in software development.

Answer: Continuous Integration (CI) means that developers regularly add their code to a shared project, often several times a day. Automated tests then check the new code and quickly show whether it causes problems.

Importance:
1. Finds problems early: Frequent checks make errors easier and faster to fix.
2. Improves teamwork: Developers share code regularly, which reduces conflicts.
3. Speeds up delivery: Quick feedback and automation help teams release features sooner.

Q.4 Explain the main steps in risk assessment and management, and state their importance in a software project.

Answer: Main steps in risk management:

1. Identify risks: List possible problems that could affect the project, such as untested technology, lack of resources, or changes in the market.
2. Analyse risks: Judge how likely each risk is and how serious its effect could be. This helps the team deal with the most serious risks first.
3. Prepare risk-reduction plans: Create steps to lower the chance of a risk or reduce its effect. Examples include adding extra time, arranging backup resources, or doing more tests.
4. Monitor and review: Keep checking for new risks and update the plans when needed.

Importance:
Risk management prevents unexpected problems, supports good decisions about time and resources, and increases the chance of completing the project successfully.

Q.5 Explain the purpose of a Use Case Diagram in software development.

Answer: A Use Case Diagram shows how users interact with a system and which features the system provides.

Its purposes are to:
1. Show what the system does and make its features clear.
2. Improve communication because both technical and non-technical people can understand it.
3. Guide development by showing the team what needs to be built.
4. Set boundaries by showing what the system will and will not do.

Q.6 Compare and contrast a Sequence Diagram with an Activity Diagram.

Answer: A Sequence Diagram focuses on how objects communicate with one another over time. It shows the exchange of messages in a particular situation, such as the communication between a user and a server during login. Time normally moves from top to bottom.

An Activity Diagram focuses on the flow of work or a process. It shows steps, decisions, and control flow, such as the steps used to process an online order. Time is not shown as directly as it is in a Sequence Diagram.

Q.7 Describe the Factory Pattern and explain how it differs from creating objects directly, with an example.

Answer: The Factory Pattern creates objects through a special factory class or method instead of creating them directly in the main code. For example, a payment factory can return a CreditCardPayment or PayPalPayment object according to the user's choice.

The Factory Pattern is more flexible because the object-creation code can change in one place. It also reduces dependence on specific classes and is easier to maintain. Direct creation uses the new keyword in many places, so changing the object type can require changes throughout the program. Direct creation is suitable when object creation is simple and used only once.

Long Questions

Q.1 Design a flowchart for a user-registration process in a software application. Outline its key steps.

Answer: Flowchart steps:

1. Start: Begin the registration process.
2. Display the registration form: Ask the user to enter details such as name, email address, and password.
3. Receive user input: The user enters the required information.
4. Validate the data: Check that all fields are filled in and that the information is correct, such as a valid email and a strong password.
   - If the data is invalid, return to the form so the user can correct it.
   - If the data is valid, continue to the next step.
5. Check whether the user already exists: Search for the same email address or username.
   - If the user already exists, show an error and return to the form so a different email address or username can be entered.
   - If the user does not exist, continue.
6. Store user data: Save the user's details in the database.
7. Display a success message: Show a message such as "Registration Successful."
8. End: Finish the registration process.

Q.2 Imagine that you are managing a project to develop a simple mobile application. Describe how you would use Agile methodology.

Answer: To develop a simple mobile application with Agile, I would use the following approach:

1. Break the project into small tasks: Divide the app into parts such as the login screen, sign-up page, user profile, and settings.
2. Work in short time periods called sprints: Plan sprints of about one or two weeks, with each sprint focused on selected features.
3. Hold short daily meetings: Team members briefly share what they completed, what they will do next, and what problems they have.
4. Get customer feedback after each sprint: Show the customer a working part of the app and use the feedback to make quick improvements.
5. Test while developing: Test each feature soon after it is created so that errors are found early.
6. Keep improving: At the end of every sprint, discuss what went well and what could be improved, then use the findings to improve teamwork and app quality.

Q.3 Consider an online banking system. Create a Use Case Diagram to show interactions between customers, bank staff, and the system.

Answer: A Use Case Diagram shows how different users, called actors, interact with system features. The following editable table represents the diagram clearly.

Actors and their tasks:
Customer — Add account; check balance; transfer money; request a loan.
Cashier — Update balance; view loan terms; process credit and debit transactions.
Manager — Approve loans; manage staff; handle promotions and salaries.

Bank system — Add account; check balance; transfer money; approve accounts; monitor transactions.

Connections:
1. Customer is connected to adding an account, checking the balance, transferring money, and requesting a loan.
2. Cashier is connected to updating the balance, viewing loan terms, and processing credit or debit transactions.
3. Manager is connected to approving loans, managing staff, and handling promotions and salaries.
4. The bank system is connected to adding accounts, checking balances, transferring money, approving accounts, and monitoring transactions.

Relationships:
• <<include>> is used when one task always requires another. For example, a transaction includes either a credit or a debit.
• <<extend>> is used when an extra task is performed only sometimes. For example, a loan process may be extended with checking interest when needed.

Q.4 You are developing a food-delivery application. Create a Sequence Diagram for placing an order, from selecting items to delivery.

Answer: Objects involved:
Customer — Starts the order.
Mobile App — Interface used to place the order.
Order Service — Processes and manages the order.
Restaurant — Prepares the food.
Delivery Service — Assigns a delivery person.
Driver — Delivers the order.

Sequence of interactions:
1. The customer browses the menu in the mobile app.
2. The customer selects items and sends the order to the mobile app.
3. The mobile app sends the order details to the order service.
4. The order service checks the order and sends confirmation to the mobile app.
5. The mobile app displays the confirmation to the customer.
6. The order service informs the restaurant about the order.
7. The restaurant confirms that it will prepare the order.
8. The order service requests delivery from the delivery service.
9. The delivery service assigns a driver and confirms the assignment.
10. The order service sends the delivery status to the mobile app.
11. The mobile app shows the delivery status to the customer.
12. The driver collects the order from the restaurant.
13. The driver delivers the order to the customer.
14. The customer confirms delivery through the mobile app.
15. The mobile app informs the order service that the order is complete.

Q.5 Discuss the importance of software development tools in the software development process.
a) Explain the role of language editors, translators, and debuggers.
b) Give examples and describe how they improve efficiency and accuracy.

Answer: Software development tools make the development process faster and more organised. They improve code quality, reduce errors, support teamwork, automate repeated tasks, maintain standards, and provide information about code performance. This allows developers to focus more on solving problems.

a) Roles of the tools:

Language editors
Role: Provide a place to write and edit code, with features such as syntax highlighting, auto-completion, and code formatting.
Contribution: Increase productivity, reduce typing mistakes, improve readability, and suggest improvements.

Translators (compilers and interpreters)
Role: Convert high-level code into instructions that a computer can understand. A compiler translates a complete program, while an interpreter translates and runs code line by line.
Contribution: Allow programs to run, identify syntax errors, and help improve program performance.

Debuggers
Role: Let developers run code step by step, inspect variables, and find run-time or logic errors.
Contribution: Make errors faster to solve, improve reliability, and reduce the chance of bugs reaching users.

b) Examples and contributions:

Visual Studio Code (VS Code) — A language editor that supports many languages, provides extensions for code analysis and formatting, and works with Git for version control and teamwork.

GCC (GNU Compiler Collection) — A translator that compiles C and C++ code into executable programs, gives useful error messages, and can improve performance.

GDB (GNU Debugger) — A debugger that supports breakpoints, variable inspection, and stack-trace analysis, helping developers locate and fix bugs quickly.

Editable Diagram References

The original exercise contains diagram images. The following tables provide editable text versions of the main diagram information so students can study, copy, and modify it in Word.

Online Banking Use Case Diagram

Actor	Connected use cases
Customer	Add Account; Check Balance; Transfer Money; Request Loan
Cashier	Update Balance; View Loan Terms; Process Credit/Debit Transactions
Manager	Approve Loan; Manage Staff; Handle Promotions and Salaries
Bank System	Add Account; Check Balance; Transfer Money; Approve Account; Monitor Transactions

Online Food-Delivery Sequence Diagram

Step	Sender / participant	Receiver / action
1	Customer → Mobile App	Browse the menu.
2	Customer → Mobile App	Select items and submit the order.
3	Mobile App → Order Service	Send the order details.
4	Order Service → Mobile App	Validate the order and send confirmation.
5	Order Service → Restaurant	Notify the restaurant of the order.
6	Order Service → Delivery Service	Request delivery.
7	Delivery Service → Order Service	Assign a driver and confirm the assignment.
8	Driver → Customer	Pick up and deliver the order.
9	Customer → Mobile App	Confirm delivery.
10	Mobile App → Order Service	Notify the service that the order is complete.