



COMSATS University, Islamabad
Department of Computer Science
Mid-Term Examination FALL-2024

Class/Section: BS(SE)

Total Marks: 25

Subject: Software Engineering Concepts (CSC-291)

Student Name: _____

Date: 4th-Nov-2024

Time Allowed: 80 Minutes

Instructors Name: Dr. Tehseen Riaz Abbasi

Registration No: _____

Please read the following instructions.

- Attempt all questions. All questions are self-explanatory and require no further explanations during exam time.
- Usage of Mobile phone, digital dairies, digital watch, or any electronic device is not allowed during exam.
- Use only Black/Blue color pen. Sharing of anything is forbidden.
- Students are required to have student ID card.
- **CUTTING/OVERWRITING** is not allowed and will be considered as an incorrect answer.
- **Attempt on ANSWER PAPER. Extra Sheet is not allowed.**

CLO-1

Explain the concept of software engineering along with its processes and deliverable.

Questions # 1:

(CLO - 1)

Marks: 10

1.1. State the fundamental software engineering activities? **Identify** the Essential attributes for a quality software? (1+1=2)

- Software specification
- Software development
- Software validation
- Software evolution.

Product characteristics	Description
Maintainability	Software should be written in such a way so that it can evolve to meet the changing needs of customers. This is a critical attribute because software change is an inevitable requirement of a changing business environment.
Dependability and security	Software dependability includes a range of characteristics including reliability, security, and safety. Dependable software should not cause physical or economic damage in the event of system failure. Malicious users should not be able to access or damage the system.
Efficiency	Software should not make wasteful use of system resources such as memory and processor cycles. Efficiency therefore includes responsiveness, processing time, memory utilization, etc.
Acceptability	Software must be acceptable to the type of users for which it is designed. This means that it must be understandable, usable, and compatible with other systems that they use.

1.2. Provide two software project examples that could be implemented using the incremental model. Be specific (2)

Each linear sequence produces deliverable "increments" of the software. Incremental development is particularly useful when staffing is unavailable for complete implementation by the business deadline that has been established for the project. **The incremental model, also known as the iterative or incremental development model**, is characterized by breaking down the software development process into smaller, manageable parts. Each iteration produces a partial but functional system, and new features or enhancements are added in subsequent iterations. This model allows for flexibility and the ability to respond to changing requirements. Here are two software project examples that could be implemented using the incremental model:

Content Management System (CMS) Development:

Increment 1: Develop the core functionality of user authentication, basic content creation, and page management.

Increment 2: Add advanced content features such as multimedia support, version control, and workflow management.

Increment 3: Integrate user permissions and access control, allowing different user roles with varying levels of authorization.

Increment 4: Enhance the user interface and user experience based on feedback from users and stakeholders.

Increment 5: Implement search functionality and optimize performance based on real-world usage data.

This incremental approach allows for the continuous delivery of a functional system after each iteration, and stakeholders can start using and providing feedback on the system early in the development process.

E-commerce Website Development:

Increment 1: Develop the basic online store with essential features such as product catalog, shopping cart, and order processing.

Increment 2: Integrate payment gateway functionality for secure online transactions.

Increment 3: Implement user accounts and personalized user experiences, including order history and recommendations.

Increment 4: Enhance the website's scalability to accommodate a growing number of products and users.

Increment 5: Introduce advanced features such as user reviews, promotional campaigns, and integration with social media platforms.

By following the incremental model, the e-commerce website can be delivered in functional increments, allowing the business to start selling products online early in the development process while continuously improving and expanding the system based on user feedback and market demands.

1.3. **Match** each description below to the SINGLE MOST APPROPRIATE TERM among the following.

(Note: Terms may apply to none, one, or more than one description.)

(0.5*4=2)

A. Boehm's spiral development	B. Throw-away prototyping	C. incremental development
D. Reuse-based development	E. RUP	F. Exploratory development
G. Extreme programming	H. Waterfall model	I. Scrum

1.3.1. **G** All developers are expected to refactor the code continuously.

1.3.2. **A** Explicitly incorporates risk assessment and resolution throughout the process.

1.3.3. **A** Limited software functionalities and expands functionalities in later software release.

1.3.4. **E** Normally described from 3 perspectives: dynamic (showing phases over time), static (showing process activities), and practice (suggesting good practices)

1.4. To which roles would you assign the following tasks?

(0.5*4=2)

1.4.1. Change a subsystem interface to accommodate a new requirement. **[system architect]**

1.4.2. Change the documentation as a result of the interface change. **[editor]**

1.4.3. Design a test suite to find defects introduced by the change. **[tester]**

1.4.4. Ensure that the change is completed on schedule. **[project manager or team leader]**

1.5. In Scrum, how can you ensure that the Daily Standup (Daily Scrum) meeting remains focused and effective, especially when team members tend to drift into unrelated discussions.

(2)

Here are some strategies to help ensure that the meeting stays focused:

1. Emphasize the Three Questions:

Reinforce the importance of answering the three standard questions (What did you do yesterday? What will you do today? Are there any impediments?) to keep team members focused on relevant updates.

2. Time-Box the Meeting:

Set a strict time limit for the Daily Scrum, typically around 15 minutes, to encourage brevity and discourage unrelated discussions.

3. Scrum Master Facilitation:

Have the Scrum Master play an active role in guiding the meeting, redirecting discussions that stray from the main points, and addressing off-topic issues separately after the meeting.

4. Continuous Improvement:

Use Sprint Retrospectives to discuss and implement improvements to the Daily Scrum process, fostering a culture of ongoing refinement and efficiency.

By implementing these strategies, you can help ensure that the Daily Scrum remains a quick, focused, and effective communication tool for the Scrum team.

CLO-2	Identify functional and non-functional requirements for a medium sized software system.
-------	---

Questions # 2: (CLO - 2) Marks: 10

2.1. Identify the following characteristic of requirement fall in which category of requirement (functional/Non-functional). (0.5*4= 2)

Statements	Category
2.2.1. The software system should be integrated with chat API	FR
2.2.2. The software automatically validates customers against the XYZ Contact Management System.	FR
2.2.3. The background color for all windows in the application will be blue and have a hexadecimal RGB color value of 0x0000FF	FR
2.2.4. Every unsuccessful attempt by a user to access an item of data shall be recorded on an audit trail	NFR

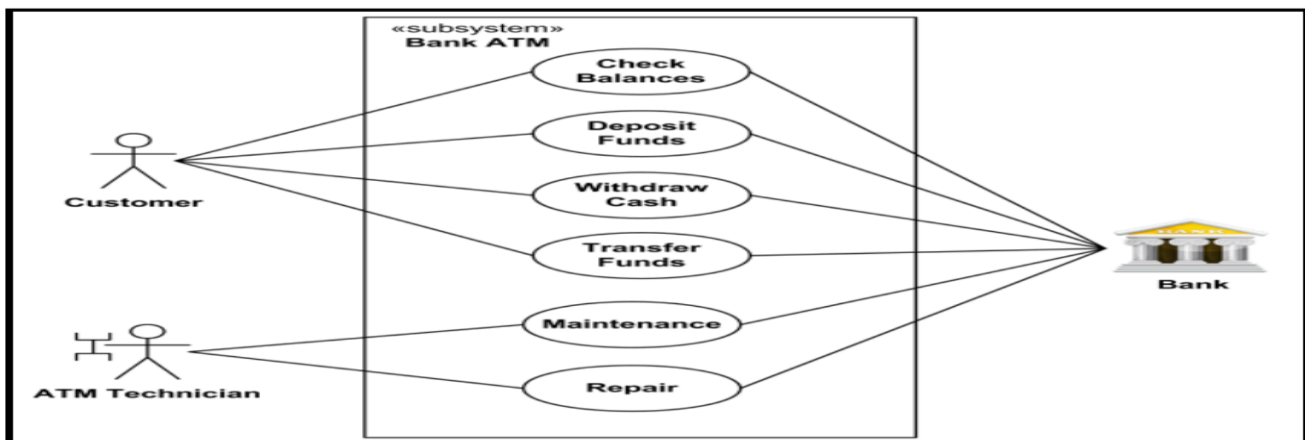
2.2. Below are examples of nonfunctional requirements. Specify which of these requirements are verifiable and which are not (0.5*4= 2)

- 2.2.1. "The availability of the system must be above 95%." [verifiable, assuming sufficient resources]
2.2.2. "The system must provide visual feedback to the user within 1 second of issuing a command." [verifiable]
2.2.3. "The software system must offer a user-friendly interface and adhere to usability standards." [not verifiable, no precise definition of "usable"]
2.2.4. "The user interface of the new system should be similar enough to the old system such that users familiar with the old system can be easily trained to use the new system." [not verifiable, no precise definition of "easily trained"]

2.3. Identify the primary and secondary actor and draw the Use Case Diagram for Scenario. (0.5+1.5=2)

Scenario: Draw a Use case diagram for the following scenario. Customer uses a bank ATM to check balances of his/her bank accounts, deposit funds, withdraw cash and/or transfer funds (use cases). ATM Technician provides maintenance and repairs to the ATM.

Primary Actor: Customer, ATM Technician
Secondary Actor: Bank



- 2.4. You are tasked with gathering functional requirements for a complex, real-time financial trading system. The system needs to handle a high volume of transactions and provide multiple user roles. **Describe the challenges** you might face when collecting these functional requirements and how you would prioritize and document them to ensure the system's success. **Additionally, discuss potential conflicts** that may arise between different stakeholders' requirements and how you would mitigate those conflicts. (1+1 = 2)

Challenges in Gathering Functional Requirements:

Complexity of Financial Markets:

The intricate nature of financial markets introduces challenges in understanding and documenting specific requirements, given the various instruments, trading strategies, and regulatory considerations.

Real-Time Processing:

Ensuring real-time processing introduces challenges related to latency, data synchronization, and system responsiveness, requiring a deep understanding of the desired level of speed and accuracy.

High Volume of Transactions:

Handling a high volume of transactions requires careful consideration of scalability, performance, and reliability to prevent system bottlenecks and ensure uninterrupted trading activities.

Regulatory Compliance:

Complying with financial regulations and industry standards adds complexity, as the system must adhere to specific rules, reporting requirements, and security measures to maintain legal and ethical standards.

Multiple User Roles:

Defining and accommodating the needs of various user roles, including traders, risk analysts, and administrators, introduces challenges related to access control, user interfaces, and role-specific functionalities.

Prioritization and Documentation:

Engage Stakeholders:

Collaborate with key stakeholders, including traders, risk managers, compliance officers, and IT experts, to understand diverse perspectives and prioritize requirements based on business value and criticality.

Use User Stories:

Break down requirements into user stories that reflect specific user interactions and system functionalities. Prioritize these stories based on their impact on core business processes and regulatory compliance.

Define Non-Functional Requirements:

Clearly document non-functional requirements related to performance, scalability, security, and regulatory compliance. These requirements are crucial for the overall success and stability of the financial trading system.

Iterative Development and Prototyping:

Adopt an iterative development approach, allowing for continuous feedback and refinement of requirements. Prototyping key functionalities can help stakeholders visualize the system and provide valuable insights for adjustments.

Risk Analysis:

Perform a risk analysis to identify potential challenges and dependencies early in the process. Prioritize requirements that address critical risks to mitigate the impact on project timelines and success.

Traceability Matrix:

Create a traceability matrix to link each requirement back to its source, ensuring that all business needs and regulatory requirements are covered. This matrix aids in validating that the delivered system aligns with the initially identified needs.

Validation and Verification:

Establish a comprehensive validation and verification process to confirm that the system meets the specified requirements. Conduct thorough testing, including functional, performance, and security testing, to ensure the system's reliability under various conditions.

By addressing these challenges and employing effective prioritization and documentation strategies, you can contribute to the successful development and deployment of a complex, real-time financial trading system that meets the needs of stakeholders and complies with industry standards.

Potential Conflicts Between Stakeholders' Requirements:

1. Performance vs. Cost:

Conflict: Traders want fast execution; IT focuses on cost.

Mitigation: Set clear benchmarks and cost constraints collaboratively.

2. User Experience vs. Security:

Conflict: Traders seek a streamlined interface; security wants strict controls.

Mitigation: Implement role-based access and balance security with usability.

3. Regulatory Compliance vs. Flexibility:

Conflict: Compliance requires strict adherence; traders seek flexibility.

Mitigation: Clearly define regulations and find areas for flexibility.

4. Scalability vs. Timelines:

Conflict: IT prioritizes scalability; business wants quick deployment.

Mitigation: Prioritize scalability initially and plan incremental improvements.

- 2.5. Summarize the significance of non-functional requirements (NFRs) and their impact on the design and performance of a software system. Also, provide specific examples of NFRs and how to quantify them.

Here are key points summarizing the significance of NFRs and their impact:

- Performance Optimization:** NFRs, such as response time and throughput requirements, drive design decisions to optimize the software's performance, ensuring efficient execution and responsiveness under various conditions.
- User Satisfaction:** Non-functional requirements, including usability and reliability criteria, directly impact the user experience. Meeting these requirements enhances user satisfaction by providing a system that is easy to use and consistently reliable.
- Scalability and Adaptability:** NFRs related to scalability influence the system's architecture, allowing it to adapt to changing workloads and user demands. This ensures the software can grow seamlessly without sacrificing performance.
- Security and Compliance:** Non-functional requirements pertaining to security guide the implementation of measures such as encryption and access controls, ensuring the software system is robust against threats and compliant with industry standards and regulations.
- Maintenance Efficiency:** NFRs focusing on maintainability and extensibility contribute to the long-term success of a software system. These requirements enable efficient maintenance and updates, reducing the risk of disruptions and facilitating future enhancements.

CLO-3

Construct appropriate design models for the structure and behavior of a medium sized software system.

Questions # 3:

(CLO - 3)

Marks: 5

3.1. Selected the most suitable architecture pattern with reasoning for given scenarios:•

Architecture Pattern Nominations are:

(2.5*2= 5 Marks)

- Client Server Architecture
- Layered Architecture
- Pipe and Filter Architecture
- Publish-Subscribe Architecture
- Three Tier Architecture

3.1.1. An enterprise web application is a web-based software system designed to meet the specific needs of businesses and organizations, providing tools for managing complex operations, data, and workflows.

3.1.1. Layered Architecture - Reasons - The main aim of the layered architecture is **to divide the design into small pieces**. This design **promotes modularity** as each layer performs only the tasks it is scheduled to perform. Easy debugging : As the layers are discrete so it is very easy to debug.

3.1.2. Why Design is important? What do you mean by design quality?

Software design is crucial in the software development process because it serves as the blueprint for building a successful and effective software system.

- **Why:**

- **Software design provides** a detailed plan and structure for implementing the intended functionality.
- **A solid design** contributes to the reliability and robustness of the software.
- **Design choices directly** impact the efficiency and performance of the software.
- **Software requirements often change**, and a good design allows the system to adapt to these changes.
- **Design documents serve as a communication tool** between different stakeholders, including developers, testers, project managers, and clients.

What is design quality?

- is to decompose “design quality” into quality attributes such as understandability, changeability, extensibility, reusability, testability, and reliability

***** Best of Luck *****