Quiz 3 Odd

Lecture 1: Software Requirement Engineering Overview

1. **Short Question**: What is the goal of software requirement engineering?
    - The goal is to develop quality software that meets customers' real needs, ensuring delivery on time and within budget.
2. **Short Question**: What is meant by the "total cost" of a software project across its life cycle?
    - The total cost includes the sum of all costs incurred from the inception of the project, through design, development, testing, deployment, maintenance, and eventual decommissioning.
3. **Long Question**: Explain how insufficient user involvement can lead to software project failure and suggest ways to mitigate this risk.
    - Insufficient user involvement can lead to project failure because developers might build a system that does not meet user needs. This can result in rework, higher costs, and dissatisfaction. To mitigate this risk, frequent stakeholder engagement, prototyping, and iterative development can ensure users' needs are properly understood.

Lecture 2: Software Requirements - Key Concepts

1. **Short Question**: What is the definition of a software requirement according to IEEE Std 729?
    - A software requirement is a condition or capability that must be met or possessed by a system to satisfy a contract, standard, or specification.
2. **Short Question**: Explain the difference between functional and non-functional requirements.
    - Functional requirements define what the system should do (e.g., allow users to log in), while non-functional requirements specify how the system should behave (e.g., performance, reliability, scalability).
3. **Long Question**: Design a functional and non-functional requirements document for a library management system.
    - **Functional Requirements**: The system shall allow users to search for books by title, author, and ISBN, borrow books, and return them.
    - **Non-functional Requirements**: The system must be available 99.9% of the time and support up to 500 simultaneous users with a response time of less than 3 seconds for searches.

Lecture 3: Requirements Engineering Process and Common Issues

1. **Short Question**: What are the common problems associated with requirements specified in natural language?
    - Ambiguity, inconsistency, and vagueness are common problems in natural language requirements. These can result in misinterpretation and incorrect system behavior.

2. **Short Question**: List of the key activities of the requirements engineering process.
    - The key activities are requirements elicitation, analysis, specification, and validation.
3. **Long Question**: Apply the requirements elicitation and validation process to a case study of your choice.
    - In a case study for a hospital management system, the requirements elicitation process could involve interviews with doctors, nurses, and administrators. The validation process might include user reviews and prototyping to ensure the system meets all user needs and works as expected.

Lecture 5: Problem Analysis and Root Cause Techniques

1. **Short Question**: What is problem analysis in the context of requirements engineering?
    - Problem analysis involves understanding the real-world issues that the system is supposed to address and ensuring that the software solution solves those problems.
2. **Short Question**: What is root cause analysis, and why is it important in software engineering?
    - Root cause analysis identifies the underlying causes of defects or issues in a software system. It helps in addressing the actual problems rather than just the symptoms.
3. **Long Question**: Design a root cause analysis for a software project where recurring bugs have been identified.
    - In a bug-ridden inventory management system, root cause analysis might involve analyzing why certain features (e.g., stock level updates) fail consistently. Investigation might reveal insufficient user testing or inadequate system resources. Solutions could include improving resource allocation and testing procedures.

Lecture 6: Business Modeling and System Engineering

1. **Short Question**: What is the purpose of business modeling in software engineering?
    - Business modeling helps in understanding the structure and dynamics of an organization to develop software that fits its needs.
2. **Short Question**: Why is system engineering crucial in embedded systems development?
    - System engineering is important in embedded systems because it ensures that all components (hardware and software) are integrated and function together efficiently.
3. **Long Question**: Design a business use-case model for a retail management system and explain how it transitions into a system model.

- A retail management system might have use cases such as "Process Customer Order" and "Update Inventory." The system model translates these into system requirements, defining how the software should support each function, such as connecting to payment gateways and inventory databases.

Lecture 7: Unified Modeling Language (UML)

1. **Short Question**: What are the key aims of modeling in software development?
   - Modeling helps visualize the system, specify its structure and behavior, guide the construction of the system, and document decisions.
2. **Short Question**: What is the difference between structural and behavioral UML diagrams?
   - Structural diagrams represent the static aspects of a system, like class and component diagrams, while behavioral diagrams capture dynamic behaviors, such as sequence and use case diagrams】.
3. **Long Question**: Design a complete UML diagram (use-case, class, and sequence diagrams) for an e-commerce system.
   - **Use-case diagram**: "Place Order" (actor: customer), "Process Payment" (actor: payment gateway).
   - **Class diagram**: Classes like Order, Payment, Customer, with relationships between them.
   - **Sequence diagram**: A customer selects items, places an order, and the system processes the payment. Each diagram helps define different aspects of the system and how components interact.

Lecture 10: Requirements Errors and Defects

1. **Short Question**: What are the types of errors found in software requirements?
   - Errors of omission, commission, clarity, ambiguity, and performance.
2. **Short Question**: What is an error of omission in requirements documents?
   - An error of omission occurs when a necessary requirement is not specified.
3. **Long Question**: Apply defect prevention techniques to a software project of your choice.
   - In a ticket booking system, defect prevention techniques such as Joint Application Development (JAD) and prototyping could be applied during requirements gathering. This helps avoid miscommunication, thus reducing defects.

Lecture 4: Introduction to Requirements Management

Short Questions:

1. **What is the difference between the problem domain and the solution domain in software engineering?**

- **Answer**: The problem domain refers to the real-world environment and the issues that the system is intended to address. It includes understanding the user needs, constraints, and business processes. The solution domain, on the other hand, focuses on how the software system will be developed to solve the problem identified in the problem domain. It deals with the technical aspects like system architecture, design, and implementation(04-Introduction To SRE …)(10-Introduction To SRE …).

2. **What are the key features of the HOLIS software development team?**

    - **Answer**: The HOLIS team, working on the Home Lighting Automation System, is composed of 15 members, mostly new hires with a few transferred from the Commercial Lighting Division. The team is cross-functional and focuses on understanding stakeholder needs and developing features that align with those needs(04-Introduction To SRE …).

Long Question:

1. **Design a requirements management strategy for a software project that integrates both the problem domain and the solution domain.**

    - **Answer**: A requirements management strategy should begin with identifying and understanding the problem domain by interviewing stakeholders, conducting workshops, and analyzing existing systems. The team must prioritize user needs and business goals. Once the problem domain is well understood, the transition into the solution domain involves defining the technical requirements, system architecture, and design. A well-structured approach, using tools like use-case diagrams and requirement traceability matrices, ensures alignment between the problem domain and the technical solution. Additionally, iterative validation and stakeholder reviews should be conducted throughout the development lifecycle to ensure the system continues to meet the original needs identified in the problem domain(04-Introduction To SRE …).

Lecture 8: Requirements Elicitation and User Needs

Short Questions:

1. **What are the three common syndromes encountered during requirements elicitation?**

    - **Answer**: The three common syndromes are:

1. **Yes, But** Syndrome: This occurs when users are not sure of their needs until they see an initial version of the system and then realize additional requirements or problems.

2. **Undiscovered Ruins** Syndrome: As more requirements are uncovered, it becomes apparent that even more requirements are still undiscovered.

3. **User and Developer** Syndrome: This reflects the communication gap between users and developers, as they often come from different backgrounds and may not fully understand each other's needs or limitations(08-Introduction To SRE …)(09-Introduction To SRE …).

2. **What is the relationship between stakeholder needs and system features?**

    - **Answer**: Stakeholder needs represent the real-world problems or opportunities that the system aims to address, while system features are the services or functionalities the system provides to fulfill those needs. Each feature of the system should directly map to one or more stakeholder needs, ensuring that the system aligns with user expectations and solves the identified problems(08-Introduction To SRE …).

Long Question:

1. **Apply the techniques of brainstorming and idea reduction to address the "Undiscovered Ruins" syndrome during the requirements elicitation process.**

    - **Answer**: In **brainstorming sessions**, all stakeholders should be encouraged to freely express their ideas about potential requirements, even those that might seem unrelated. By creating a **comprehensive list** of requirements, the team ensures that they cover all possible areas. After brainstorming, idea reduction is applied to **prioritize** the most critical and feasible requirements, using techniques such as voting or a decision matrix. This process helps manage the "**Undiscovered Ruins**" syndrome by bringing potential requirements to light early on and **focusing** on the most relevant ones to address the project's goals(08-Introduction To SRE …).

Lecture 9: Requirements Analysis and Negotiation

Short Questions:

1. **What is an interaction matrix and how is it used in requirements analysis?**

    - **Answer**: An interaction matrix is a tool used to discover and analyze the interactions between different requirements. It shows how each

requirement relates to others—whether they are independent, overlap, or conflict. This helps in identifying potential areas of inconsistency or redundancy in the requirements, making it easier to resolve conflicts or streamline overlapping requirements(09-Introduction To SRE …).

2. **Explain the significance of resolving conflicting requirements during the negotiation process.**

    - **Answer**: Resolving conflicting requirements is essential because unresolved conflicts can lead to incompatible system behaviors or unmet stakeholder needs. During the negotiation process, stakeholders must reach a compromise to ensure that the final set of requirements is cohesive and meets the overall project goals. Failure to resolve conflicts early can lead to costly rework and project delays later in the development cycle(09-Introduction To SRE …)(10-Introduction To SRE …).

Long Question:

1. **Apply the stages of negotiation meetings (information, discussion, resolution) to resolve conflicts in a distributed system's requirements.**

    - **Answer**:

        1. **Information Stage**: In the information stage, all stakeholders provide their input on the requirements and share their perspectives. For a distributed system, this could include technical teams discussing bandwidth limitations while business stakeholders emphasize the need for real-time data synchronization.

        2. **Discussion Stage**: During the discussion stage, the stakeholders engage in a dialogue to address the conflicts. For example, a compromise might be reached where real-time synchronization is applied to critical data only, while less critical data is updated in batches.

        3. **Resolution Stage**: In the resolution stage, a formal agreement is reached, and changes are made to the requirements based on the compromises made. The revised requirements are documented, and all stakeholders sign off on the final version(09-Introduction To SRE …)(10-Introduction To SRE …).

Quiz 3 Even

Lecture 1: Software Requirement Engineering Overview

1. **Short Question**: Why is it important for a software system to meet stakeholders' expectations?
    - Meeting stakeholders' expectations ensures the software is usable and satisfies the needs of its users, which increases adoption and reduces the risk of failure.
2. **Short Question**: Identify the root causes of project failure according to the Standish Group.
    - The root causes include lack of user input, incomplete or changing requirements, and technical skill gaps.
3. **Long Question**: Apply the principles of requirements engineering to a real-world example where incomplete requirements led to project delays or failures.
    - In a real-world project like a government payroll system, incomplete requirements might result in delays due to constant revisions. Applying structured elicitation methods and validation (e.g., prototyping) could prevent such issues.

Lecture 2: Software Requirements - Key Concepts

1. **Short Question**: List of the different sources of software requirements.
    - Sources include stakeholders, domain experts, existing systems, regulations, and business documents.
2. **Short Question**: What are examples of abstract statements of services in software requirements?
    - Abstract statements include vague descriptions such as "The system shall allow users to manage accounts" without specifying how.
3. **Long Question**: Explain how software requirements should be documented to ensure clarity and avoid misinterpretation.
    - Software requirements should be written clearly with examples, diagrams, and detailed use cases to prevent misinterpretation. Requirements should be validated with stakeholders to ensure alignment.

Lecture 3: Requirements Engineering Process and Common Issues

1. **Short Question**: Why is it expensive to make changes to requirements after they have been agreed upon?
    - Changes after agreement can require rework, affecting design, code, and tests, leading to additional time and cost.
2. **Short Question**: What is the impact of wrong requirements on a software project's success?
    - Wrong requirements lead to software that does not meet user needs, which may result in project failure, financial loss, and reputation damage.
3. **Long Question**: Explain the role of process models in software development.
    - Process models help structure activities in software development, from requirements gathering to testing, ensuring consistent outcomes. They help improve understanding of how tasks should be completed and ensure the development process is transparent.

Lecture 5: Problem Analysis and Root Cause Techniques

1. **Short Question**: List the steps to gain agreement on the problem definition.
    - Steps include brainstorming, discussing with stakeholders, writing down the problem, and ensuring all stakeholders agree on the definition.
2. **Short Question**: Explain how brainstorming can be used to identify root causes.
    - Brainstorming allows teams to openly discuss and generate ideas about potential causes of a problem, which can then be analyzed to identify the root cause.
3. **Long Question**: Explain the importance of identifying and addressing constraints early in the requirements engineering process.
    - Identifying constraints early ensures that the system design accounts for limitations such as budget, time, technology, and user capabilities. Addressing these constraints early helps avoid costly rework.

Lecture 6: Business Modeling and System Engineering

1. **Short Question**: Explain the difference between a business use-case model and a business object model.
    - A business use-case model focuses on the functional activities of the business, while a business object model defines the entities (objects) involved in the business and their interactions.

2. **Short Question**: What is the role of system engineers in decomposing complex systems?
    - System engineers decompose complex systems into manageable subsystems, ensuring each component functions properly and integrates well with others.
3. **Long Question**: Explain the principles of systems engineering and how they contribute to the development of complex software solutions.
    - Systems engineering ensures that all components (hardware, software, processes) are integrated and function cohesively. It emphasizes a holistic approach, optimizing both technical and business aspects to develop reliable systems.

Lecture 7: Unified Modeling Language (UML)

1. **Short Question**: Explain the significance of UML in object-oriented software development.
    - UML provides a standardized way to visualize system architecture and behavior, which helps in designing, implementing, and maintaining object-oriented systems.
2. **Short Question**: What is an example of a class diagram in UML?
    - A class diagram for a library system may include classes such as Book, Member, and Loan, with relationships like borrowing and returning books.
3. **Long Question**: Explain how behavioral and structural diagrams in UML work together to model both the static and dynamic aspects of a software system.
    - Structural diagrams, such as class diagrams, show the static architecture of the system. Behavioral diagrams, like sequence diagrams, depict the flow of operations. Together, they provide a complete picture of the system's functionality.

Lecture 10: Requirements Errors and Defects

1. **Short Question**: Why is preventing requirements errors more effective than removing them later?
    - Preventing errors early avoids costly fixes downstream, which can occur during coding or testing phases. Prevention is more efficient than correction.
2. **Short Question**: Explain the impact of performance errors on software systems.

- Performance errors, such as slow response times or system crashes, degrade user experience and can lead to system failure under heavy loads.

3. **Long Question**: Explain how requirements inspections can be used to prevent errors in requirements documentation.

    - Requirements inspections involve reviewing documents with stakeholders and experts to identify ambiguities, omissions, or misunderstandings. This process helps ensure clarity and completeness before development begins, improving project outcomes.

Lecture 4: Introduction to Requirements Management

1. **Short Question**: List the formal processes used in requirements management such as CMM and ISO 9000.

    - Formal processes include the Capability Maturity Model (CMM) and ISO 9000, which provide frameworks for improving software development processes.

2. **Short Question**: Explain the importance of understanding stakeholder needs during the requirements process.

    - Understanding stakeholder needs ensures that the final product meets user expectations, reducing the risk of rework and failure.

3. **Long Question**: Apply the six team skills needed for effective requirements management to a case study of a home automation system.

    - In a home automation system, applying skills such as problem analysis, understanding user needs, managing scope, and refining requirements helps ensure that the system meets all stakeholder expectations and functions properly.

Lecture 8: Requirements Elicitation and User Needs

1. **Short Question**: Explain the purpose of the "Yes, But" syndrome in requirements gathering.

    - The "Yes, But" syndrome occurs when users see a system and begin to realize additional requirements or changes. Addressing it early in the process helps avoid scope creep.

2. **Short Question**: Why is interviewing considered a key technique in requirements elicitation?

    - Interviewing allows direct interaction with stakeholders to gather detailed requirements and uncover any hidden needs or constraints.

3. **Long Question**: Design an effective requirements elicitation process for a new inventory management system.

    - The process should include interviews with key stakeholders (e.g., warehouse managers, clerks), workshops to gather feedback, and brainstorming sessions to ensure all potential requirements are covered. Prototyping and validation should follow to ensure the accuracy of gathered requirements.

Lecture 9: Requirements Analysis and Negotiation

1. **Short Question**: List the stages of requirements negotiation.

    - The stages include requirements discussion, prioritization, and agreement.

2. **Short Question**: Why is it important to prioritize requirements during the negotiation phase?

    - Prioritizing helps identify critical requirements that must be implemented first and ensures that conflicts between stakeholders are resolved efficiently.

3. **Long Question**: Design a requirements interaction matrix for a project of your choice and explain how it helps in identifying overlaps and conflicts among requirements.

    - In a project to build a university grading system, the interaction matrix might compare requirements such as "Calculate Final Grades" and "Display Grades to Students." Overlaps and conflicts between these requirements could be identified, such as data handling issues or different user access levels, helping avoid inconsistencies.