Lecture Title: Chapter 6: Requirements Modeling: Scenarios, Information, and Analysis Classes
Subject: Software Engineering
Program: BTech Computer Science and Engineering
Duration: 1 Hour

I. LECTURE INTRODUCTION & OBJECTIVES

Opening Hook: "Last lecture, we learned how to gather requirements. Now we face the critical question: How do we transform a messy list of stakeholder wishes into a precise, analyzable blueprint? A blueprint that developers can build from and testers can validate? The answer is Requirements Modeling—creating abstract representations that help us understand, communicate, and validate what we need to build. Today, we focus on three key modeling perspectives: Scenarios (Use Cases), Information (Data Models), and Analysis Classes."

Objectives: By the end of this lecture, you will be able to:

1. Explain the objectives and philosophy of requirements analysis.
2. Create and refine use cases at different levels of formality.
3. Use UML activity and swimlane diagrams to supplement use cases.
4. Apply core data modeling concepts: objects, attributes, and relationships.
5. Understand how these models work together to represent requirements.

II. PART 1: THE FOUNDATIONS OF REQUIREMENTS ANALYSIS

6.1 Requirements Analysis

- Definition: The process of translating user needs (often vague) into detailed system specifications (precise and actionable).
- It bridges the gap between *what the customer wants* and *what the developer builds*.

6.1.1 Overall Objectives and Philosophy

- Primary Goal: To describe what the system must do, not how it will do it. (How = Design).
- Key Objectives:
    1. Discover & Clarify: Uncover ambiguities, contradictions, and omissions.
    2. Categorize: Separate functional from non-functional requirements.
    3. Prioritize: Identify what's essential for the first release.
    4. Model: Create visual and textual representations.
    5. Create a Baseline: Establish a validated specification that will guide design.

6.1.2 Analysis Rules of Thumb

Practical guidelines for analysts:

1. Focus on the Problem Domain: Understand the business context, not jump to technical solutions.
2. Create Models That Can Be Understood by Non-Technical People. Use diagrams and simple language.
3. Strive for Consistency and Completeness within the models.
4. Move from Essential Information to Implementation Detail Gradually. Start abstract, then refine.

6.1.3 Domain Analysis

- Definition: Identifying, analyzing, and specifying common requirements from a specific application domain (e.g., e-commerce, banking, healthcare) *before* project requirements are defined.
- Goal: To discover analysis patterns and reusable knowledge.
- Process: Collect domain samples → Identify commonalities and variabilities → Develop an analysis model for the domain.
- Benefit: Reduces reinvention, speeds up the analysis phase of new projects in the same domain.

6.1.4 Requirements Modeling Approaches

- Three complementary modeling views form the core of the Requirements Model:
    1. Scenario-Based Modeling (Today's Focus): Represents the system from the user's point of view. (Use Cases)

2. Data Modeling (Today's Focus): Represents the information domain of the system. (What data is stored/manipulated?)
3. Class-Based Modeling (Next Chapter): Represents objects, their attributes, and operations.
4. Flow-Oriented Modeling: Represents how data transforms as it flows through the system (e.g., Data Flow Diagrams).

- A good model uses a combination of these views.

III. PART 2: SCENARIO-BASED MODELING - THE USER'S STORY

- Premise: The best way to understand system requirements is to see how specific actors use the system to achieve specific goals.

6.2.1 Creating a Preliminary Use Case (Informal)

- Use Case: A description of one specific interaction between an actor and the system to achieve a goal.
- Actor: A role played by an external entity (person, hardware, other system) that interacts with the system.
- Steps to Create:
    - Identify Actors: "Who/what uses the system?" (e.g., Student, Lecturer, Billing System).
    - Identify Major Use Cases: "What goals do they have?" (e.g., "Register for Course," "Submit Grade").
    - Write a Brief Description: 2-3 sentences outlining the main success scenario.
- Example (Library System):
    - Actor: Member
    - Use Case: Borrow Book
    - Brief Description: A member presents a book at the counter. The librarian scans the member ID and book barcode. The system records the loan and due date. The member takes the book.

6.2.2 Refining a Preliminary Use Case (Expanded Narrative)

- Add more detail, but keep it in paragraph/narrative form.
- Include:
    - Actors
    - Preconditions: What must be true before the use case starts? (e.g., "Member is registered and in good standing.")
    - Trigger: What starts the use case? (e.g., "Member presents book for checkout.")
    - Main Success Scenario (Basic Flow): The step-by-step "happy path" where everything goes right.
    - Extensions (Alternate/Exception Flows): What can go differently? (e.g., "Book is reserved," "Member has overdue fines.")
    - Postconditions: State of the system after use case completes. (e.g., "Loan is recorded, book status is 'checked out'.")

6.2.3 Writing a Formal Use Case (Fully Detailed)

- Uses a structured, tabular or numbered template. Crucial for complex or safety-critical systems.
- Template Includes:
    - Use Case ID & Name
    - Primary Actor & Stakeholders
    - Preconditions, Postconditions
    - Flow of Events:
        - Actor Action | System Response (Two-column format)
    - Alternative Flows (Referenced by step number, e.g., "3a. If the book is reserved...")

In-Class Exercise (5 min): In groups, take the "Borrow Book" preliminary use case and write Step 1 of the main success scenario in the two-column formal format.

- Actor Action: Librarian scans member ID card.
- System Response: System validates member status (active, no blocks) and displays member name.

6.3 UML Models That Supplement the Use Case

6.3.1 Developing an Activity Diagram

- A flowchart that models the workflow or business process involved in a use case or across multiple use cases.
- Elements:
    - Start & End Nodes (filled circle, bulls-eye).
    - Activities (rounded rectangles) - "Scan book," "Calculate due date."
    - Decisions/Merges (diamonds) - For alternate flows.
    - Control Flow Arrows.
- Use: Excellent for showing parallel activities and complex logic that is hard to describe textually.

6.3.2 Swimlane Diagrams

- A special type of activity diagram where lanes (like swimming pool lanes) partition activities based on who performs them (Actor or System).
- Columns/Lanes: "Member," "Librarian," "Library System."
- Benefit: Clearly shows responsibility and interaction between actors and the system. Very useful for identifying system boundaries.

IV. PART 3: DATA MODELING - THE INFORMATION BACKBONE

- Answers: What information does the system need to remember?

6.4.1 Data Objects

- A Data Object is a representation of a composite information item that the system must know about.
- It can be a person, place, thing, concept, or event.
- Examples: `Customer`, `Order`, `Product`, `Invoice`, `Reservation`.
- Represented as a noun.

6.4.2 Data Attributes

- Attributes define the properties of a data object.

- They name a data object characteristic and have a value.
- Examples: For `Customer` object -> `customerID`, `name`, `email`, `address`.
- Identifier (Key): One or more attributes that uniquely identify an instance of the object (e.g., `customerID`).

6.4.3 Relationships

- A Relationship indicates a connection between data objects.
- Cardinality/Multiplicity: Defines the number of occurrences of one object that can relate to the number of occurrences of another.
    - 1:1 (One-to-One): One `Student` has one `Student ID Card`.
    - 1:M (One-to-Many): One `Library` has many `Books`.
    - M:N (Many-to-Many): Many `Students` can register for many `Courses`. (This often requires resolution into two 1:M relationships via an associative entity).
- Modeling: Shown as a line connecting objects, with cardinality notation (1, *, M, 0..1, etc.).
- Example ERD (Entity-Relationship Diagram) on board: `Student` ---`<registers for>`--- `Course`. Cardinality: `Student` (1..*) to* `Course` *(0..*).

V. CONCLUSION & KEY TAKEAWAYS

1. Requirements Analysis is the disciplined process of transforming vague needs into a precise, actionable specification. It focuses on what, not how.
2. Scenario-Based Modeling (Use Cases) captures functional requirements from the user's perspective at varying levels of detail (Brief, Expanded, Formal).
3. UML Activity & Swimlane Diagrams visually supplement use cases, showing workflow and responsibility.
4. Data Modeling defines the system's information backbone through Objects (nouns), their Attributes, and the Relationships between them.
5. These models are complementary. A Use Case (scenario) will manipulate the Data Objects defined in the data model.

Final Thought: "A model is a lie that helps you see the truth. These requirements models are simplified, abstract lies about the future system. But they are essential lies that

allow stakeholders, analysts, and developers to see and agree on the truth of what needs to be built before a single line of code is written."

Suggested Reading:

- *Applying UML and Patterns* by Craig Larman – Excellent for use-case driven analysis and design.
- UML 2.0 Specification for activity diagram syntax.