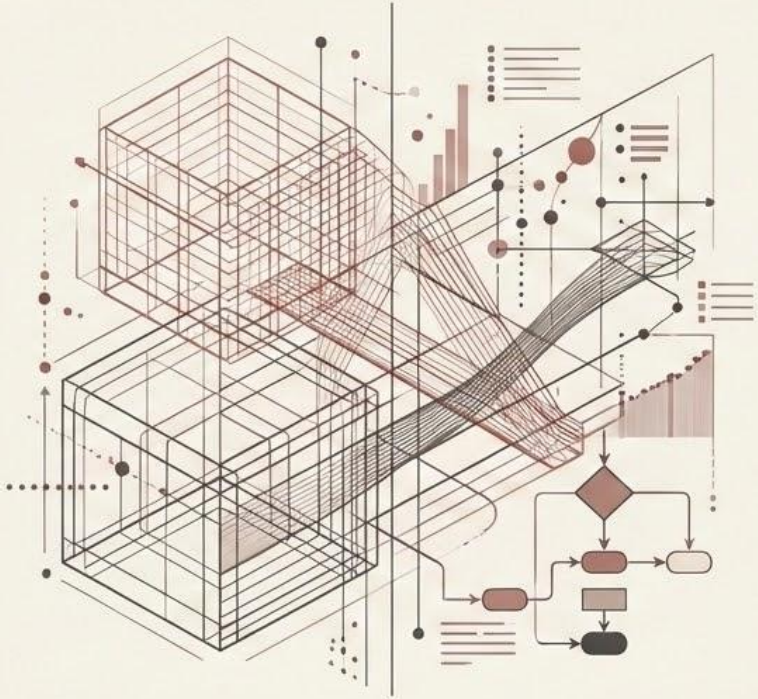
Chapter 6: Requirements Modeling

Scenarios, Information, and Analysis Classes

Duration: 1 Hour



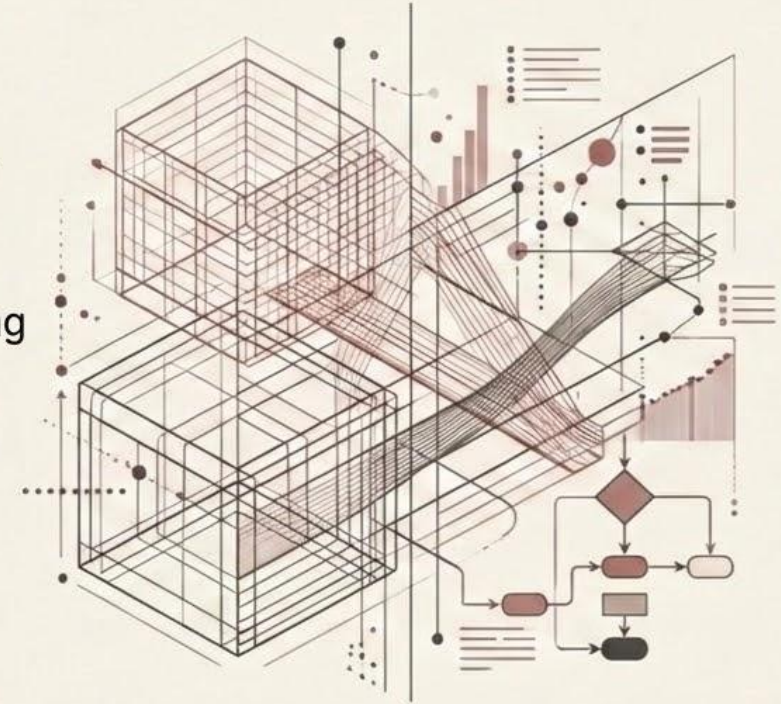
Introduction

The Critical Question

- **Challenge:** How do we transform a messy list of stakeholder wishes into a precise, analyzable blueprint?
- **Solution:** Requirements Modeling—creating abstract representations to understand, communicate, and validate needs.

Focus Areas:

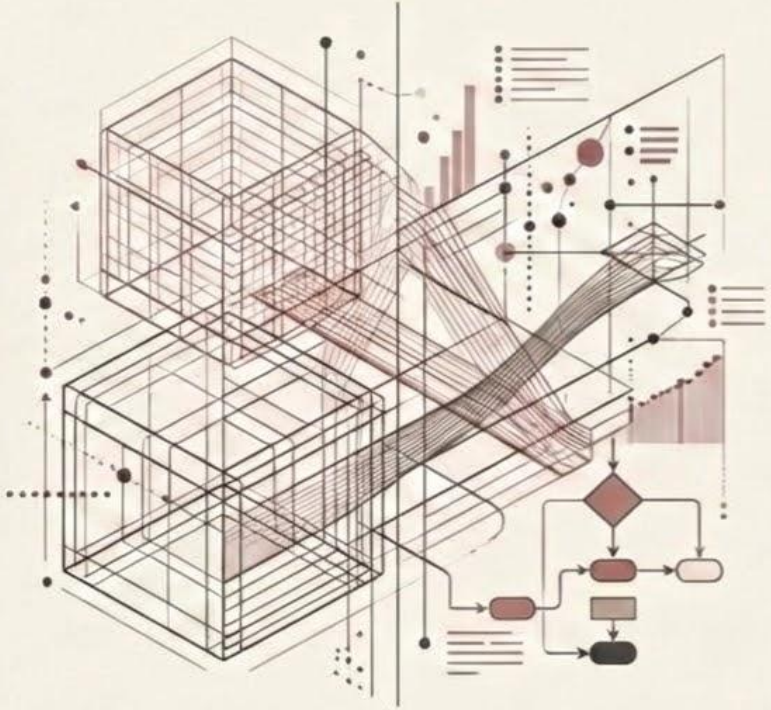
- Scenarios (Use Cases)
- Information (Data Models)
- Analysis Classes



Learning Objectives

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

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

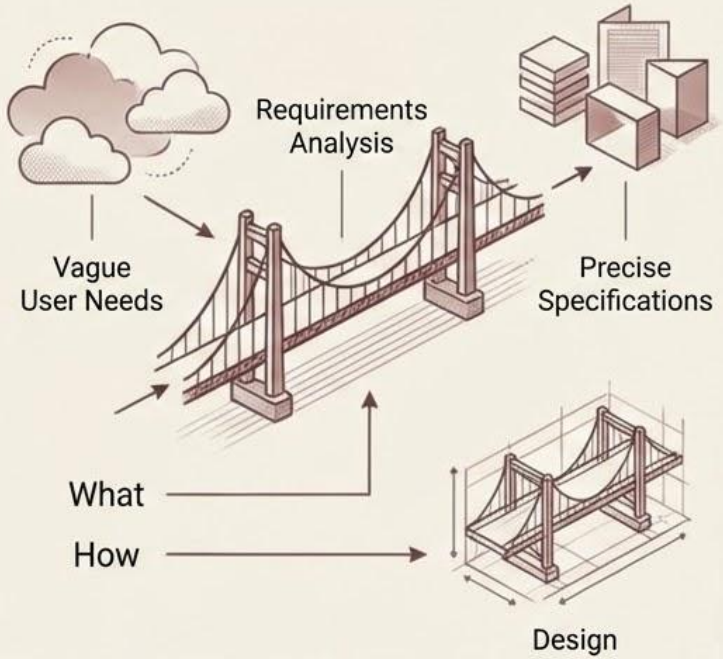


Part 1: The Foundations

What is Requirements Analysis?

- **Definition:** The process of translating vague user needs into precise, actionable system specifications.
- **The Bridge:** It connects what the customer wants with what the developer builds.
- **Primary Goal:** To describe what the system must do, not how it will do it.

What = Analysis
How = Design



Key Analysis Objectives

- **Discover & Clarify:** Uncover ambiguities, contradictions, and omissions.

- **Categorize:** Separate functional from non-functional requirements.
- **Prioritize:** Identify what is essential for the first release.
- **Model:** Create visual and textual representations.
- **Create a Baseline:** Establish a validated specification to guide design.

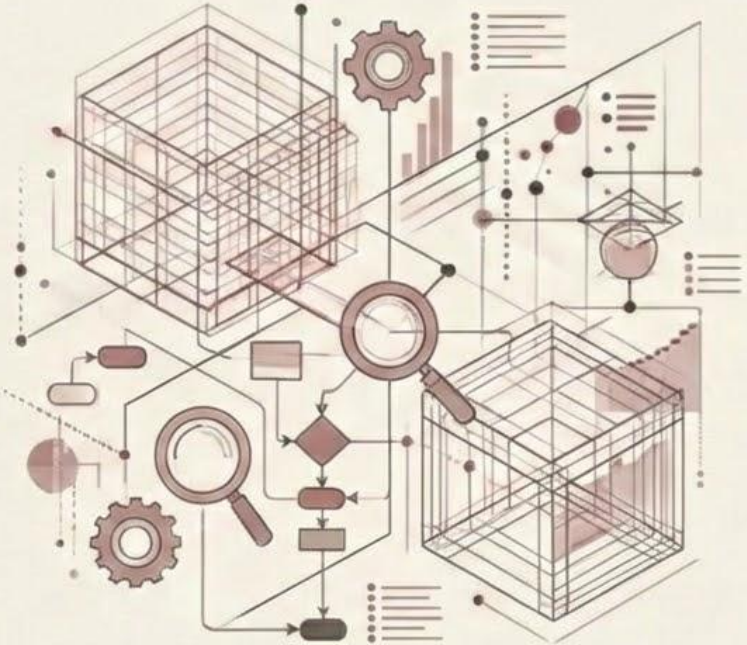
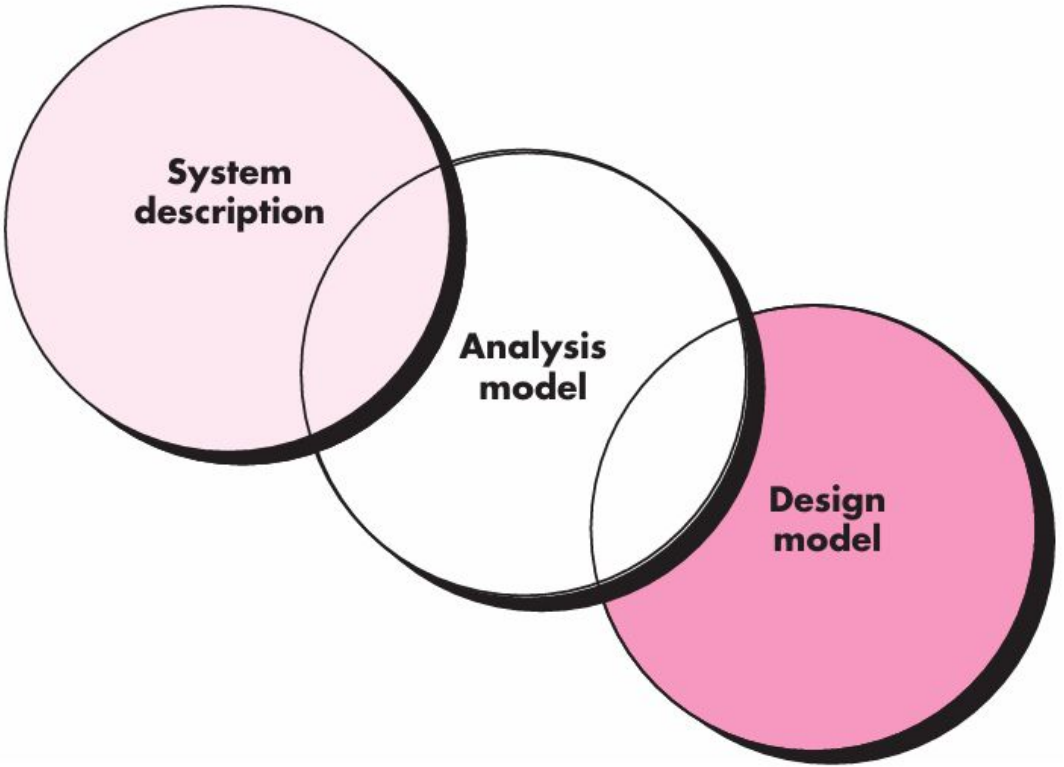


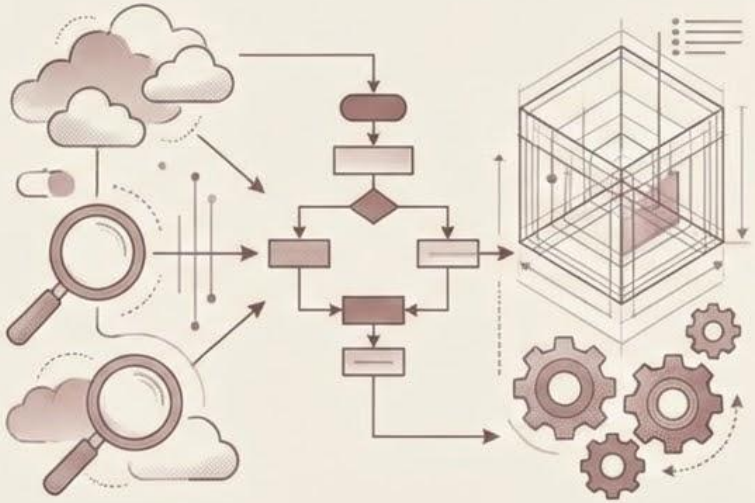
FIGURE 6.1

The requirements model as a bridge between the system description and the design model



Rules of Thumb for Analysts

- **Focus on the Problem Domain:** Understand the business context rather than jumping to technical solutions.
- **Accessible Models:** Create models understandable by non-technical people using simple language and diagrams.
- **Consistency:** Strive for consistency and completeness within the models.
- **Gradual Detail:** Move from essential information to implementation details gradually (start abstract, then refine).

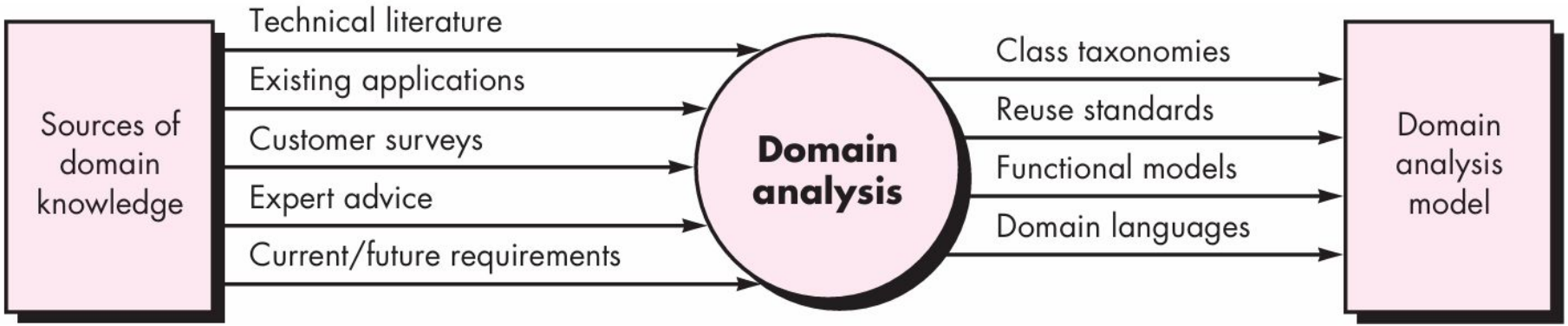


Domain Analysis

- **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.

- **Benefit:** Reduces reinvention, speeds up the analysis phase of new projects in the same domain.

FIGURE 6.2 Input and output for domain analysis



Requirements Modeling Approaches

- **Scenario-Based Modeling:** Represents the system from the user's point of view (Use Cases).
- **Data Modeling:** Represents the information domain (data stored and manipulated).
- **Class-Based Modeling:** Represents objects, attributes, and operations.
- **Flow-Oriented Modeling:** Represents how data transforms as it flows through the system.

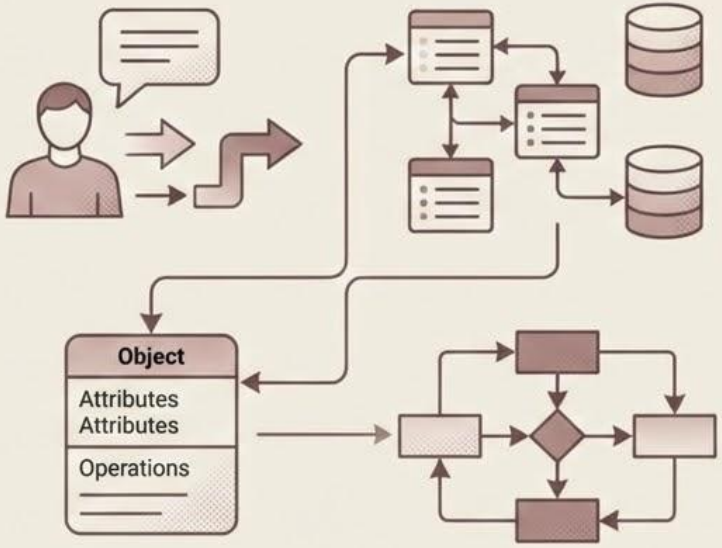
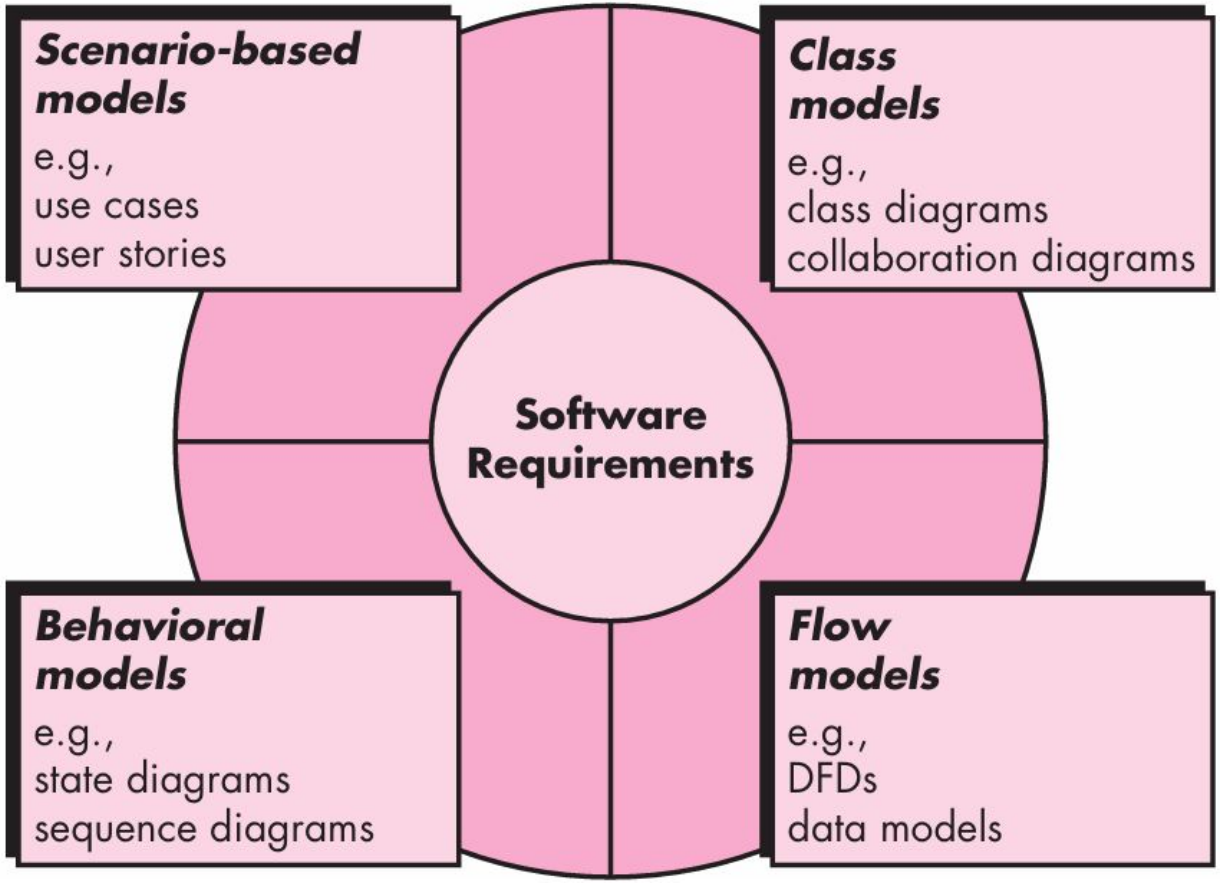


FIGURE 6.3

Elements of the analysis model



? What different points of view can be used to describe the requirements model?

Part 2: Scenario-Based Modeling

The User's Story

- **Premise:** Understand requirements by seeing how specific actors use the system to achieve goals.
- **Use Case:** A description of a specific interaction between an actor and the system to achieve a goal.
- **Actor**: A role played by an external entity (person, hardware, system).

Creation Steps:

1. Identify Actors (Who uses the system?).
2. Identify Major Use Cases (What are their goals?).
3. Write a Brief Description (Main success scenario).

Refining Use Cases

From Preliminary to Expanded Narrative

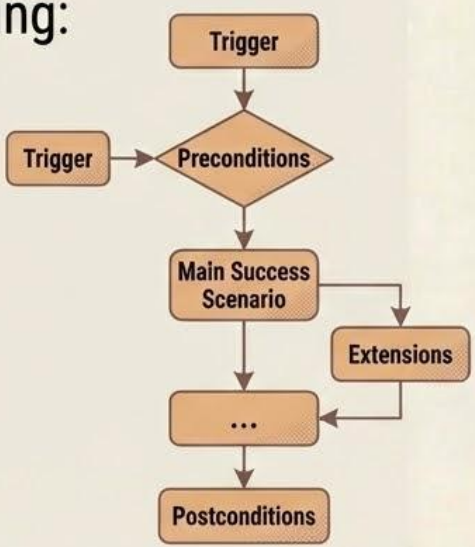
Preliminary

Brief description of the main success scenario.

Expanded Narrative

Adds detail in paragraph form, including:

- Preconditions: What must be true before starting (e.g., Member is registered).
- Trigger: What starts the use case?
- Main Success Scenario: The “happy path” where everything goes right.
- Extensions: Alternate flows or exceptions (e.g., Book is reserved).
- Postconditions: State of the system after completion.



Writing a Formal Use Case

Format: Structured, tabular, or numbered template.

Usage: Crucial for complex or safety-critical systems.

Key Template Elements:

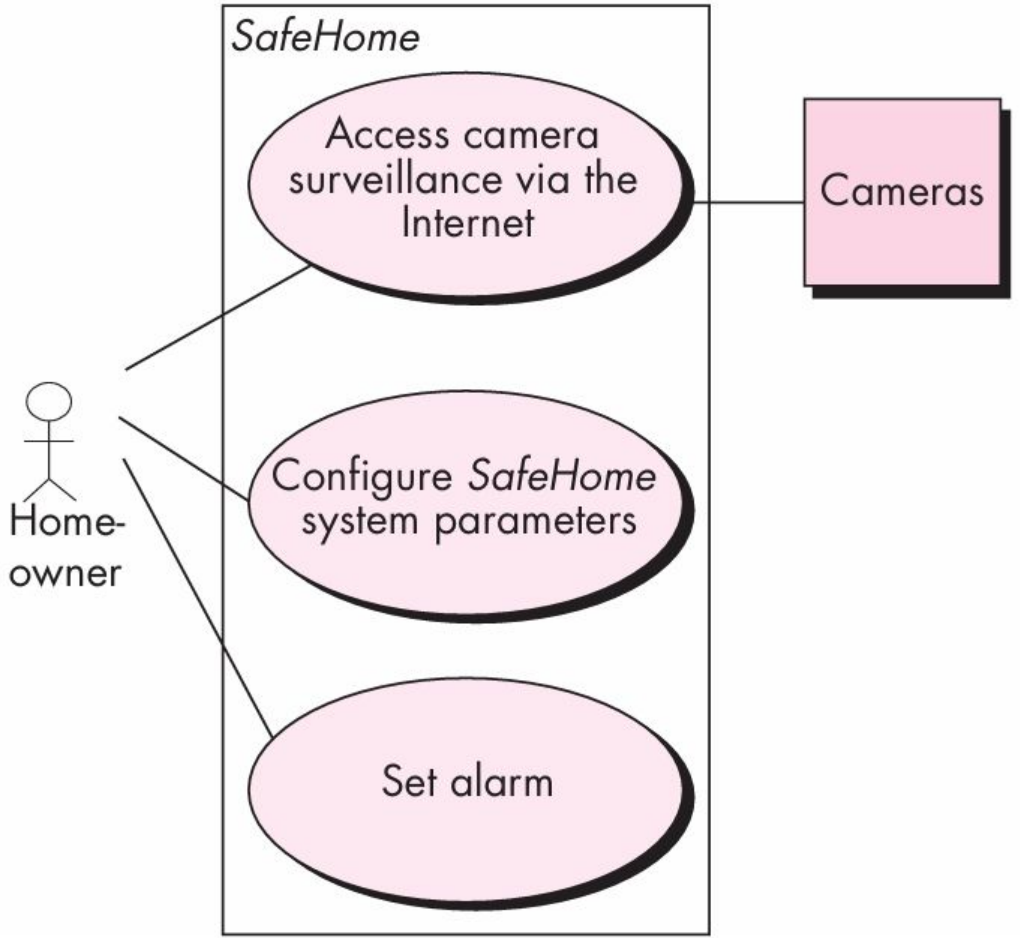
- Use Case ID & Name
- Primary Actor
- Preconditions
- Postconditions

Flow of Events: Two-column format (Actor Action | System Response). **Alternative Flows** (referenced by step number).

Use Case ID:	Use Case ID:	
Name:	Name:	
Primary Actor:	Primary Actor:	
Preconditions:	Preconditions:	
Postconditions:	Postconditions:	
	Actor Action	**System Response**
Flow of Events	1. [Action]	[Response]
	2. ...	...
	3. ...	...
Alternative Flows →	...	
	4. ...	...

FIGURE 6.4

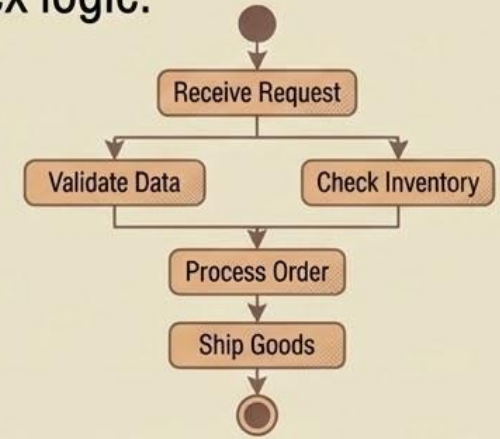
Preliminary use-case diagram for the *SafeHome* system



UML Activity & Swimlane Diagrams

Supplementing the Use Case

Activity Diagram
A flowchart modeling the workflow or business process.
Good for: Parallel activities and complex logic.



Swimlane Diagram
Partitions activities into lanes based on who performs them (Actor vs. System).
Benefit: Clearly shows responsibility and interaction boundaries.

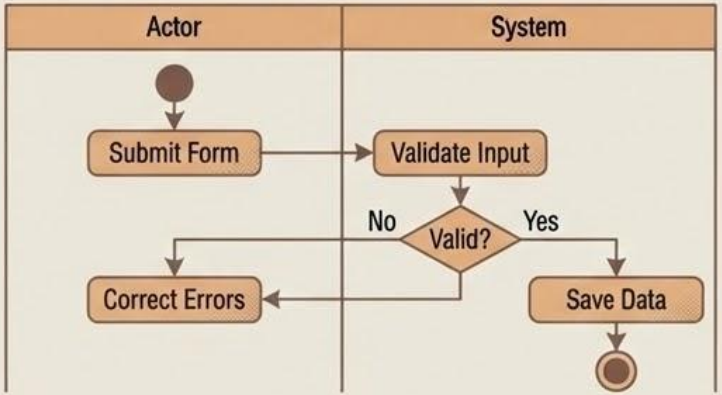


FIGURE 6.5

Activity diagram for Access camera surveillance via the Internet—display camera views function.

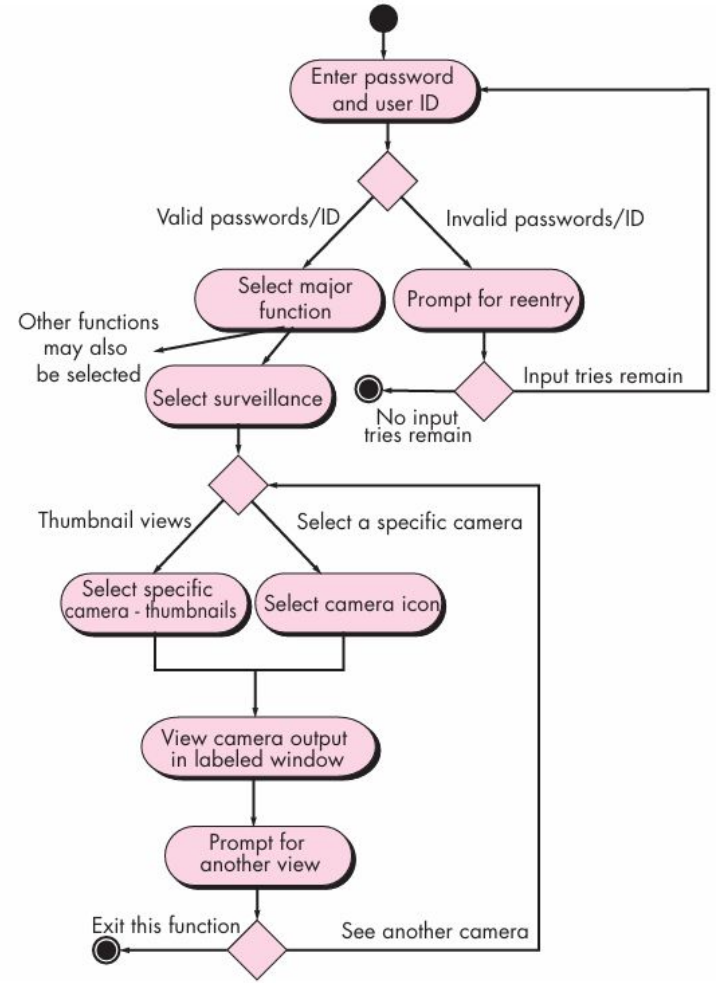
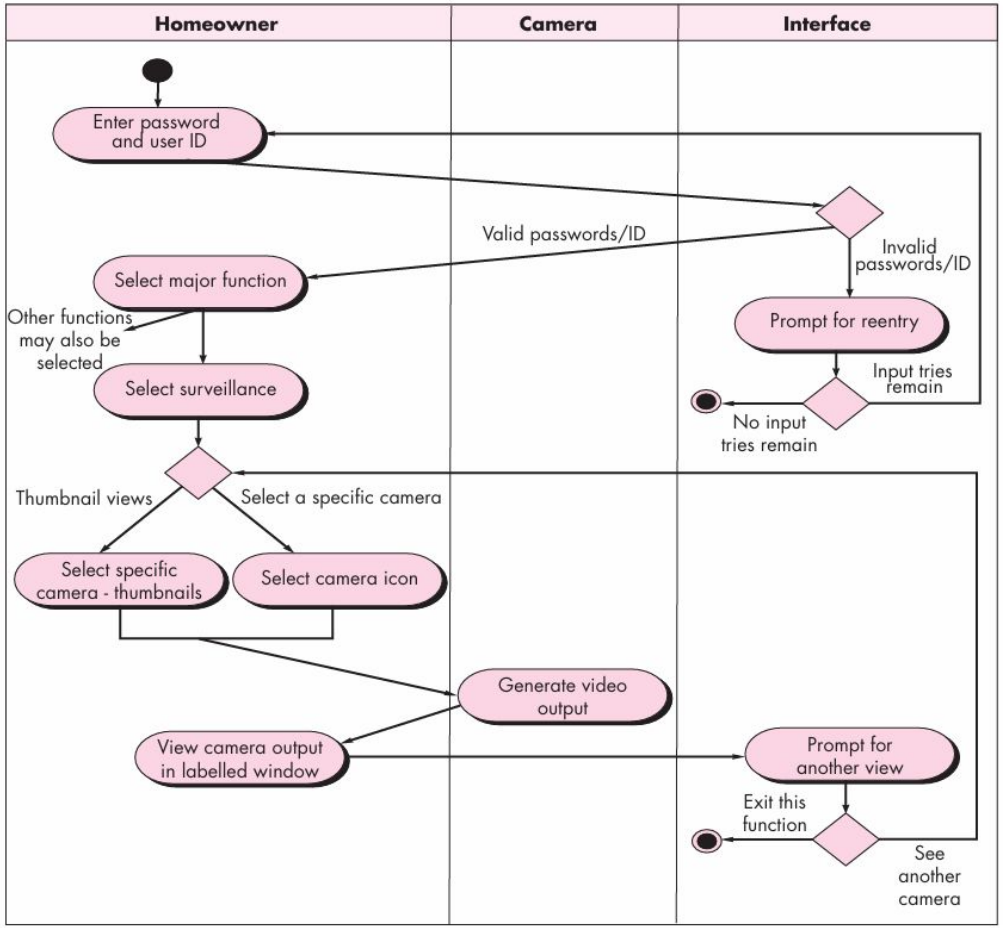


FIGURE 6.6 Swimlane diagram for Access camera surveillance via the Internet—display camera views function



Part 3: Data Modeling

The Information Backbone

Goal:
Determine what information the system needs to remember.

Data Objects:

- Composite information items the system must know about (Person, Place, Thing, Event).
- Represented as nouns (e.g., Customer, Order).

Data Attributes:
Properties of a data object.

Identifier (Key):
Attributes that uniquely identify an object instance (e.g., CustomerID).

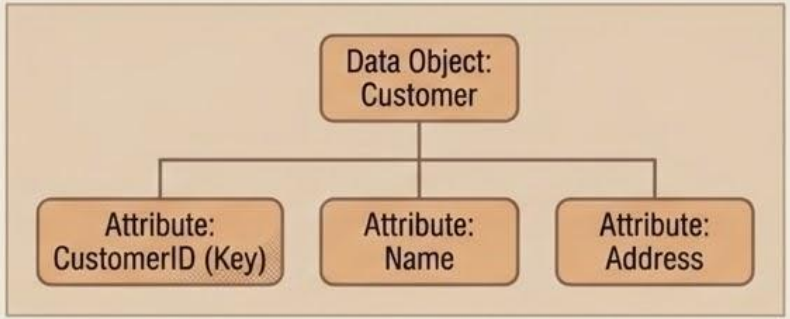


FIGURE 6.7

Tabular representation of data objects

Instance

Naming attributes

Identifier

Descriptive attributes

Ties one data object to another, in this case, owner

Referential attributes

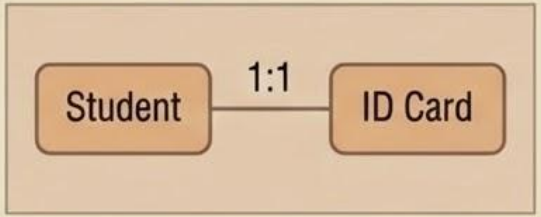
Make	Model	ID#	Body type	Color	Owner
Lexus	LS400	AB123. . .	Sedan	White	RSP
Chevy	Corvette	X456. . .	Sports	Red	CCD
BMW	750iL	XZ765. . .	Coupe	White	LJL
Ford	Taurus	Q12A45. . .	Sedan	Blue	BLF

Relationships & Cardinality

Relationship:
A connection between data objects.

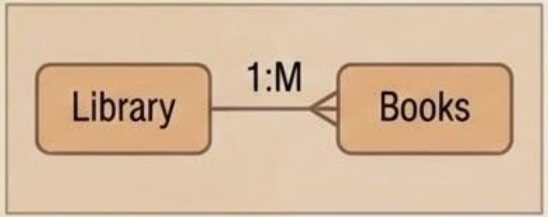
Cardinality:
Defines the number of occurrences of one object relating to another.

1:1 (One-to-One)



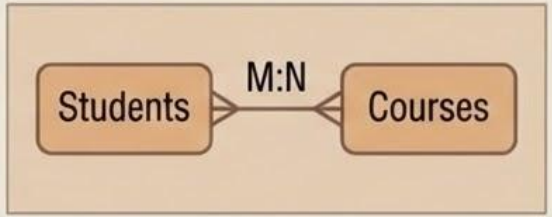
e.g., Student to ID Card.

1:M (One-to-Many)



e.g., Library to Books.

M:N (Many-to-Many)

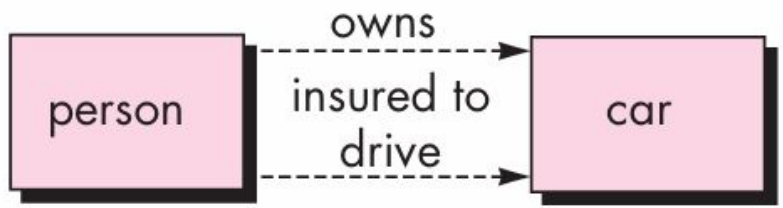


e.g., Students to Courses.

FIGURE 6.8

Relationships between data objects

(a) A basic connection between data objects



(b) Relationships between data objects

Conclusion

Key Takeaways

Analysis: Transforming vague needs into precise specifications (What, not How).

Scenario-Based: Captures requirements from the user's perspective (Use Cases).

Visuals: UML Activity & Swimlane diagrams show workflow and responsibility.

Data Modeling: Defines the backbone via Objects, Attributes, and Relationships.

Integration: Use cases manipulate the data objects defined in the data model.

Final Thought: "A model is a lie that helps you see the truth."