KAIST

Spring 2025

CS30500:
Introduction to Software Engineering

Ch 8. Requirements Modeling

Prof. In-Young Ko

School of Computing

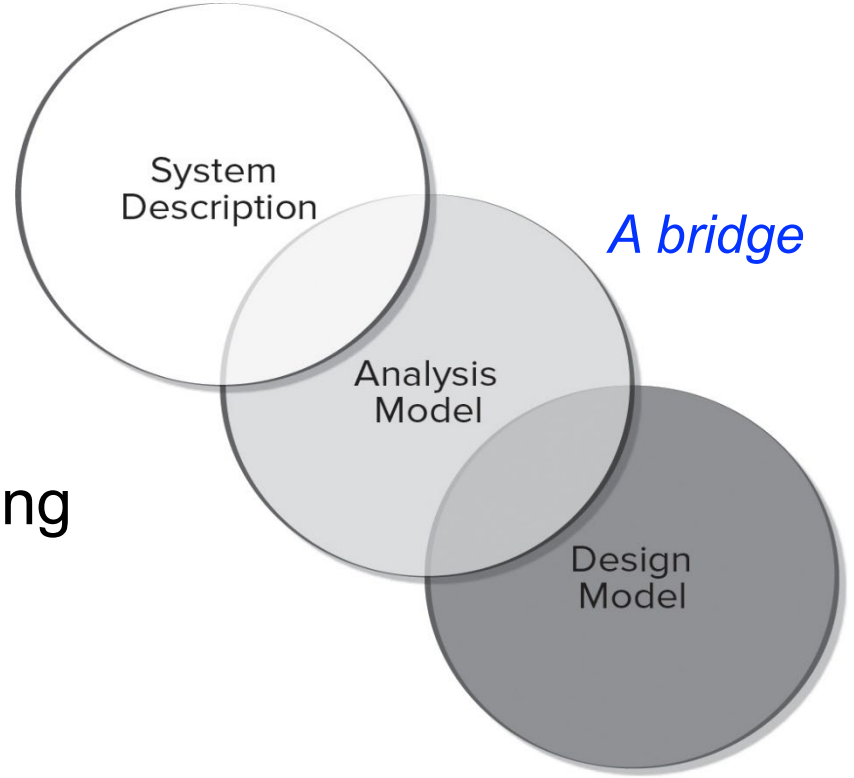
REQUIREMENTS ANALYSIS

Requirements Problems…

https://gjconstructs.wordpress.com/2012/10/15/colliding-doors-bathroom-danger/

Requirements Analysis

- Specifies software's operational characteristics
- Indicates software's interface with other system elements
- Establishes constraints that software must meet
- Requirements analysis allows the software engineer to:
  - Elaborate on basic requirements established during earlier requirement engineering tasks
  - Build models that depict the user's needs from several different perspectives

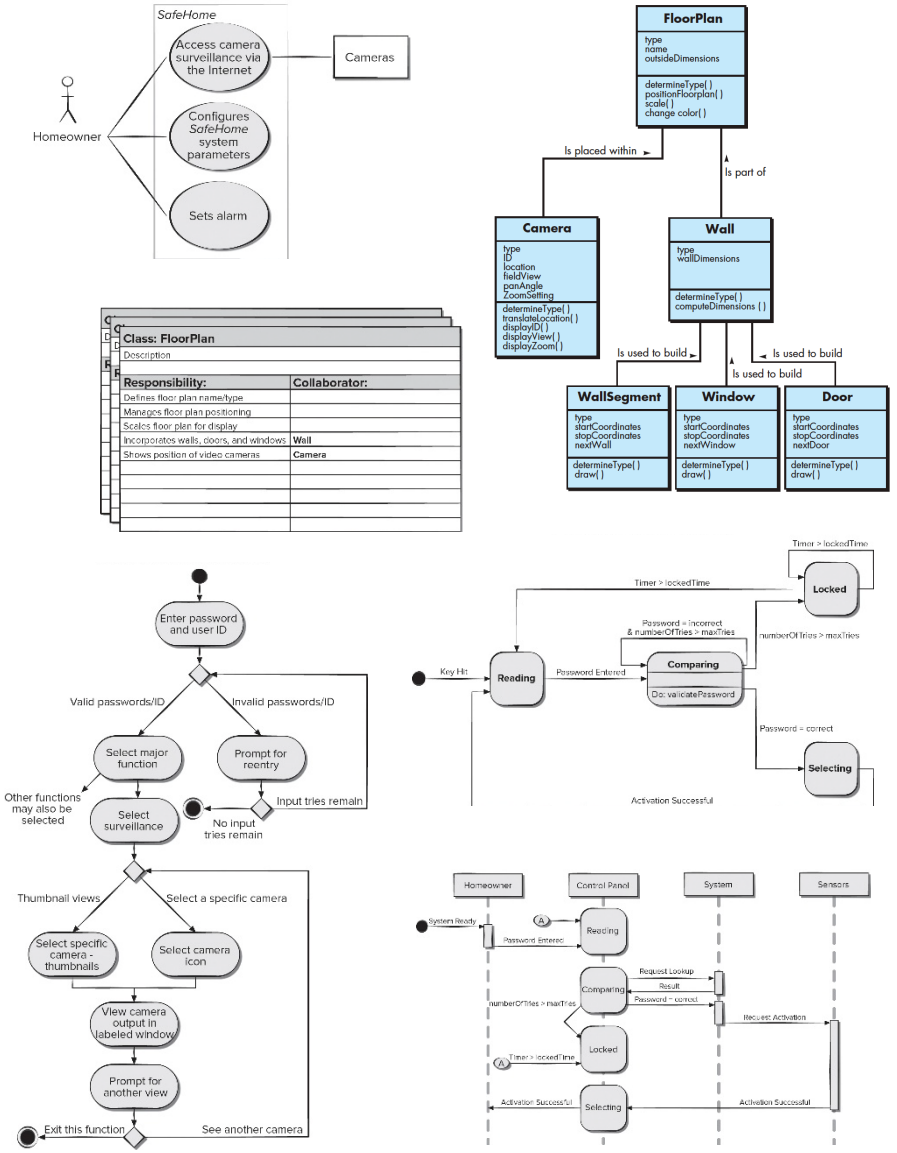


A bridge

[PrMa20]

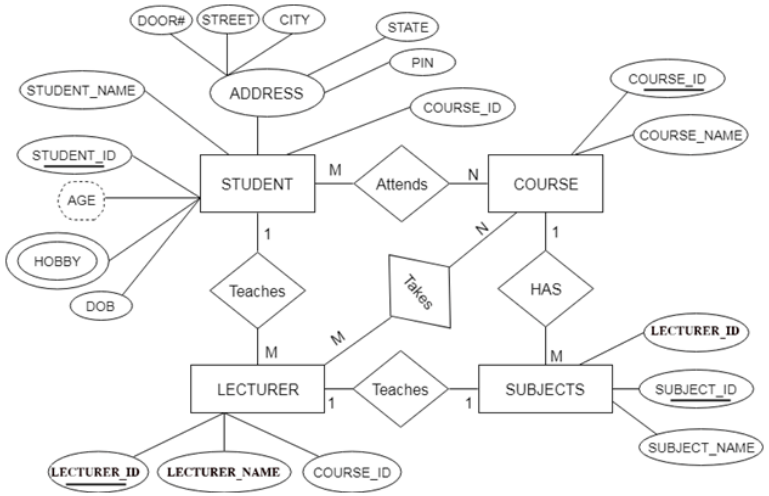
Requirements Models (1/2)

- **Scenario-based models**
  - Depict requirements from the point of view of "actors"
  - e.g., storyboards, use case model
- **Class-oriented models**
  - Represent object-oriented classes (attributes and operations) and how classes collaborate
  - e.g., class diagrams, CRC model
- **Behavioral models**
  - Depict how the software reacts to internal or external "events"
  - e.g., control flow diagrams, Petri nets, sequence diagrams, activity diagrams, state diagrams

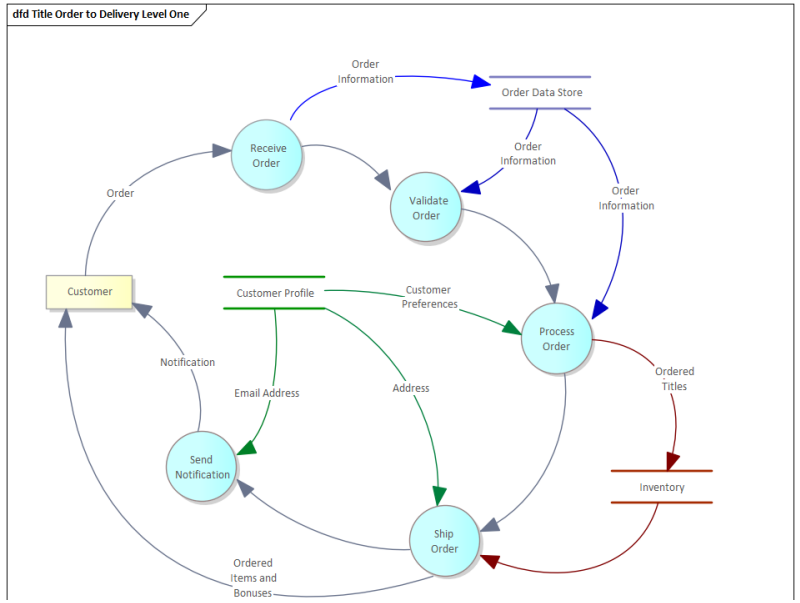


Requirements Models (2/2)

- **Data models**
  - Depict the information domain for the problem
  - e.g., entity relationship (ER) diagrams, domain model (class diagrams)
- **Flow-oriented models**
  - Represent functional elements and how they transform data
  - e.g., data flow diagrams



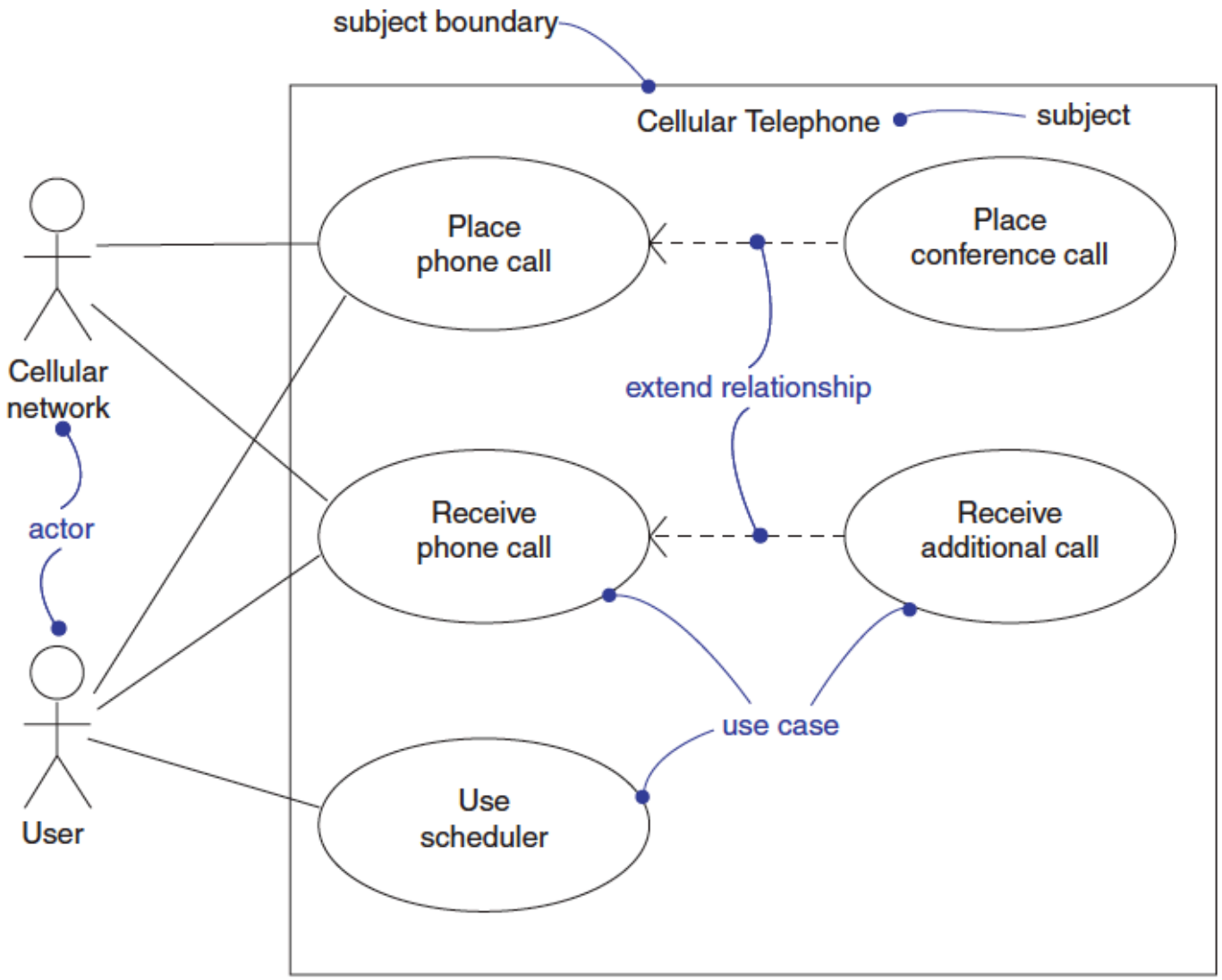
https://itwiki.kr/w/ER_%EB%8B%A4%EC%9D%B4%EC%96%B4%EA%B7%B8%EB%9E%A8



https://sparxsystems.com/enterprise_architect_user_guide/15.2/guidebooks/tools_ba_data_flow_diagram.html

USE CASE MODELING

Use Case Diagram…



[BRJ05]

Use-Case Technique

- Textual, structural, and visual modeling techniques for specifying use cases (Ivar Jacobson, 1986)
- An object-oriented software engineering technique
- For describing a system's behavior from the perspective of how the various users interact with the system
- A user centric approach by which system behaviors can be explored with early user involvement
- **Use Case**: describes a sequence of actions a system performs that yields a result of value to a particular *actor*
- **Use Case Model**: consists of *actors* and various *use cases* by which the actors interact with the system
  - Describe functional behavior of the system, and relationships among use cases

[LeWi99]

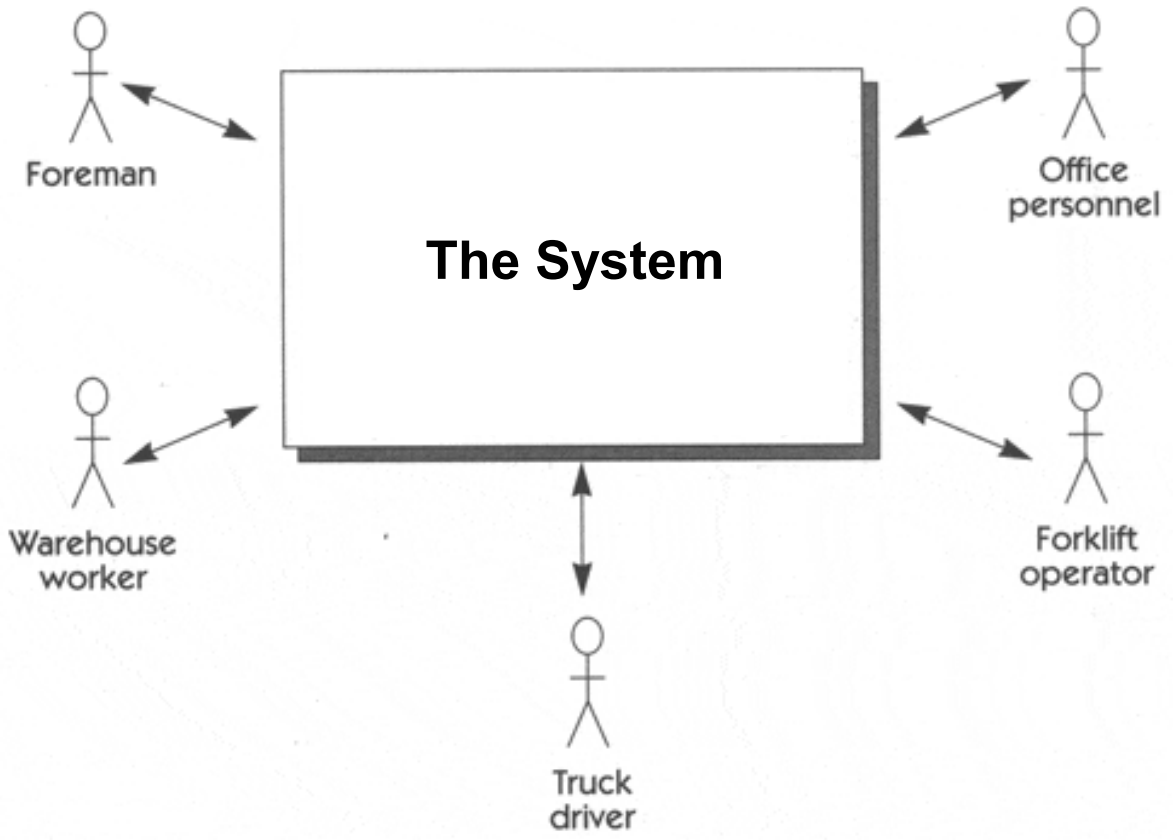
Use Case Modeling

- *Use case model* is the collection of diagrams, descriptions needed to represent the use cases and actors
- Use case modeling process
  1. Defines the system boundary
  2. Finds the actors
  3. Finds the use cases
  4. Describes each use case
  5. Refactors the use case model
  6. Prioritizes use cases
  7. Adds future requirements (change cases)
  8. Organizes the use case model (packages)

The content of this slide is adopted from the lecture materials of the Methods course (17-652) at Carnegie Mellon University.

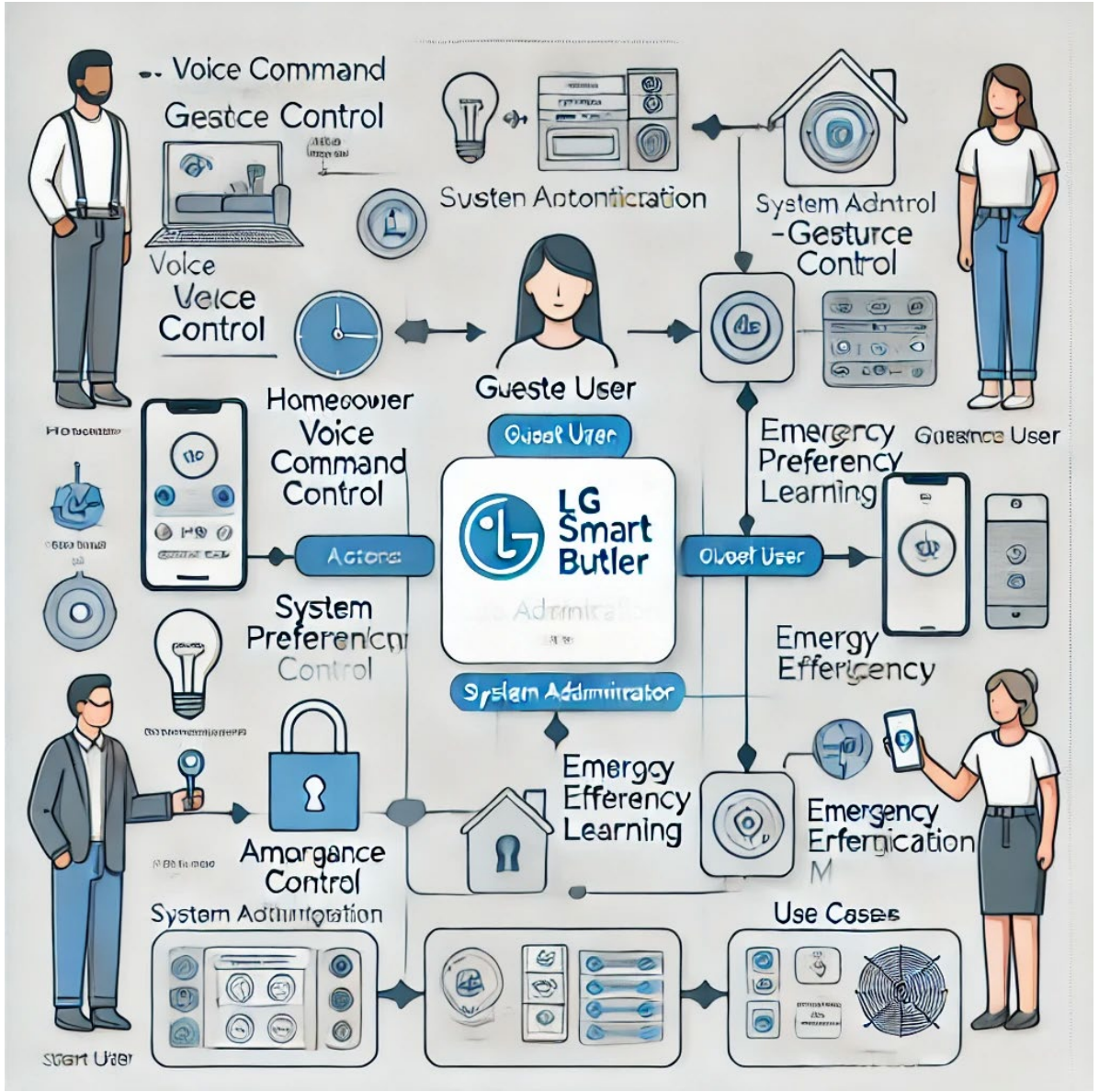
System Diagram (Context Diagram)

- Describes the system boundaries and identifies the actors of the system
- Creating a system diagram is the first step in use-case modeling



[LeWi99]

Vision Diagram Generated by ChatGPT



Actors

- *Actor*: an entity that interacts with the system
  - The stakeholders that interact with the system influence how it may be designed
  - Software development is seldom from scratch, often there are multiple interacting components where the new application is the member of a larger collaborative scenario of multiple applications
- Defining the actors helps define the boundaries of the system
- Actors that interact with the system are *roles*, i.e. generally defined designations, not specific people or instances

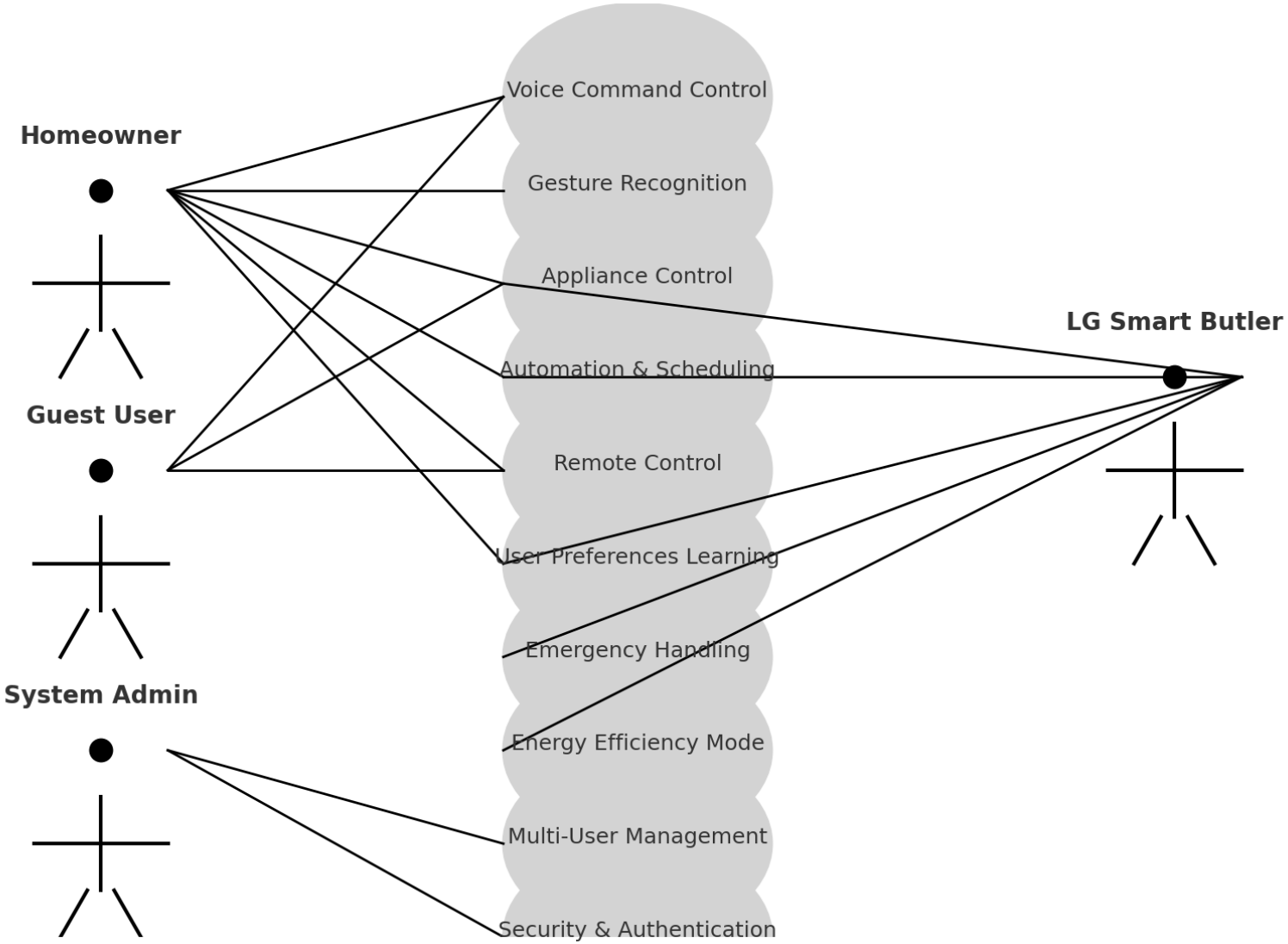
The content of this slide is adopted from the lecture materials of the Methods course (17-652) at Carnegie Mellon University.

Use Cases

- A use case is a story about some way of using a system to do something useful
  - Captures a contract between the stakeholders of a system about its behavior [Coc01]
  - Describes what the system does at a conceptual level
- A use case is goal oriented
  - Not necessarily functionality oriented, but functionality covers a lot of the use cases
  - A use case outlines a single goal and all the possible things that can happen as the users tries to reach that goal
  - Use cases are not simply menu items or functions

The content of this slide is adopted from the lecture materials of the Methods course (17-652) at Carnegie Mellon University.

Use Case Diagram Generated by ChatGPT



DESCRIBING USE CASES

Describing a Use Case (1/2)

- Describing in text the detailed sequence interaction between an actor and the system
- An important requirement elicitation tool
- The templates give guidance for the thinking process
  - Which *actor(s)*, *other use cases* that initiate the use case
  - What are the *preconditions* of the use case
  - How does the use case proceed (the *flow of events*)
  - What happens once the use case is over, what are the *post conditions*
  - What are the *alternative sequences of events (exceptions)*, what are the *decision points* that may lead the user into alternative paths

The content of this slide is adopted from the lecture materials of the Methods course (17-652) at Carnegie Mellon University.

Use Case Description Example

Name	Make reservation
Summary	Reservation clerk reserves a room for a hotel guest
Actor	
Precondition	
Description	
Alternatives	
Postcondition	

Gomaa, H., & Olimpiew, E. M. (2005). The role of use cases in requirements and analysis modeling. In *Workshop on Use Cases in Model-Driven Software Engineering. Montego Bay, Jamaica*.

Use Case Description Example

Name	Make reservation
Summary	Reservation clerk reserves a room for a hotel guest
Actor	Reservation clerk
Precondition	Reservation clerk has logged into the Hotel system
Description	1. Guest gives personal details to reservation clerk, such as name, room type, number of occupants, and dates and times of arrival and departure 2. Reservation clerk enters information into the system and looks up availability and room reservation rate 3. If room is available, clerk requests the guest to guarantee the reservation with a credit card number 4. The clerk enters the guest's credit card number to guarantee the reservation. 5. If credit card number is valid, the clerk updates the reservation as guaranteed.
Alternatives	Room not available: Line 3: If a room is not available, clerk requests the customer for another date and / or time of arrival and departure Invalid card: Line 5: If the credit card number is not valid, the clerk requests the customer for another credit card number
Postcondition	The reservation is created for the guest.

[GoOi05] Gomaa, H., & Olimpiew, E. M. (2005). The role of use cases in requirements and analysis modeling. In *Workshop on Use Cases in Model-Driven Software Engineering. Montego Bay, Jamaica.*

Use Case Description Generated by ChatGPT (1/2)

Use Case: Voice Command Control

1. Use Case ID: UC-VC-001

2. Use Case Name: Voice Command Control

3. Actors

Primary Actor: User (Homeowner, Resident)

Secondary Actors: LG Butler Robot, Smart Home Appliances (e.g., Air Conditioner, Refrigerator, Washing Machine, Oven)

4. Description

This use case describes how the LG Butler Robot allows users to control and coordinate smart home appliances using voice commands. The butler robot listens to user commands, processes them using natural language understanding (NLU), and sends instructions via APIs to the relevant home appliances for execution.

5. Preconditions

The LG Butler Robot is powered on and connected to the home network.

Smart home appliances are registered, online, and integrated with the LG ThinQ platform.

The user has an authenticated profile in the system.

Voice recognition is enabled and trained to recognize the user's voice.

6. Postconditions

The requested action is successfully performed, and feedback is provided to the user.

In case of failure, the user is notified with an appropriate message.

Use Case Description Generated by ChatGPT (2/2)

7. Main Flow

User initiates voice command (e.g., “Turn on the air conditioner to 22°C.”).

LG Butler Robot captures and processes the voice command using speech recognition and NLU.

Command validation: The system verifies user authentication and checks if the requested appliance is available and online.

Command execution:

If valid, the butler robot sends the instruction via API to the corresponding appliance.

The appliance processes the request and changes its state accordingly.

Feedback to user:

The butler robot confirms that the action has been executed successfully (e.g., “The air conditioner has been set to 22°C.”).

If an issue occurs (e.g., device offline), the butler robot informs the user (e.g., “The air conditioner is currently unavailable.”).

8. Alternative Flows

8.1. Invalid or Unrecognized Command

The butler robot requests clarification (e.g., “I didn’t understand. Could you please repeat?”).

8.2. Appliance Unavailable

The system notifies the user that the requested appliance is offline or unresponsive.

8.3. User Authentication Failure

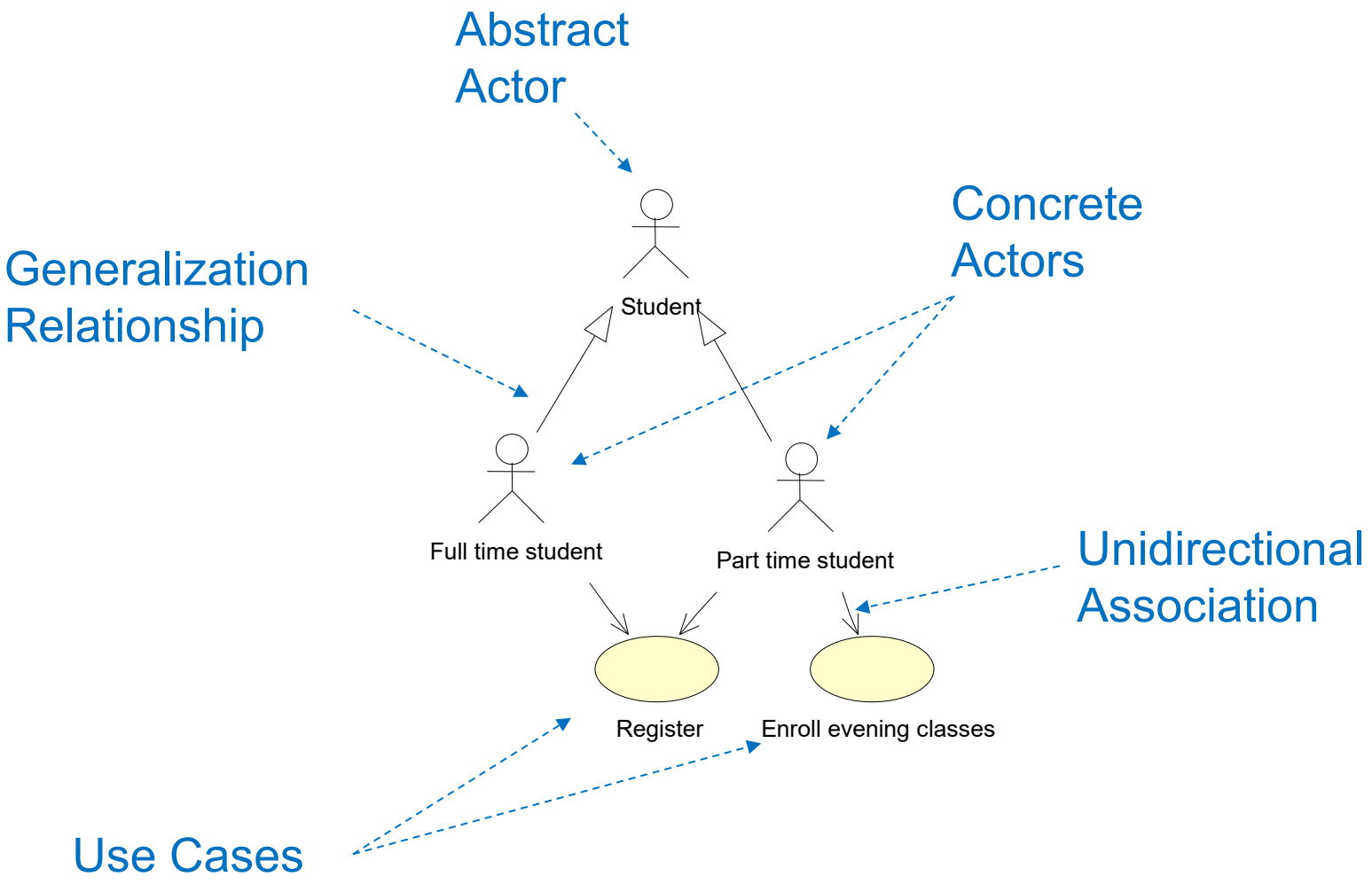
If the user is not recognized, the system prompts for authentication before proceeding.

Use Case Template & Sample

- Cockburn's Use Case Template & Sample:
  - https://home.cs.colorado.edu/~kena/classes/6448/s01/examples/edmonb3.pdf

STRUCTURING USE CASES

Use Case Diagram – The UML Context (1/3)



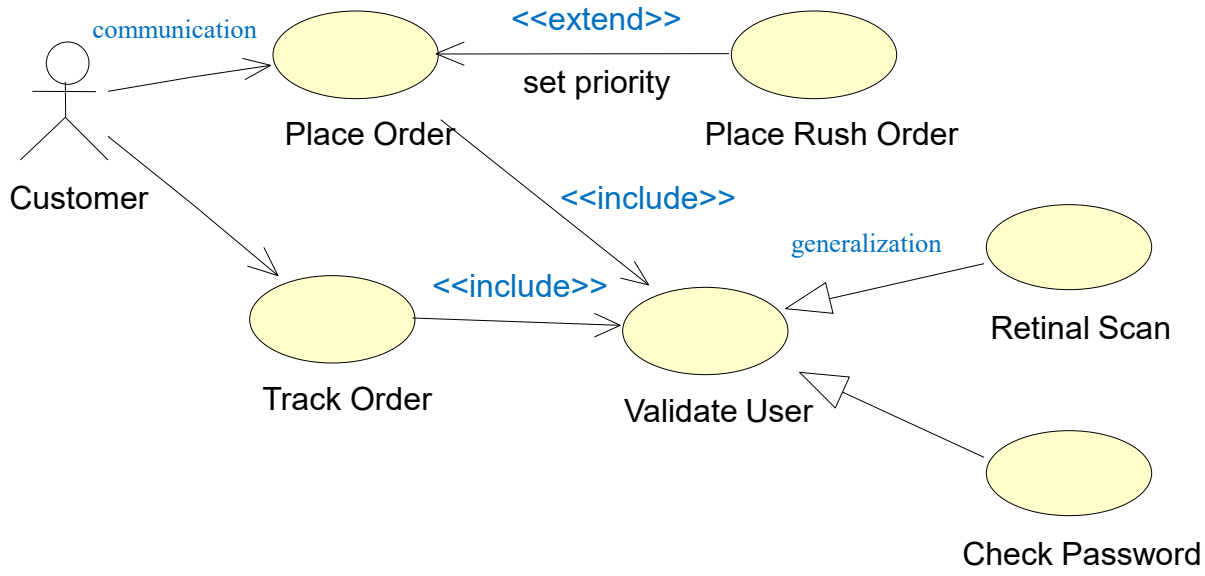
The content of this slide is adopted from the lecture materials of the Methods course (17-652) at Carnegie Mellon University.

Use Case Diagram – The UML Context (2/3)

Relationships

- Communication
- Generalization
- Includes or uses
- Extends

[Booch, Rumbaugh, Jacobsen (1999)
The Unified Modeling Language Guide]



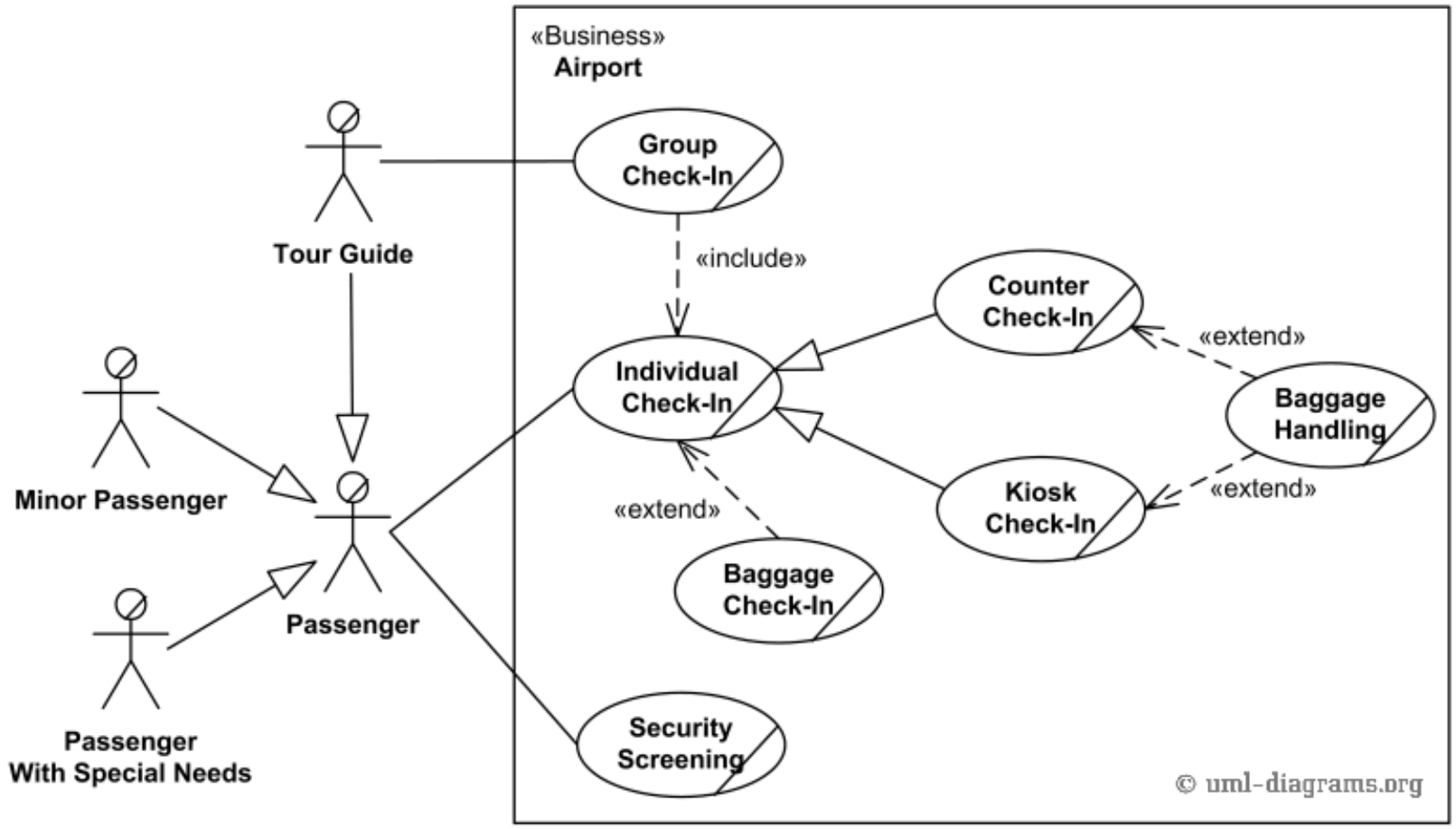
The content of this slide is adopted from the lecture materials of the Methods course (17-652) at Carnegie Mellon University.

Use Case Diagram – The UML Context (3/3)

- Communication
  - The only relationship permissible between actors and use-cases
- Generalization
  - Similar to parent/child relationship
  - The specific elements share structure and behavior with the general element
- Includes/Uses
  - Used when a base use-case instance includes the behavior specified by another use-case
  - Models common behavior among use-cases
- Extends
  - Used when one use-case adds functionality onto another use case
  - Extending use-case continues the activity of the starting base use-case

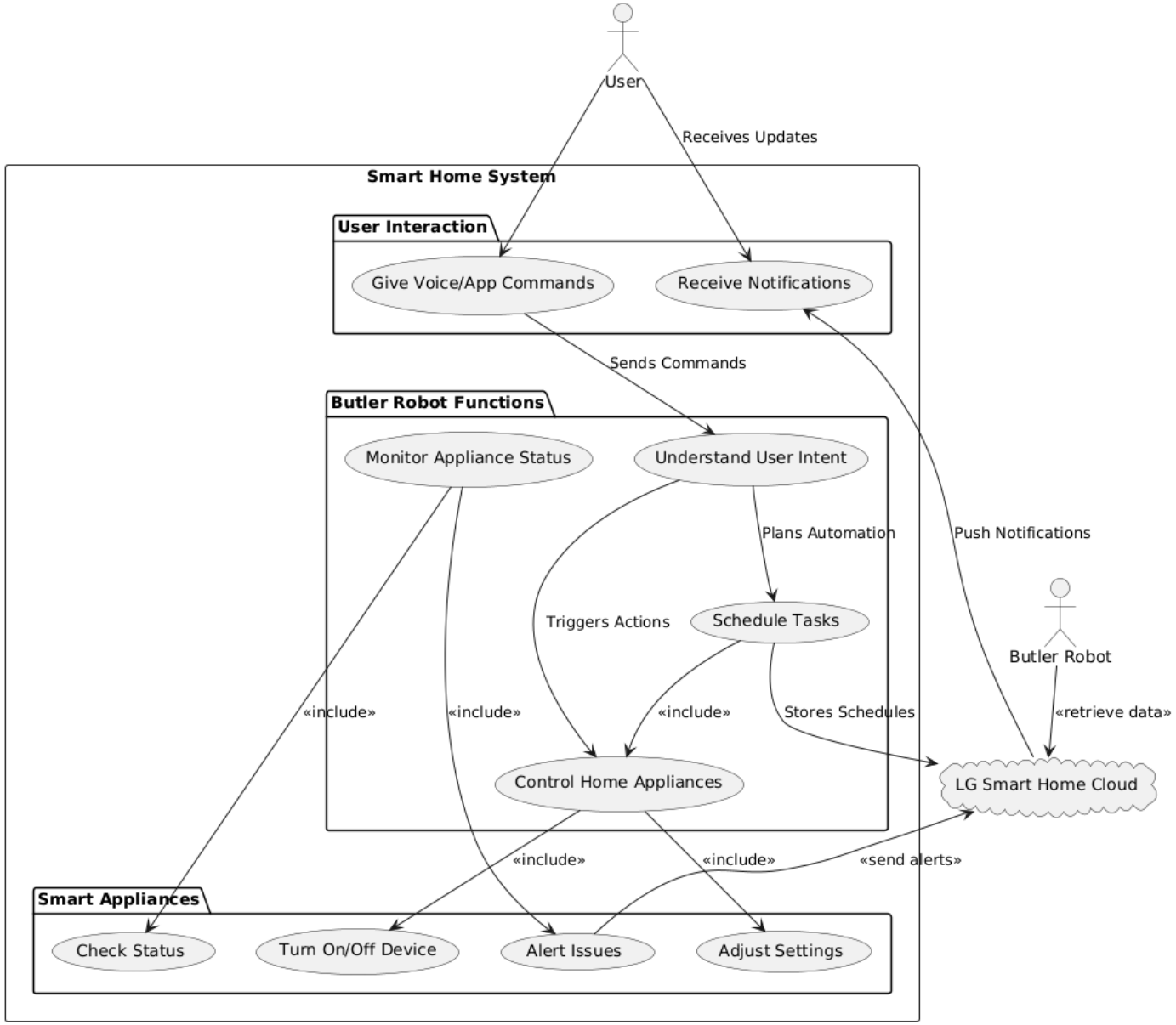
The content of this slide is adopted from the lecture materials of the Methods course (17-652) at Carnegie Mellon University.

Use Case Diagram Example



http://www.uml-diagrams.org/airport-checkin-uml-use-case-diagram-example.html

Use Case Model Structured by ChatGPT + UML Diagram Creator



USE CASE SCENARIOS

Use Case Scenarios

- Use cases describe abstract and general behavior
- Use cases do not describe what happens when a specific actor performs a specific action with specific values
- Scenarios: specific executions ➔ also denoted as *use case instances*
  - Variations of input and output parameters
  - Variations on flow of events
- Scenarios carry use cases to a larger requirement elicitation context
  - Validate requirements
  - Identify any unaddressed nonfunctional requirements
    - Performance and usability commonly
  - Create a broader system prototype
  - Flow nicely into test cases
  - Success and failure scenarios (*primary* and *secondary* use cases)

Use Case Scenario Generated by ChatGPT

Primary (Main) Scenario for “Voice Command Control”:

1. **User** activates the system using a wake word (e.g., "Hey LG").
2. **System** detects the wake word and starts listening.
3. **User** gives a voice command (e.g., "Turn on the air conditioner").
4. **System** processes the command using speech-to-text conversion.
5. **System** interprets the command and identifies the corresponding appliance.
6. **System** executes the command and provides feedback (voice or display).
7. **System** confirms successful execution to the **User**.

Alternative Use Case Scenarios Generated by ChatGPT

1. Incorrect Wake Word Detection

1.1 User tries to wake up the system with an unrecognized phrase.
1.2 System does not activate or asks the user to repeat.
1.3 User retries with the correct wake word, and the system proceeds as normal.

2. Ambiguous Command Given

2.1 User says, "Turn it on."
2.2 System cannot determine which appliance is being referred to.
2.3 System asks for clarification (e.g., "Which device would you like to turn on?").
2.4 User provides a specific command, and the system executes it.

...

4. Background Noise Interference

4.1 User gives a command in a noisy environment.
4.2 System misinterprets the command or does not recognize it.
4.3 System asks the user to repeat the command.
4.4 If the environment remains too noisy, the system suggests using manual input instead.

Use Case Modeling for Your Project (Individual Practice)

- **Goal**: Describe the requirements of a subsystem of your project by doing use case modeling
- **Purpose**:
  - Exercise use case modeling by using a UML tool
- **Suggested outline of the outcome**
  - Give a brief description of your problem (subsystem)
  - Identify and briefly describe actors (not stakeholders)
  - Identify an initial set of use cases, structure the use case diagram
  - Write use case descriptions
  - Describe alternative scenarios for the use cases

➔ *This outcome can be used for writing Software Requirements Specification (SRS) for your team project.*

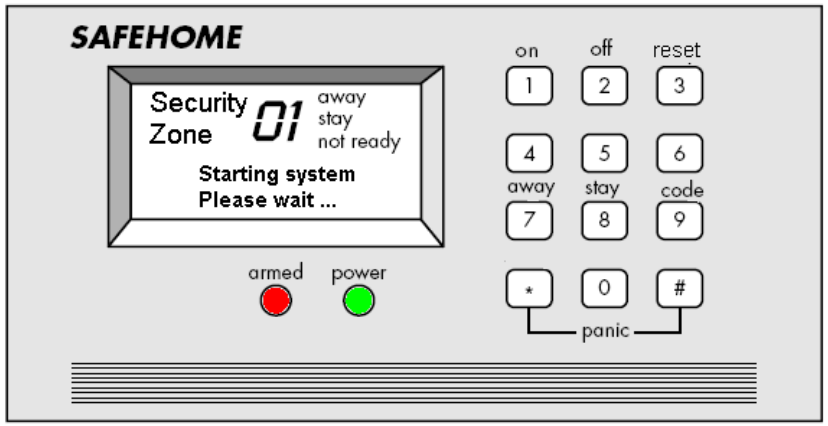
Conducting a Requirements Gathering Meeting (pg145)

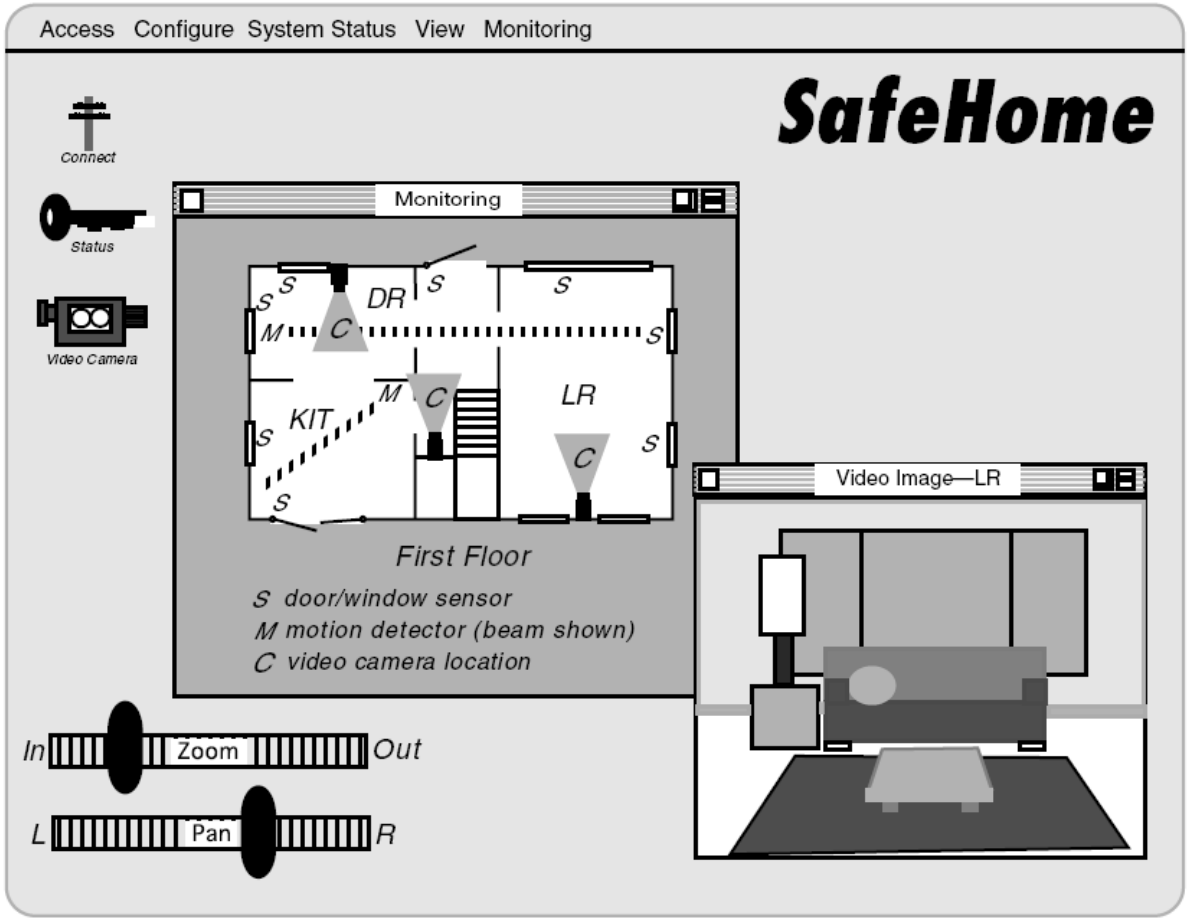
- **The scene:**
  - A meeting room. The first requirements gathering meeting is in progress.
- **The players:**
  - Jamie Lazar, software team member;
  - Vinod Raman, software team member;
  - Ed Robbins, software team member;
  - Doug Miller, software engineering manager;
  - three members of marketing;
  - a product engineering representative;
  - a facilitator.

- **The conversation:**
- Facilitator **(pointing at white board):** So that's the current list of objects and services for the home security function.
- **Marketing person:** That about covers it from our point of view.
- **Vinod:** Didn't someone mention that they wanted all *SafeHome* functionality to be accessible via the Internet? That would include the home security function, no?
- **Marketing person:** Yes, that's right ... we'll have to add that functionality and the appropriate objects.

- **Facilitator:** Does that also add some constraints?
- **Jamie:** It does, both technical and legal.
- **Production rep:** Meaning?
- **Jamie:** We better make sure an outsider can't hack into the system, disarm it, and rob the place or worse. Heavy liability on our part.
- **Doug:** Very true.
- **Marketing:** But we still need Internet connectivity. Just be sure to stop an outsider from getting in.
- **Ed:** That's easier said than done and....
- **Facilitator (interrupting):** I don't want to debate this issue now. Let's note it as an action item and proceed. (Doug, serving as the recorder for the meeting, makes an appropriate note.)
- **Facilitator:** I have a feeling there's still more to consider here.
- (The group spends the next 45 minutes refining and expanding the details of the home security function.)

SafeHome Product





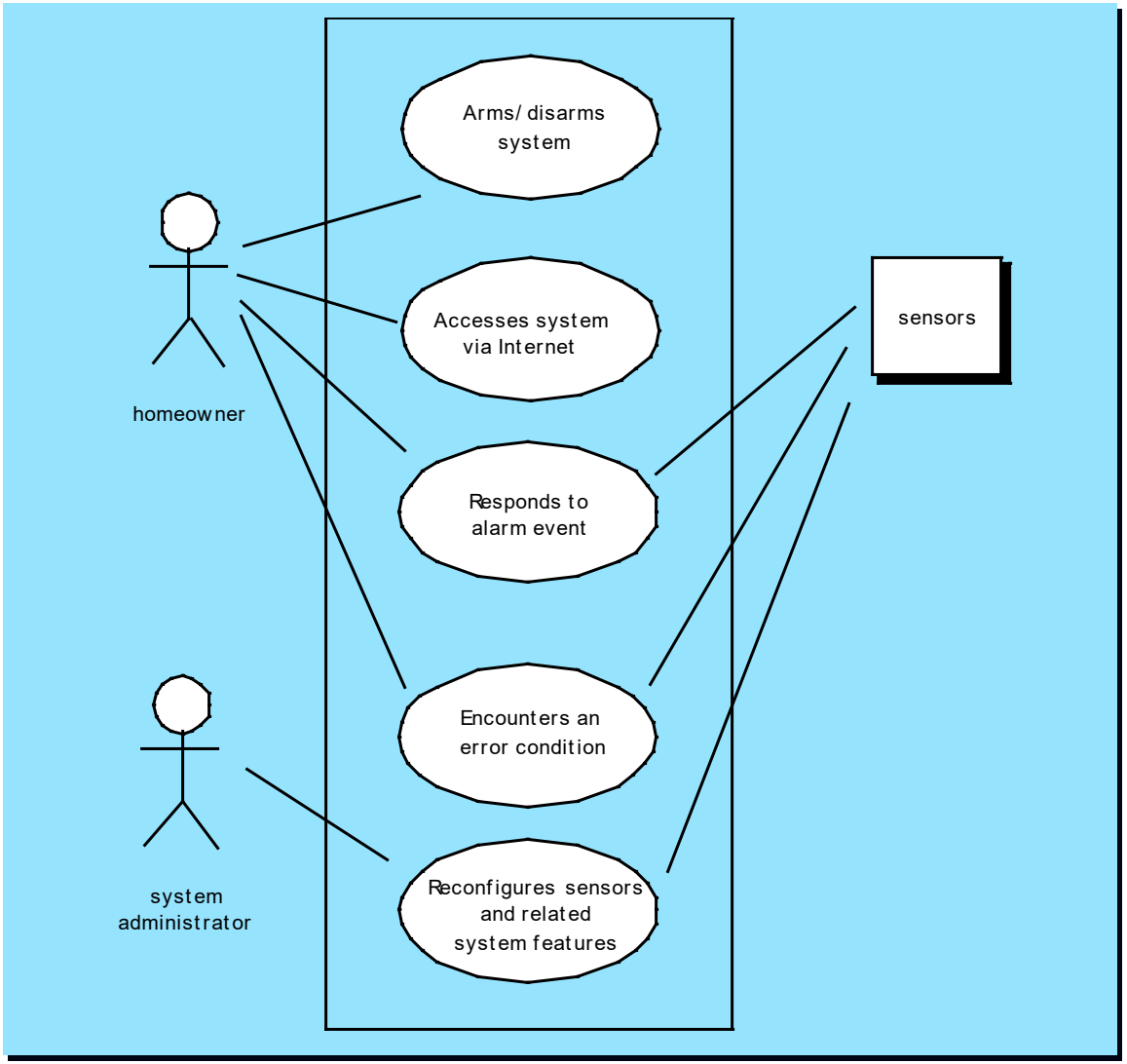
Use-Cases

- A collection of user scenarios that describe the thread of usage of a system
- Each scenario is described from the point-of-view of an "actor"—a person or device that interacts with the software in some way
- Each scenario answers the following questions:
  - Who is the primary actor, the secondary actor (s)?
  - What are the actor's goals?
  - What preconditions should exist before the story begins?
  - What main tasks or functions are performed by the actor?
  - What extensions might be considered as the story is described?
  - What variations in the actor's interaction are possible?
  - What system information will the actor acquire, produce, or change?
  - Will the actor have to inform the system about changes in the external environment?
  - What information does the actor desire from the system?
  - Does the actor wish to be informed about unexpected changes?

Example of Use Case for SafeHome

- Use-case: InitiateMonitoring
- Primary actor: Homeowner
- Goal in context: To set the system to monitor sensors when the homeowner leaves the house or remains inside
- Preconditions: System has been programmed for a password and to recognize various sensors
- Trigger: The homeowner decides to "set" the system, i.e., to turn on the alarm functions
- Scenario:
  1. Homeowner: observes control panel
  2. Homeowner:enters password
  3. Homeowner: selects "stay" or "away"
  4. Homeowner: observes red alarm light to indicate that SafeHome has been armed
- Exceptions:
  - 1a. Control panel is not ready: homeowner checks all sensors to determine which are open; closes them
  - 2a. Password is incorrect
- Priority: Essential, must be implemented
- When available: first increment
- Frequency of use: Many times per day
- Channel to actor: Via control panel interface
- Secondary actors: Support technician
- Channels to secondary actors: support technician: phone line
- Open issues:
  - Do we enforce time limit for password entering?

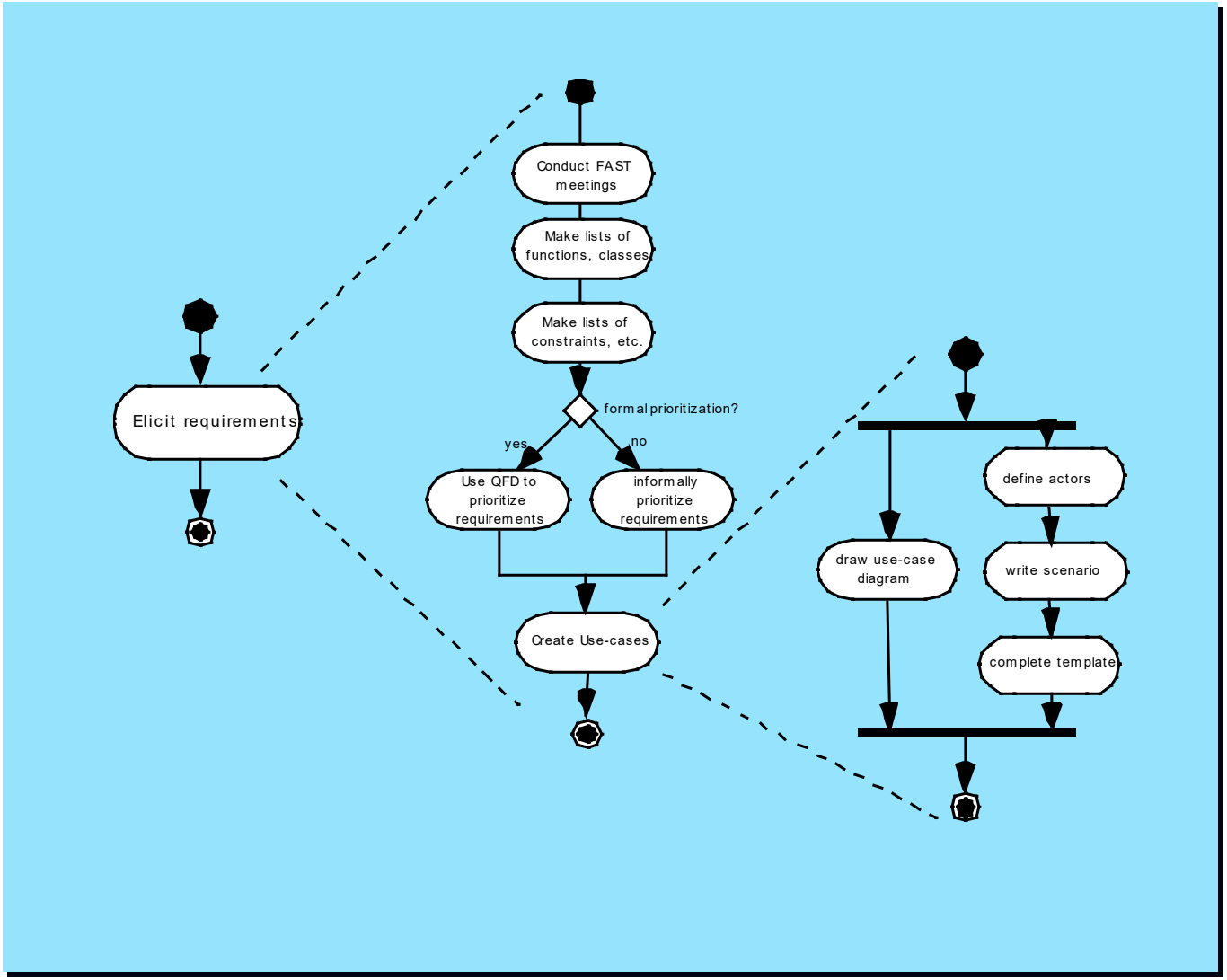
Use-Case Diagram



Building the Analysis Model

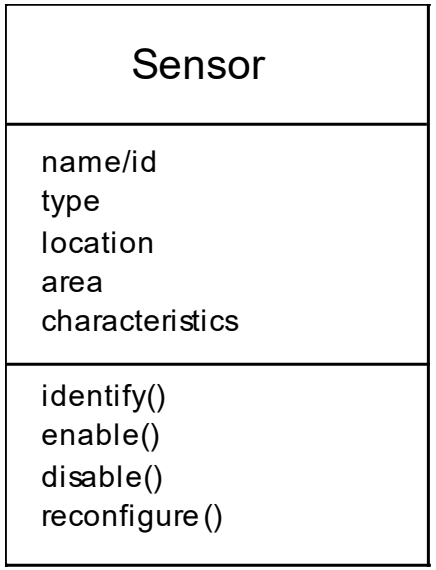
- Elements of the analysis model
  - Scenario-based elements
    - Functional—processing narratives for software functions
    - Use-case—descriptions of the interaction between an "actor" and the system
  - Class-based elements
    - Implied by scenarios
  - Behavioral elements
    - State diagram

Eliciting Requirements



Class Diagram

From the *SafeHome* system …



State Diagram

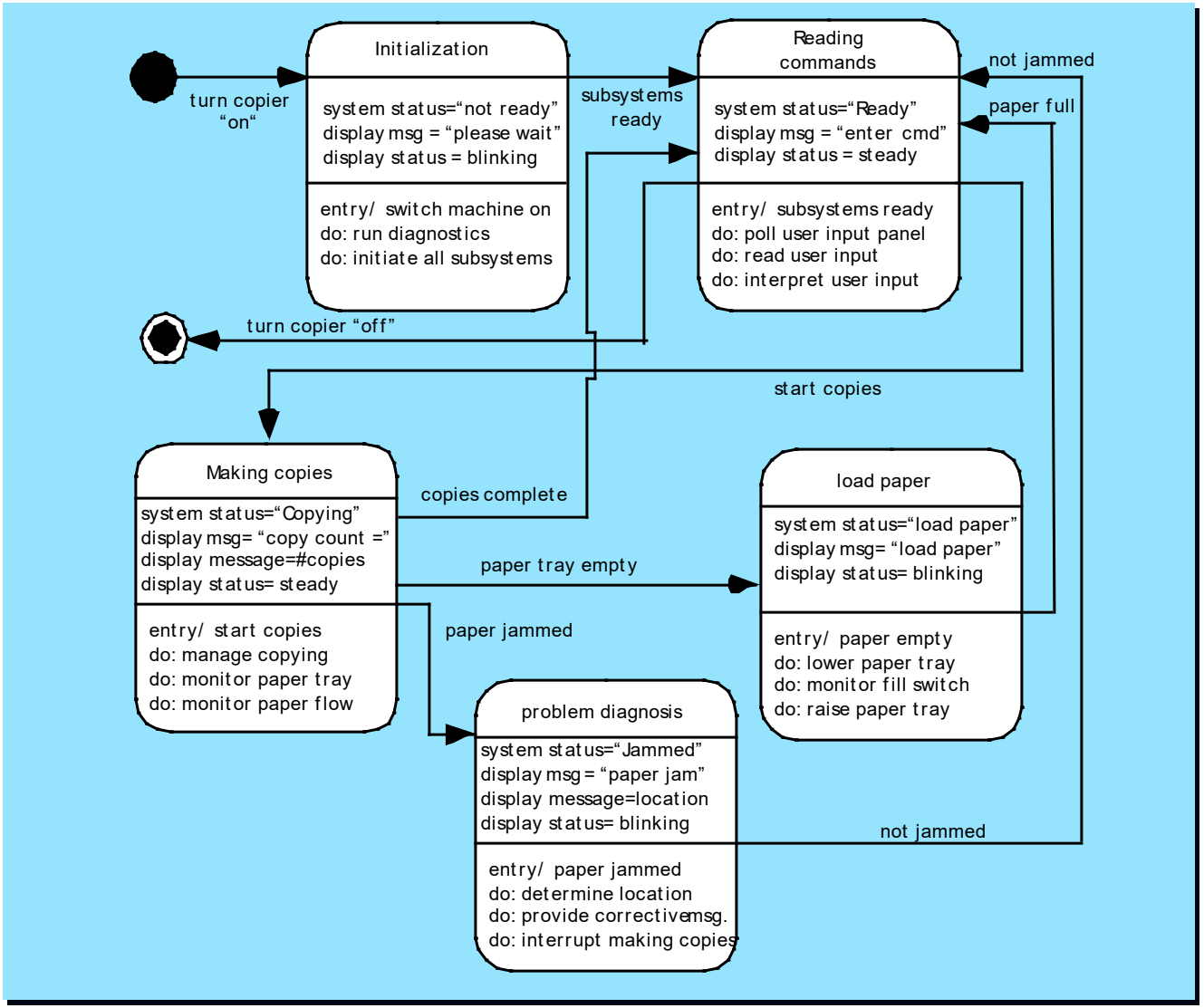


Figure 7.6 Preliminary UML state diagram for a photocopier

Negotiating Requirements

- Identify the key stakeholders
  - These are the people who will be involved in the negotiation
- Determine each of the stakeholders “win conditions”
  - Win conditions are not always obvious
- Negotiate
  - Work toward a set of requirements that lead to “win-win”

Validating Requirements-I

- Is each requirement consistent with the overall objective for the system/product?
- Have all requirements been specified at the proper level of abstraction? That is, do some requirements provide a level of technical detail that is inappropriate at this stage?
- Is the requirement really necessary or does it represent an add-on feature that may not be essential to the objective of the system?
- Is each requirement bounded and unambiguous?
- Does each requirement have attribution? That is, is a source (generally, a specific individual) noted for each requirement?

Validating Requirements-II

- Do any requirements conflict with other requirements?
- Is each requirement achievable in the technical environment that will house the system or product?
- Is each requirement testable, once implemented?
- Does the requirements model properly reflect the information, function and behavior of the system to be built.
- Has the requirements model been “partitioned” in a way that exposes progressively more detailed information about the system.

Specification Guidelines

- use a layered format that provides increasing detail as the "layers" deepen
- use consistent graphical notation and apply textual terms consistently (stay away from aliases)
- be sure to define all acronyms
- be sure to include a table of contents; ideally, include an index and/or a glossary
- write in a simple, unambiguous style (see "editing suggestions" on the following pages)
- always put yourself in the reader's position, "Would I be able to understand this if I wasn't intimately familiar with the system?"

Specification Guidelines

Be on the lookout for persuasive connectors, ask why?
keys: *certainly, therefore, clearly, obviously, it follows that ...*

Watch out for vague terms
keys: *some, sometimes, often, usually,ordinarily, most, mostly ...*

When lists are given, but not completed, be sure all items are understood
keys: *etc., and so forth, and so on, such as*

Be sure stated ranges don't contain unstated assumptions
e.g., *Valid codes range from 10 to 100.* Integer? Real? Hex?

Beware of vague verbs such as *handled, rejected, processed, ...*

Beware "passive voice" statements
e.g., *The parameters are initialized.* By what?

Beware "dangling" pronouns
e.g., *The I/O module communicated with the data validation module and its contol flag is set.* Whose control flag?

Specification Guidelines

When a term is explicitly defined in one place, try substituting the definition forother occurrences of the term

When a structure is described in words, draw a picture

When a structure is described with a picture, try to redraw the picture to emphasize different elements of the structure

When symbolic equations are used, try expressing their meaning in words

When a calculation is specified, work at least two examples

Look for statements that imply certainty, then ask for proof
keys; always, every, all, none, never

Search behind certainty statements뻚e sure restrictions or limitations are realistic

QUESTIONS?