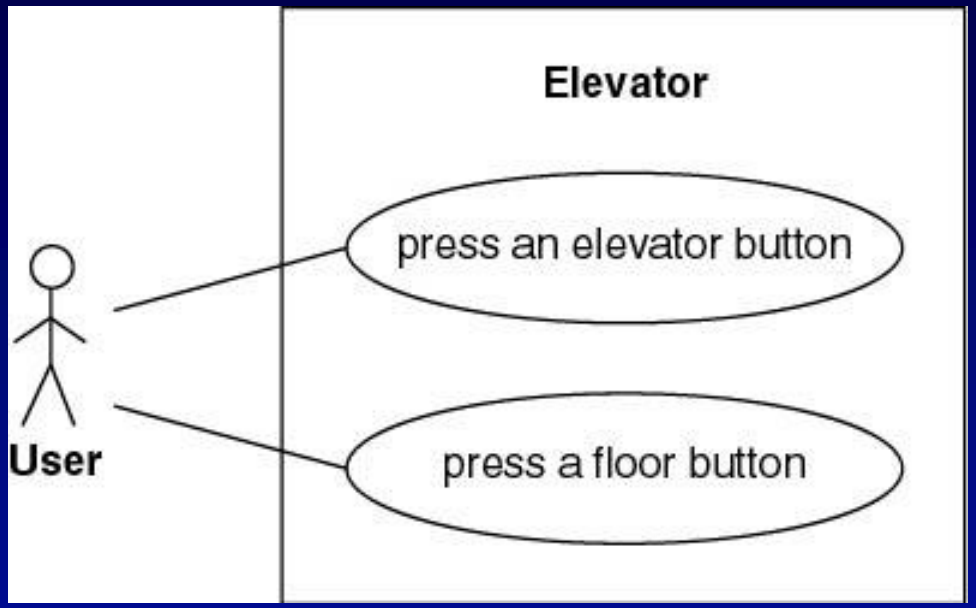
Analysis Modeling Case Study: Evaluator Control Software

Elevator Problem

- A product is to be installed to control n elevators in a building with m floors
- The problem concerns the logic required to move elevators between floors according to the following constraints
  - Each elevator has a set of m buttons, one for each floor. These illuminate when pressed and cause elevator to visit corresponding floor. Illumination is canceled when corresponding floor is visited by elevator
  - Each floor, except the first and the top floor, has 2 buttons, one to request an up-elevator, one to request a down-elevator. These buttons illuminate when pressed. The illumination is canceled when an elevator visits the floor, then moves in the desired direction
  - If an elevator has no requests, it remains at its current floor with its doors closed

Elevator Problem

- Use Case Modeling



- Get comprehensive insight of system behavior
  - Normal scenarios and exception scenarios

Normal Scenario

1. User A presses the Up floor button at floor 3 to request an elevator. User A wishes to go to floor 7.
2. The Up floor button is turned on.
3. An elevator arrives at floor 3. It contains User B, who has entered the elevator at floor 1 and pressed the elevator button for floor 9.
4. The Up floor button is turned off.
5. The elevator doors open.
6. The timer starts.
   User A enters the elevator.
7. User A presses the elevator button for floor 7.
8. The elevator button for floor 7 is turned on.
9. The elevator doors close after a timeout.
10. The elevator travels to floor 7.
11. The elevator button for floor 7 is turned off.
12. The elevator doors open to allow User A to exit from the elevator.
13. The timer starts.
    User A exits from the elevator.
14. The elevator doors close after a timeout.
15. The elevator proceeds to floor 9 with User B.

Noun Extraction

- Stage 1. Concise Problem Definition
  - Define product as briefly and concisely as possible, preferably in a single sentence

"Buttons in elevators and on the floors control the motion of n elevators in a building with m floors."

Noun Extraction

- Stage 2. Informal Strategy
  - Incorporate constraints, express result in a single paragraph

"Buttons in elevators and on the floors control movement of n elevators in a building with m floors. Buttons illuminate when pressed to request the elevator to stop at a specific floor; illumination is canceled when the request has been satisfied. When an elevator has no requests, it remains at its current floor with its doors closed."

Noun Extraction

- Stage 3. Formalize the Strategy
  - Identify nouns in informal strategy, then use these nouns as candidate classes

"**Buttons** **in** **elevators** and on the **floors** control **movement** of n **elevators** in a **building** with m **floors**. **Buttons** illuminate when pressed to request the **elevator** to stop at a specific **floor**; **illumination** is canceled when the **request** has been satisfied. When an **elevator** has no **requests**, it remains at its current **floor** with its **doors** closed."

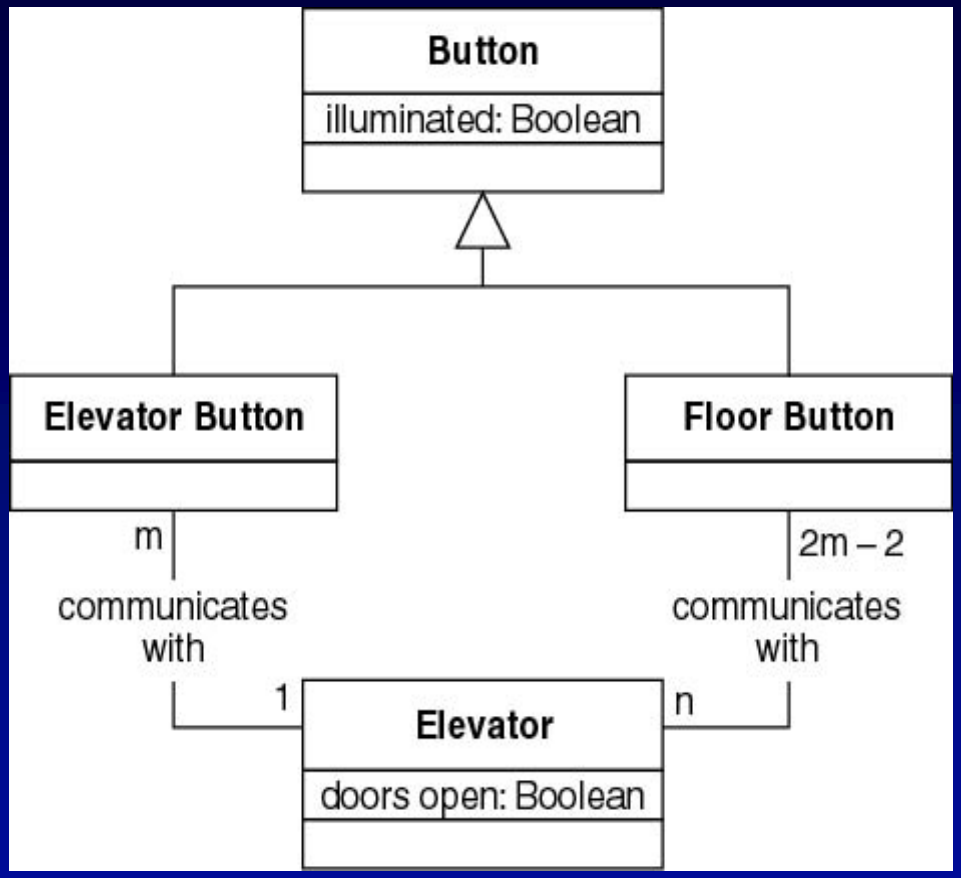
Noun Extraction

- Nouns
  - **button**, **elevator**, **floor**, **movement**, **building**, **illumination**, **request**, **door**
- **floor**, **building**, **door** are outside the problem boundary
  - Exclude
- **movement**, **illumination**, **request** are abstract nouns
  - Nouns identifying ideas or quantities that have no physical existence
  - Rarely end up corresponding to classes but frequently are attributes of classes
  - For example, **Illumination** is an attribute of **button**
  - Also exclude

Noun Extraction

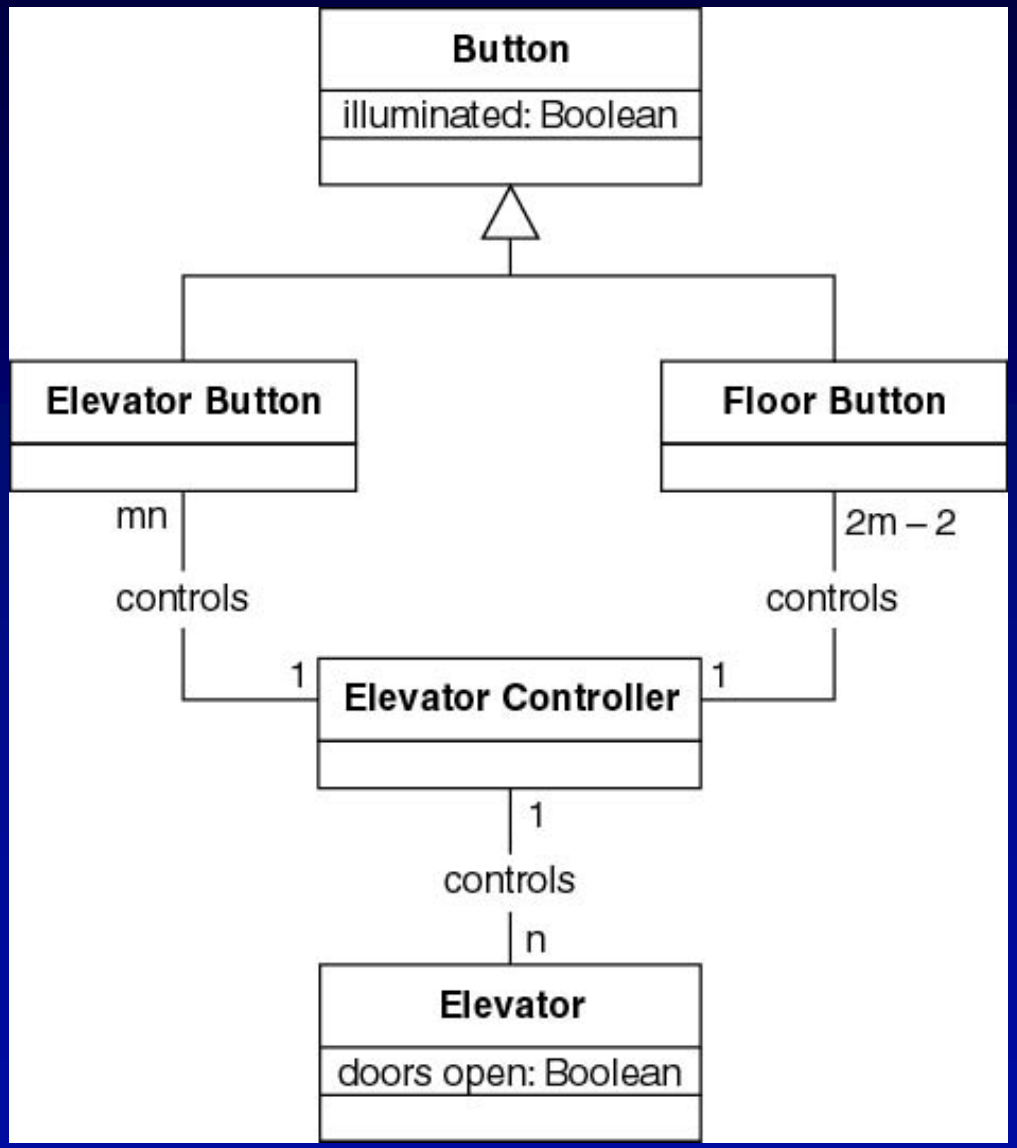
- Candidate classes
  - **Elevator**
  - **Button**
- Subclasses
  - **Elevator Button**
  - **Floor Button**

First Iteration of Class Diagram



- Problem
  - Buttons do not communicate directly with elevators
  - We need an additional class: **Elevator Controller**

Second Iteration of Class Diagram



- It is possible to return to class modeling at any time
  - Even as late as integration phase
- All relationships are now 1-to-n
  - Makes design and implementation easier

CRC Cards

- CRC cards are useful in both design and testing

CLASS **Elevator Controller**
RESPONSIBILITY 1. Turn on elevator button 2. Turn off elevator button 3. Turn on floor button 4. Turn off floor button 5. Move elevator up one floor 6. Move elevator down one floor 7. Open elevator doors and start timer 8. Close elevator doors after timeout 9. Check requests 10. Update requests
COLLABORATION 1. Class **Elevator Button** 2. Class **Floor Button** 3. Class **Elevator**

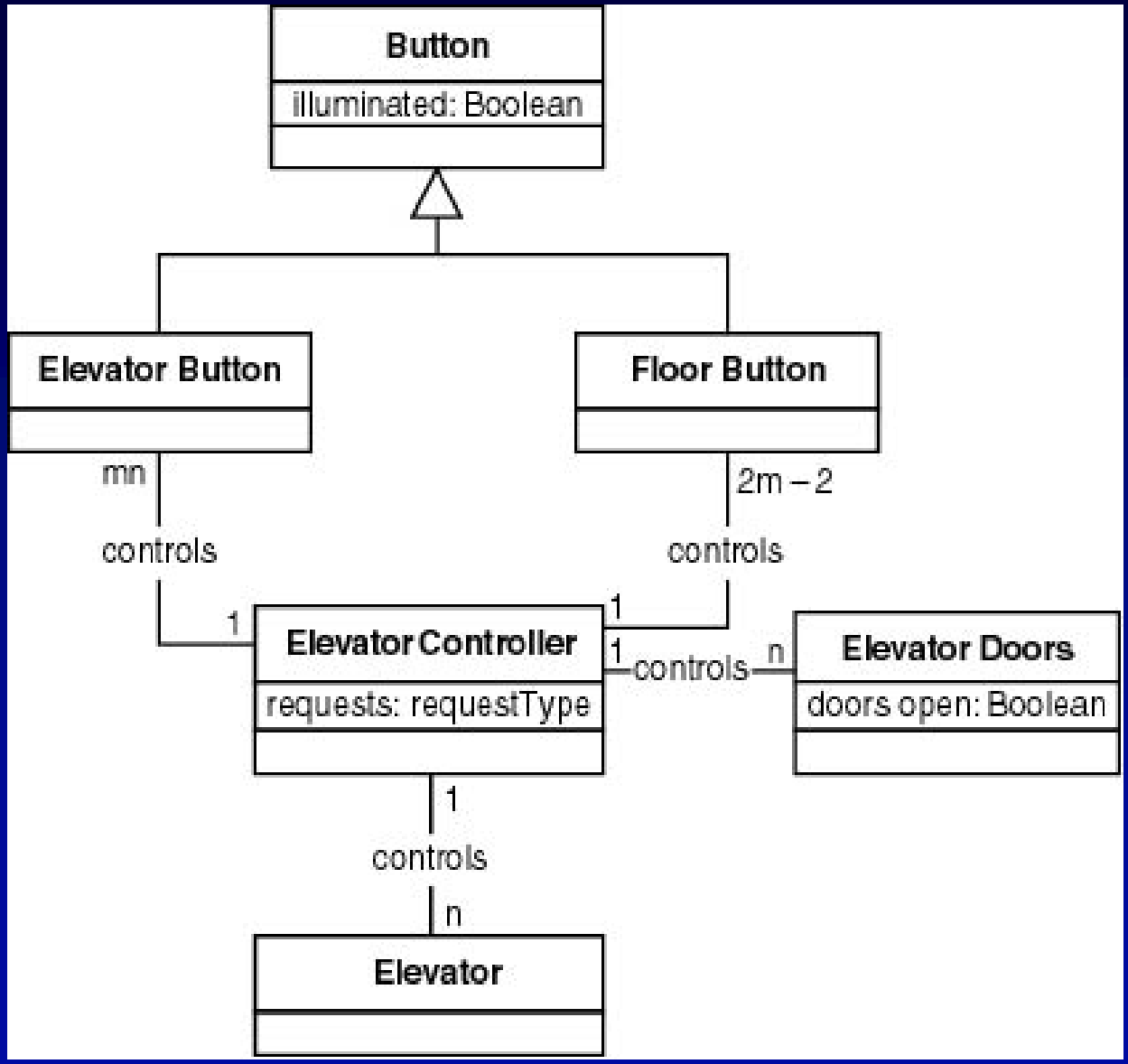
CRC Cards

- A class has been overlooked
  - Elevator doors have a *state* that changes during execution (class characteristic)
  - Add class **Elevator Doors**
  - Safety considerations
- Need to
  - Reconsider class structures
  - Reconsider use cases and dynamic behavior

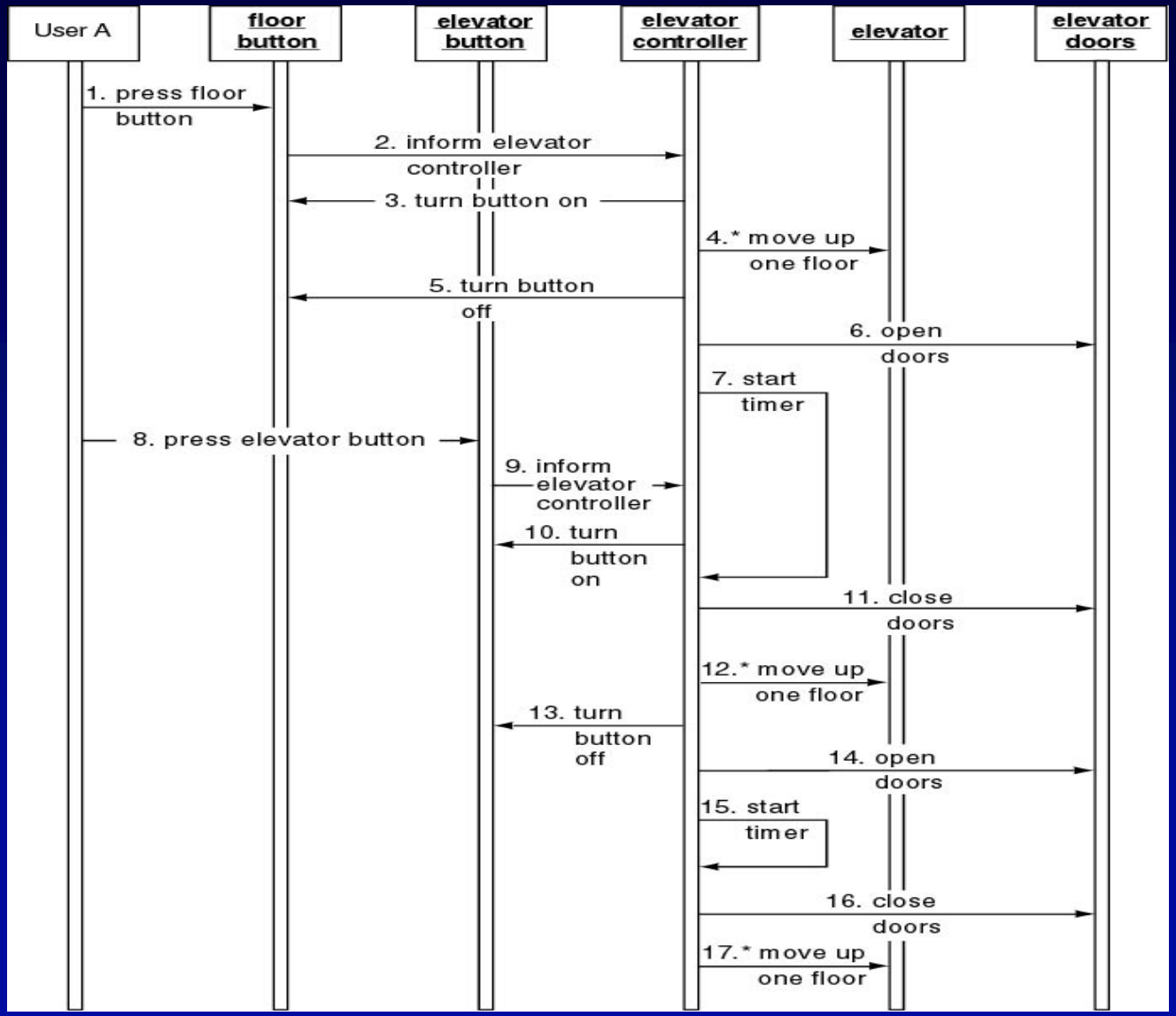
Second Iteration of CRC Card

CLASS **Elevator Controller**
RESPONSIBILITY 1. Send message to **Elevator Button** to turn on button 2. Send message to **Elevator Button** to turn off button 3. Send message to **Floor Button** to turn on button 4. Send message to **Floor Button** to turn off button 5. Send message to **Elevator** to move up one floor 6. Send message to **Elevator** to move down one floor 7. Send message to **Elevator Doors** to open 8. Start timer 9. Send message to **Elevator Doors** to close after timeout 10. Check requests 11. Update requests
COLLABORATION 1. Subclass **Elevator Button** 2. Subclass **Floor Button** 3. Class **Elevator Doors** 4. Class **Elevator**

Third Iteration of Class Diagram



Sequence Diagram



- Sequence diagram is used to model flows of control by time ordering

Detailed Class Diagram

- Do we assign an action to a class or to a client of that class?
- Criteria
  - Information hiding
    - Actions performed on state variables should be local
  - Reducing number of copies of action
    - Multiple clients, a single action
  - Responsibility-driven design
    - If a client sends a message to an object, the object is responsible for carrying out the request of the client
    - The client does not know how it is carried out and it not allowed to know
- Examples

  close doors is assigned to **Elevator Doors**

  move one floor down is assigned to **Elevator**

Detailed Class Diagram

- Add actions to the class diagram