CS342 Software Engineering

Dr. Ismail Hababeh
German Jordanian University

Lecture 13
OBJECT- ORIENTED ANALYSIS

Adapted from Software Engineering, by Dr. Paul E. Young & slides by Dr. Mohammad Daoud

Object - Oriented Analysis

- Extracting the entity classes
- Extracting the boundary classes
- Extracting the control classes
- The specification document in the Unified Process
- More on actors and use cases
- CASE tools for the object-oriented analysis
- Challenges of the object-oriented analysis

Object - Oriented Analysis Overview

- The Elevator problem case study
  - Object-Oriented analysis
  - Functional modeling
  - Entity class modeling
  - Dynamic modeling

Object - Oriented Analysis Overview

- The MSG Foundation case study
  - The initial functional model
  - The initial class diagram
  - The initial dynamic model
  - Extracting the entity classes
  - Extracting the boundary classes
  - Extracting the control classes
  - Use-case realization
  - Incrementing the class diagram

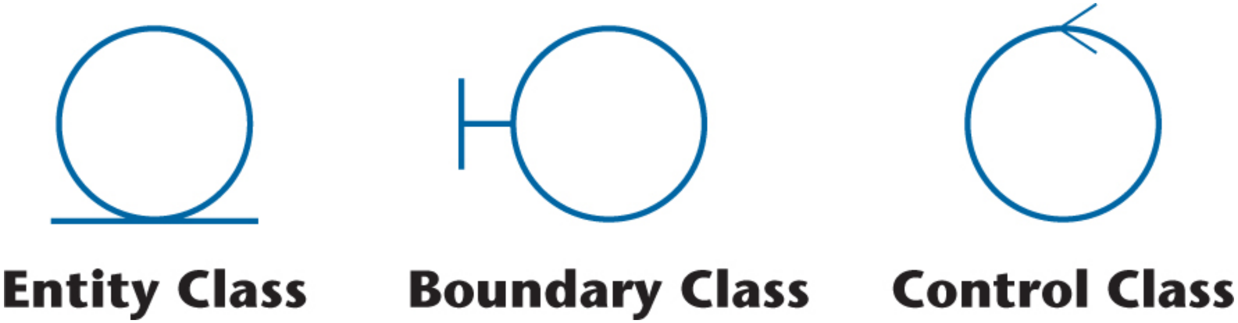
Object - Oriented Analysis (OOA)

- OOA is a semiformal analysis technique
  - There are many equivalent techniques
  - Unified Process is the only feasible alternative
- During OOA workflow, the following **classes are extracted**
  - Entity classes
  - Boundary classes
  - Control classes

Object-Oriented Analysis – Classes Extraction

- Entity class
  - Models long-live information
  - Examples: **Account Class, Investment Class**
- Boundary class
  - Models the interaction between the product and the environment
  - Associated with input/output
  - Examples: **Investments Report Class, Mortgages Report Class**
- Control class
  - Models the complex computations and algorithms
  - Example: **Estimate Funds for Week Class**

UML Notation of the Object-Oriented Classes



Extracting the Entity Classes

- Perform the following three steps incrementally and iteratively

  Step 1: Functional modeling

  - Extract scenarios of all the use cases (a *scenario* is an instance of a use case)

  Step 2: Class (diagram) modeling

  - Determine the entity classes and their attributes
  - Determine the interrelationships and interactions between the entity classes
  - Draw this information in the form of a *class diagram*

  Step 3: Dynamic modeling (UML statechart)

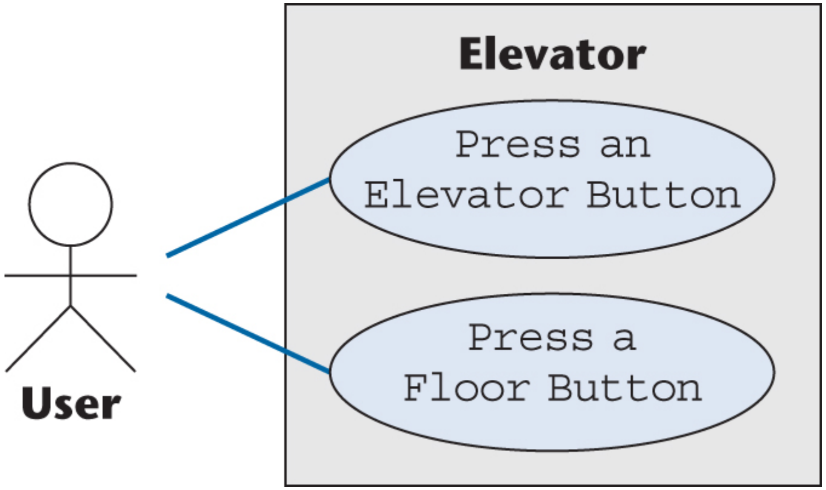
  - Determine the operations performed by/to each entity class
  - Draw this information in the form of a UML *statechart*

Step 1 - Functional Modeling Use Cases vs. Scenarios

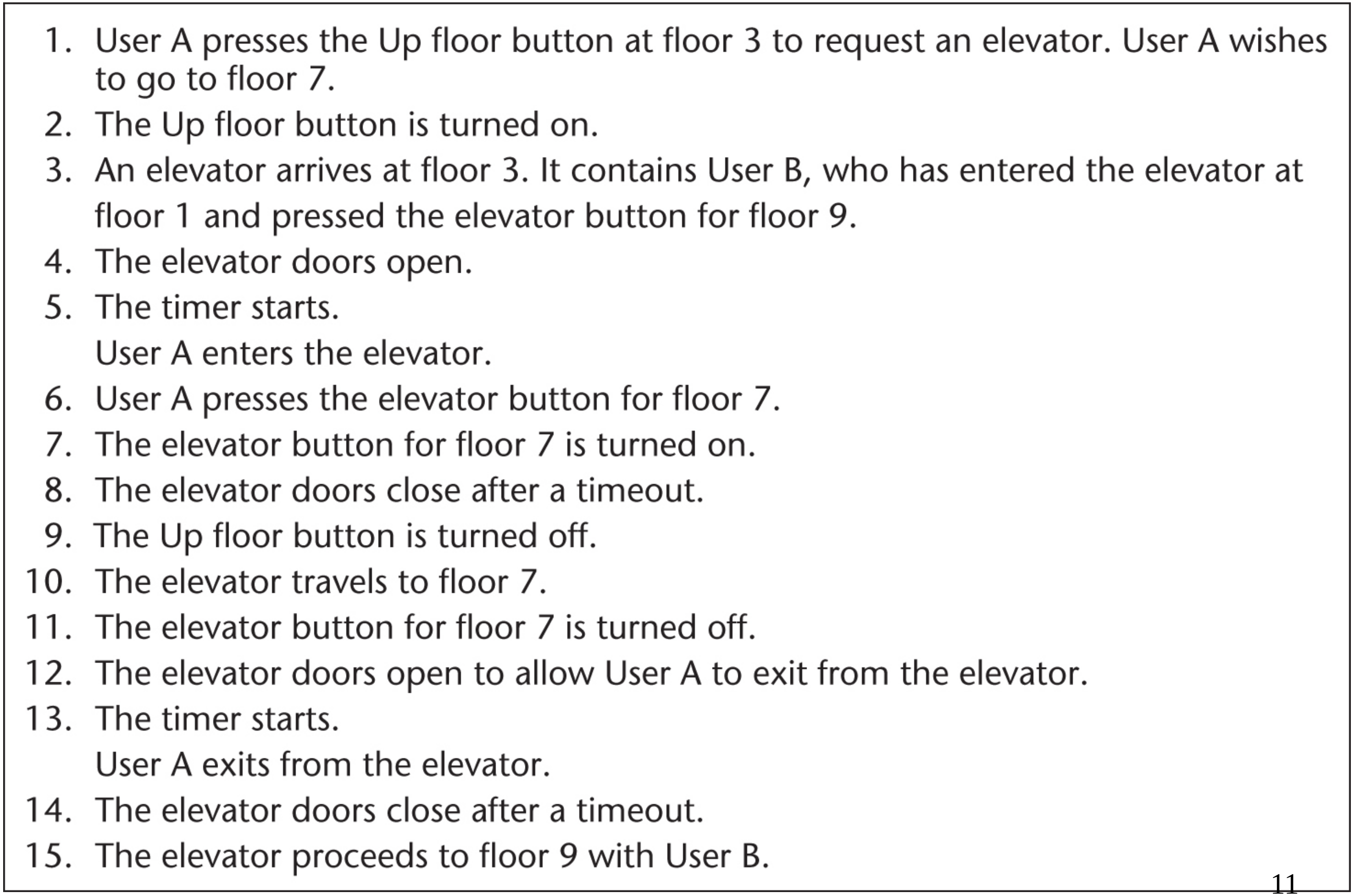
- A use case provides a generic description of the overall functionality
- A scenario is an instance of a use case
- Sufficient scenarios are needed to get a comprehensive insight into the target product being modeled

Step 1 - Functional Modeling

- Example: Elevator case study
- There are only two possible use cases:
  - Pressing an Elevator Button
  - Pressing a Floor Button



Example: Elevator Scenario 1

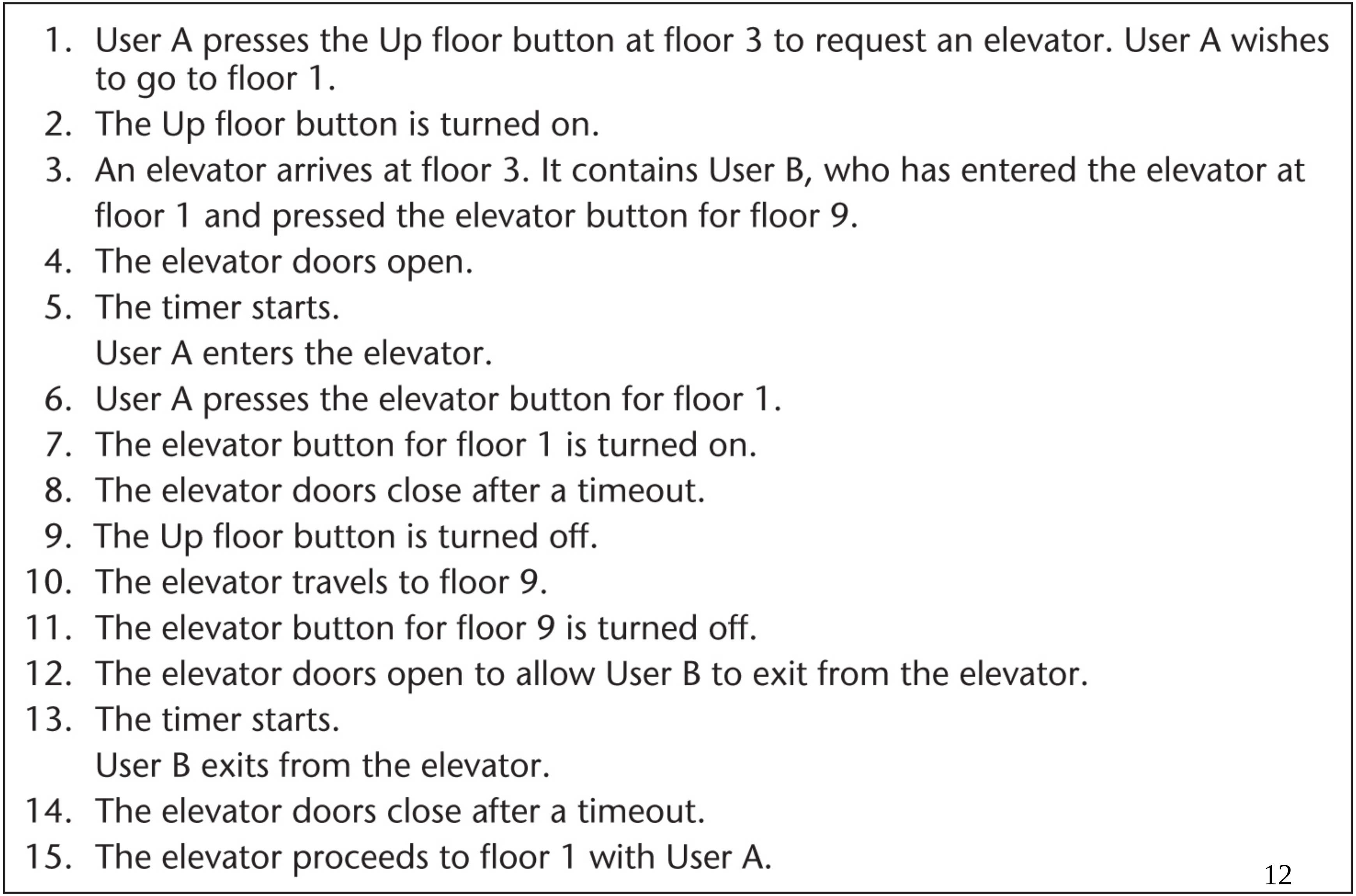


1. User A presses the Up floor button at floor 3 to request an elevator. User A wishes to go to floor 7.
2. The Up floor button is turned on.
3. An elevator arrives at floor 3. It contains User B, who has entered the elevator at floor 1 and pressed the elevator button for floor 9.
4. The elevator doors open.
5. The timer starts.

   User A enters the elevator.
6. User A presses the elevator button for floor 7.
7. The elevator button for floor 7 is turned on.
8. The elevator doors close after a timeout.
9. The Up floor button is turned off.
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.

Example: Elevator Scenario 2



1. User A presses the Up floor button at floor 3 to request an elevator. User A wishes to go to floor 1.
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 elevator doors open.
5. The timer starts.

   User A enters the elevator.
6. User A presses the elevator button for floor 1.
7. The elevator button for floor 1 is turned on.
8. The elevator doors close after a timeout.
9. The Up floor button is turned off.
10. The elevator travels to floor 9.
11. The elevator button for floor 9 is turned off.
12. The elevator doors open to allow User B to exit from the elevator.
13. The timer starts.

    User B exits from the elevator.
14. The elevator doors close after a timeout.
15. The elevator proceeds to floor 1 with User A.

Step 2 - Entity Class Modeling

- Extract (classes and their attributes) and represent them using a UML diagram.
  - Main method: extract the classes from use cases and their scenarios
    - Possible risks:
      - Many scenarios
      - Too many candidate classes
  - Alternatives:
    1. Noun extraction
    2. Class Responsibility Collaboration CRC cards (if you have domain knowledge)

Entity Class Modeling - Noun Extraction

- A two-phase process:

1. Brief problem definition

   - Describe the **software** product in a single paragraph.

   Example: “*Buttons in elevators and on the floors control the movement of n elevators in a building with m floors. Buttons illuminate when pressed to request the elevator to stop at a specific floor; the 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.*”

Entity Class Modeling - Noun Extraction

2. Identify the nouns:
   - Identify the nouns and use them as candidate classes

Example: *“Buttons in elevators and on the floors control the movement of n elevators in a building with m floors. Buttons illuminate when pressed to request the elevator to stop at a specific floor; the 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.”*

Entity Class Modeling - Noun Extraction

- Nouns

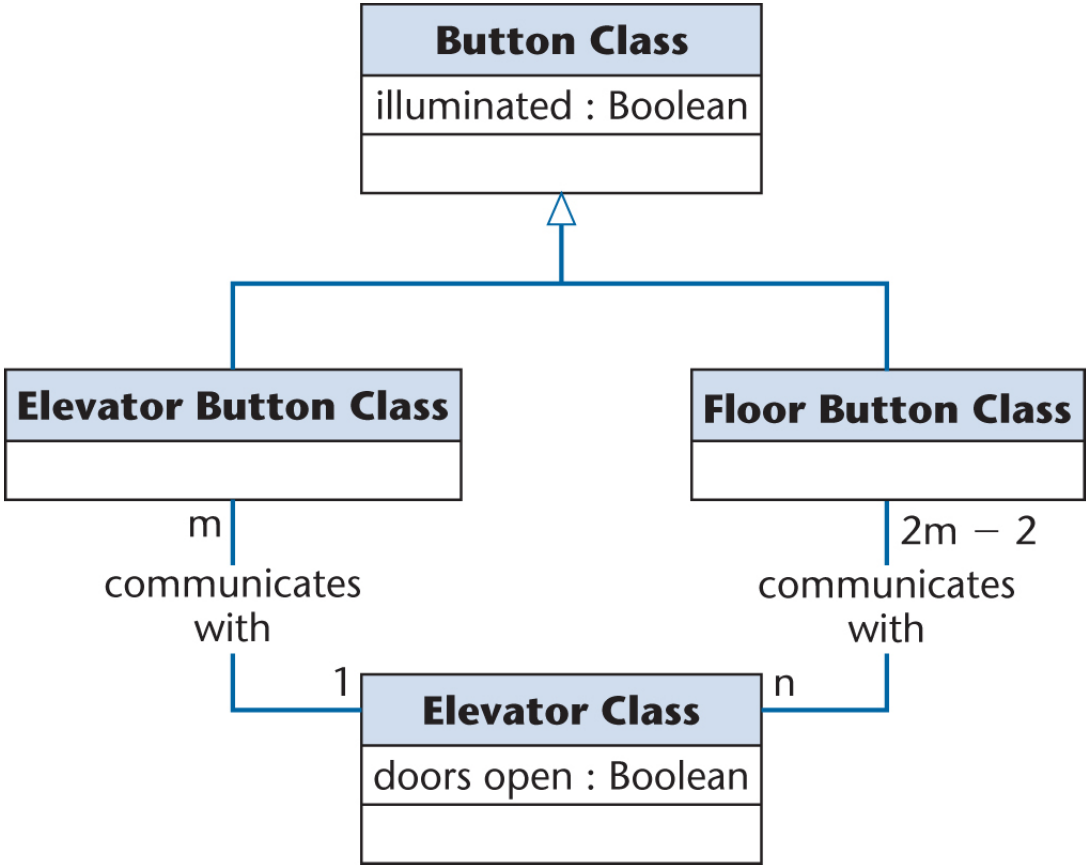
  button, elevator, floor, movement, building, illumination, request, door

  - Outside problem boundary nouns (floor, building, door) are excluded.
  - Abstract nouns (movement, illumination, request) are excluded (they may become attributes)

- Candidate classes:
  - Elevator
  - Button

- Subclasses (of Button class):
  - Elevator Button
  - Floor Button

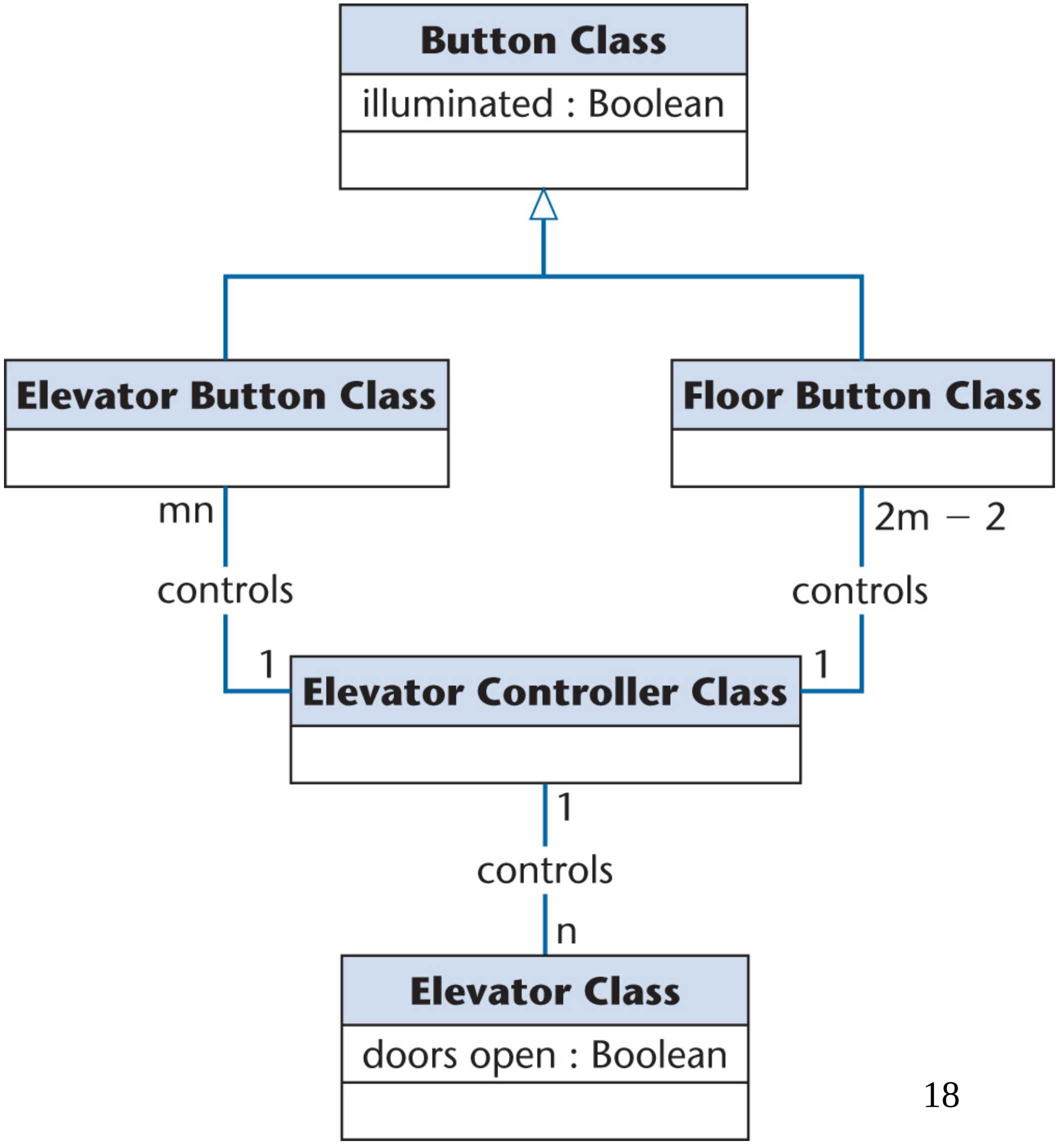
Entity Class Diagram – Iteration 1



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

Entity Class Diagram – Iteration 2

All relationships are now 1-to-n (this makes design and implementation easier)

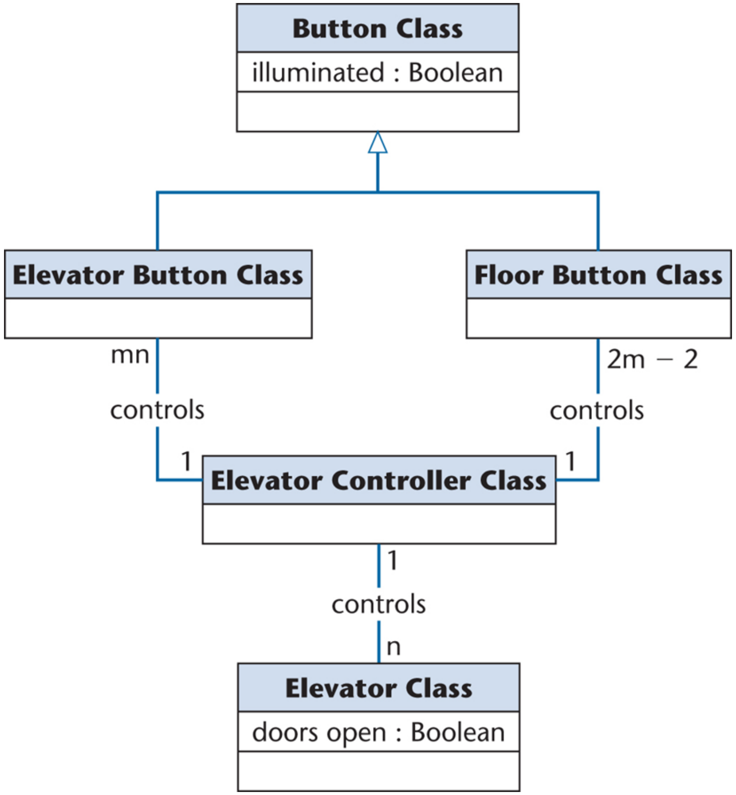


Entity Class Modeling - CRC
Class Responsibility Collaboration Cards

- Used for Object Oriented Analysis
- For each class, fill in a card showing
  - Class name
  - Functionality (Responsibility)
  - List of classes it invokes (Collaboration)
- Strength: When acted out by team members, CRC cards are a powerful tool for highlighting missing or incorrect items.
- Weakness: If CRC cards are used to identify entity classes, domain expertise is needed.

Example: CRC Controller Class – Iteration 1

- CRC cards are useful testing technique



CLASS

Elevator Controller Class

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. **Elevator Button Class**
2. **Floor Button Class**
3. **Elevator Class**

Object-Oriented Analysis - Test Workflow

- Consider (Turn on elevator button) responsibility:

  This is totally inappropriate for the object-oriented paradigm

  - Responsibility-driven design has been ignored
  - Information hiding has been ignored

- Enhancement:

  Turn on elevator button responsibility should be changed to:

  Send message to → Elevator Button Class

  to turn itself on

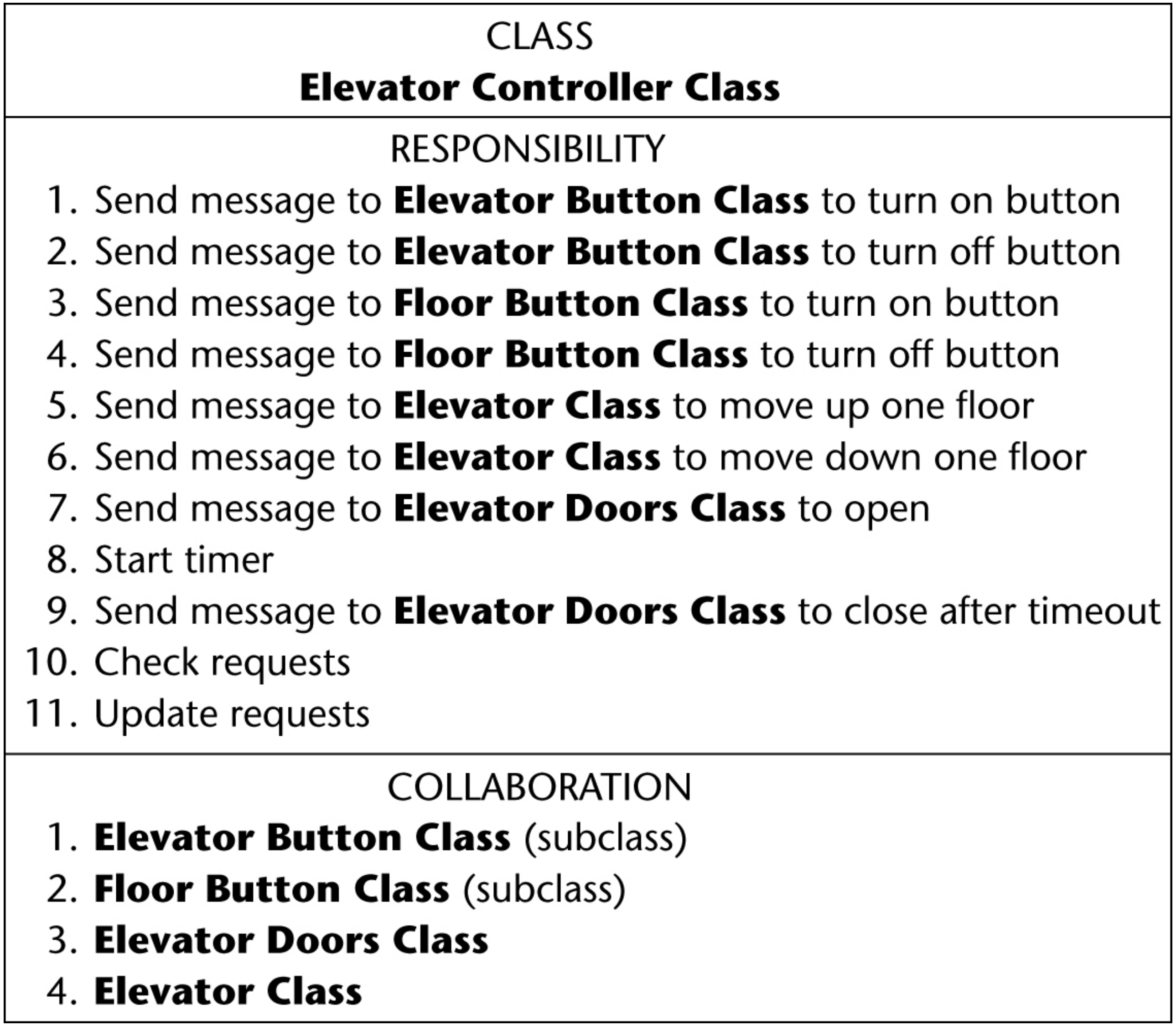
Object-Oriented Analysis - Test Workflow

- The elevator doors have a *state* that changes during execution (class characteristic)

⇨ Add **Elevator Doors Class**

- Modify the CRC card

Example: CRC Controller Class – Iteration 2



CLASS

Elevator Controller Class

RESPONSIBILITY

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

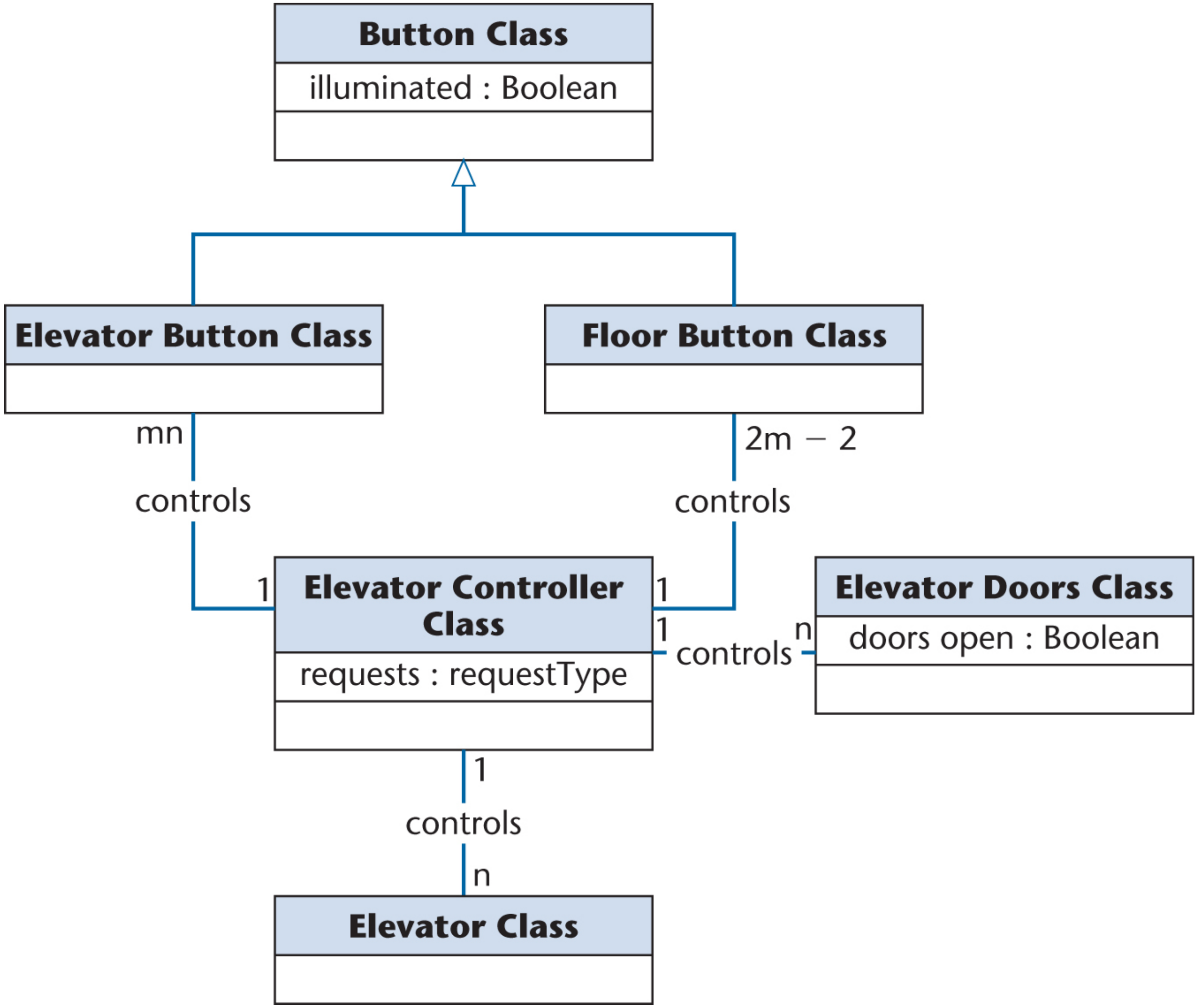
COLLABORATION

1. **Elevator Button Class** (subclass)
2. **Floor Button Class** (subclass)
3. **Elevator Doors Class**
4. **Elevator Class**

CRC Cards Modification

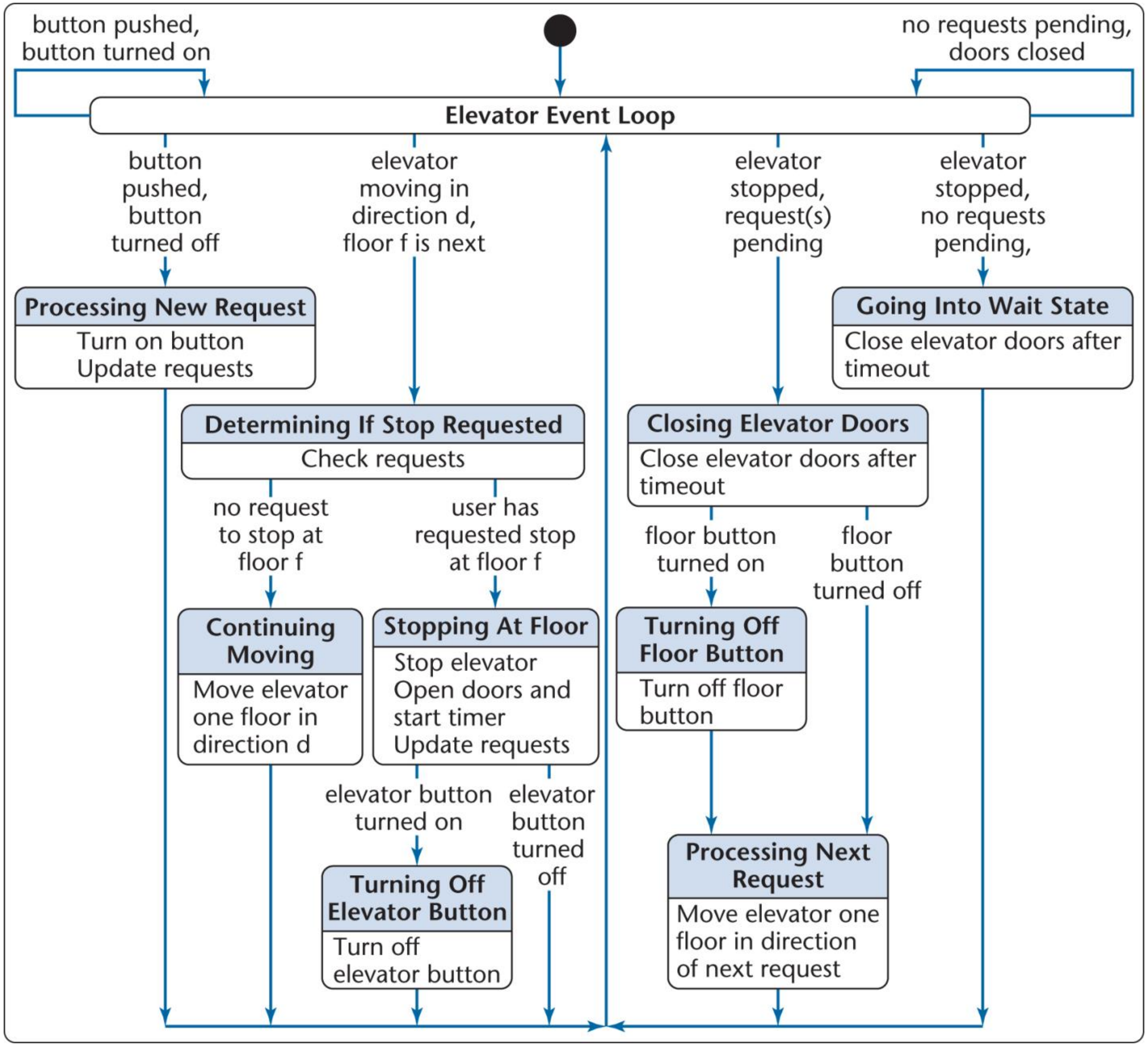
- Having modified the class diagram, reconsider the following:
  - Scenarios
  - Use-case diagram (general and no change)
  - UML Statecharts: describes the flow of control from one state to another. It determines the states, events, and predicates of the dynamic modeling.

Modified Entity Class Diagram - Iteration 3



UML Statechart (Dynamic Modeling)

- **Produce UML statechart**
- States, events, and predicates are distributed over the statechart
- A statechart is constructed by modeling the events of the scenarios
- UML statechart is equivalent to the state transition diagram in static modeling.


button pushed, button turned on
no requests pending, doors closed
Elevator Event Loop
button pushed, button turned off
elevator moving in direction d, floor f is next
elevator stopped, request(s) pending
elevator stopped, no requests pending,
Processing New Request
Turn on button
Update requests
Going Into Wait State
Close elevator doors after timeout
Determining If Stop Requested
Check requests
Closing Elevator Doors
Close elevator doors after timeout
no request to stop at floor f
user has requested stop at floor f
floor button turned on
floor button turned off
Continuing Moving
Move elevator one floor in direction d
Stopping At Floor
Stop elevator
Open doors and start timer
Update requests
Turning Off Floor Button
Turn off floor button
elevator button turned on
elevator button turned off
Turning Off Elevator Button
Turn off elevator button
Processing Next Request
Move elevator one floor in direction of next request

Entity Class Diagram – Problems!

- **Elevator Controller Class** is running everything
  - – Exposed to too much information
  - – Has too much control
- **Solution:**

  Distributed (Decentralized) Architecture

  - – Distribute the control instead of having one central elevator controller

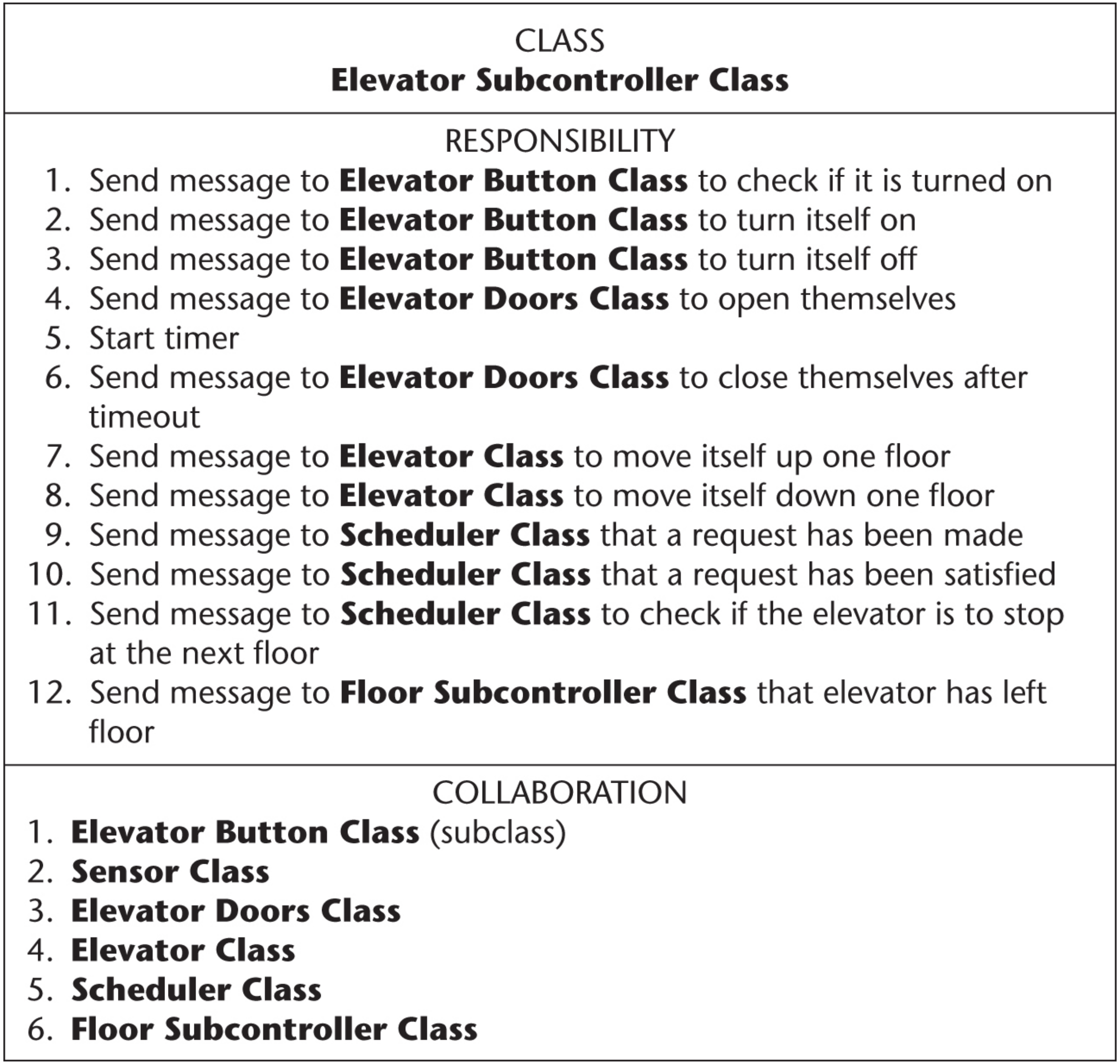
Distributed Controllers Architecture

- Each of the n elevators now has its own elevator sub-controller
- Each of the m floors now has its own floor sub-controller
- The (m + n) sub-controllers all communicate with a scheduler, which processes requests

Distributed Controllers Architecture

- A Floor Button Class is controlled by its corresponding Floor Sub-controller Class.
- An Elevator Button Class is controlled by its corresponding Elevator Sub-controller Class.
- There is a sensor just above and just below each floor in each elevator shaft.
- When an Elevator approaches or leaves a floor
  - The corresponding Sensor Class informs the corresponding Elevator Sub-controller Class.
- The class diagrams now need to be updated to reflect the fourth iteration of the class diagram.

CRC Subcontroller Class - Iteration 1



CLASS

Elevator Subcontroller Class

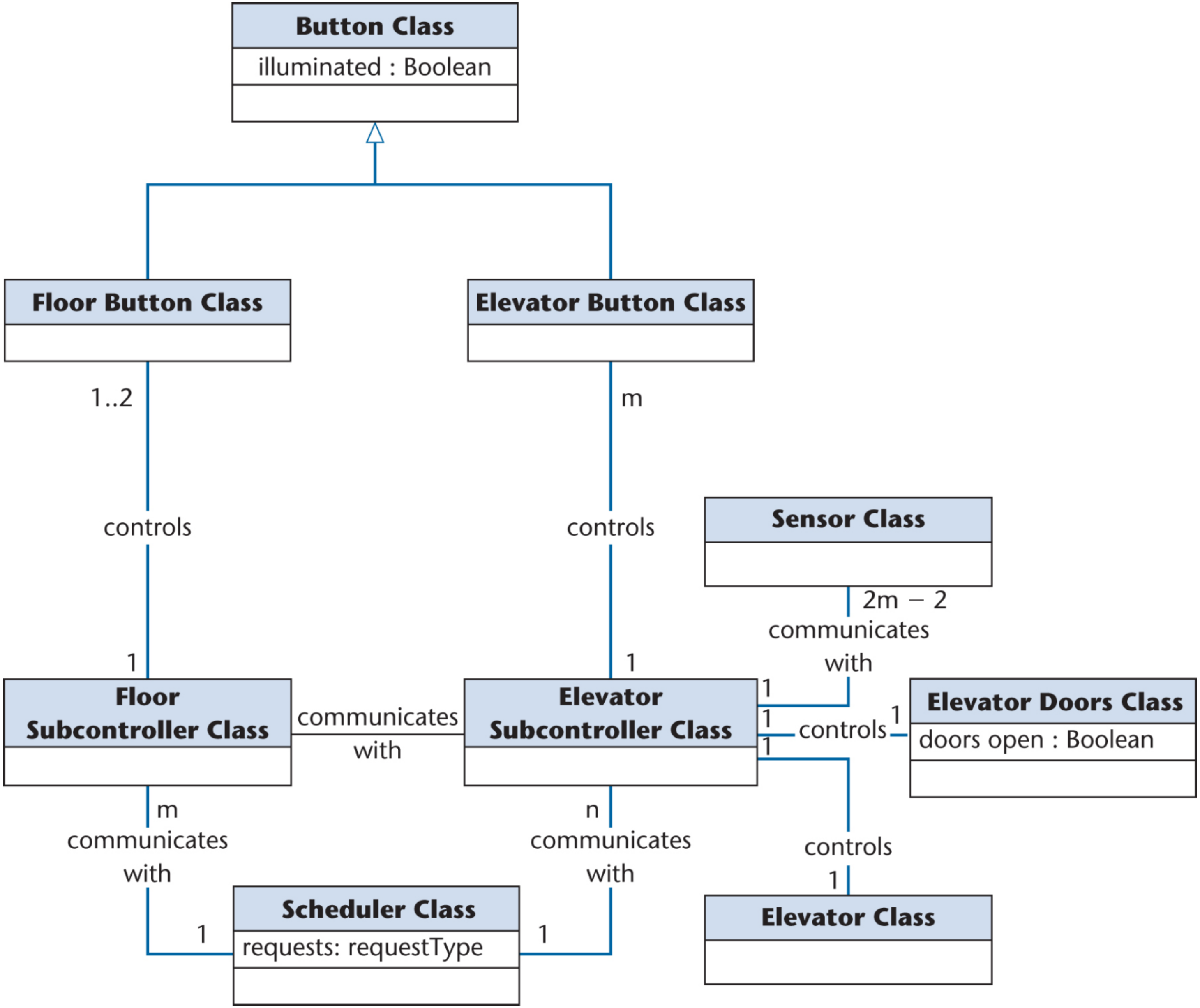
RESPONSIBILITY

1. Send message to **Elevator Button Class** to check if it is turned on
2. Send message to **Elevator Button Class** to turn itself on
3. Send message to **Elevator Button Class** to turn itself off
4. Send message to **Elevator Doors Class** to open themselves
5. Start timer
6. Send message to **Elevator Doors Class** to close themselves after timeout
7. Send message to **Elevator Class** to move itself up one floor
8. Send message to **Elevator Class** to move itself down one floor
9. Send message to **Scheduler Class** that a request has been made
10. Send message to **Scheduler Class** that a request has been satisfied
11. Send message to **Scheduler Class** to check if the elevator is to stop at the next floor
12. Send message to **Floor Subcontroller Class** that elevator has left floor

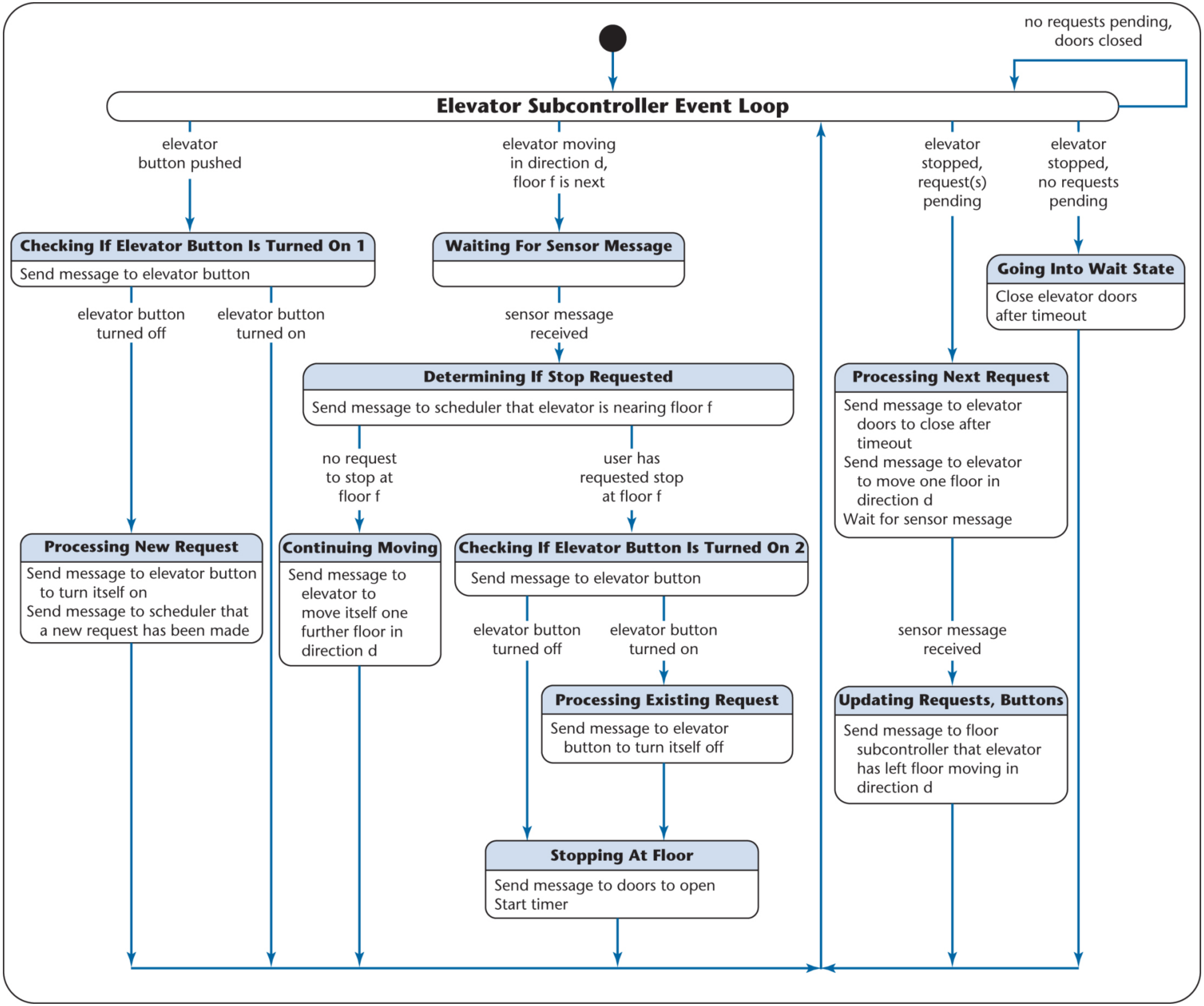
COLLABORATION

1. **Elevator Button Class** (subclass)
2. **Sensor Class**
3. **Elevator Doors Class**
4. **Elevator Class**
5. **Scheduler Class**
6. **Floor Subcontroller Class**

Entity Class Diagram – Iteration 4



Updated UML statechart



no requests pending, doors closed

Elevator Subcontroller Event Loop

elevator button pushed

Checking If Elevator Button Is Turned On 1

Send message to elevator button

elevator button turned off

elevator button turned on

elevator moving in direction d, floor f is next

Waiting For Sensor Message

sensor message received

Determining If Stop Requested

Send message to scheduler that elevator is nearing floor f

no request to stop at floor f

user has requested stop at floor f

Processing New Request

Send message to elevator button to turn itself on

Send message to scheduler that a new request has been made

Continuing Moving

Send message to elevator to move itself one further floor in direction d

Checking If Elevator Button Is Turned On 2

Send message to elevator button

elevator button turned off

elevator button turned on

Processing Existing Request

Send message to elevator button to turn itself off

Stopping At Floor

Send message to doors to open

Start timer

elevator stopped, request(s) pending

Processing Next Request

Send message to elevator doors to close after timeout

Send message to elevator to move one floor in direction d

Wait for sensor message

sensor message received

Updating Requests, Buttons

Send message to floor subcontroller that elevator has left floor moving in direction d

elevator stopped, no requests pending

Going Into Wait State

Close elevator doors after timeout