Development Methodologies

Prof. Dr. Josef M. Joller
jjoller@hsr.ch

OBJECT-ORIENTED ANALYSIS PHASE

Overview

Object-oriented analysis

Use-case modeling

Class modeling

Dynamic modeling

Testing during the object-oriented analysis phase

CASE tools for the object-oriented analysis phase

Object-Oriented Analysis Phase

Object-oriented paradigm

- Reaction to perceived shortcomings in structured paradigm
- Problem of larger products
- Data and action treated as equal partners

Object consists of

- Data (attributes, state variables, instance variables, fields, data members), and
- Actions (methods, member functions)

Objects are independent units

- Conceptual independence
- Physical independence

Object-Oriented Analysis (contd)

Semi-formal specification technique

Multiplicity of different methods

- Booch
- Rumbaugh & al (GE Labs) : OMT
- Jacobson (Objectory / Ericsson) : Objectory
- Shlaer-Mellor
- Coad-Yourdon

All essentially equivalent

Nowadays, we represent OOA using UML (unified modeling language)

The Three Steps of OOA

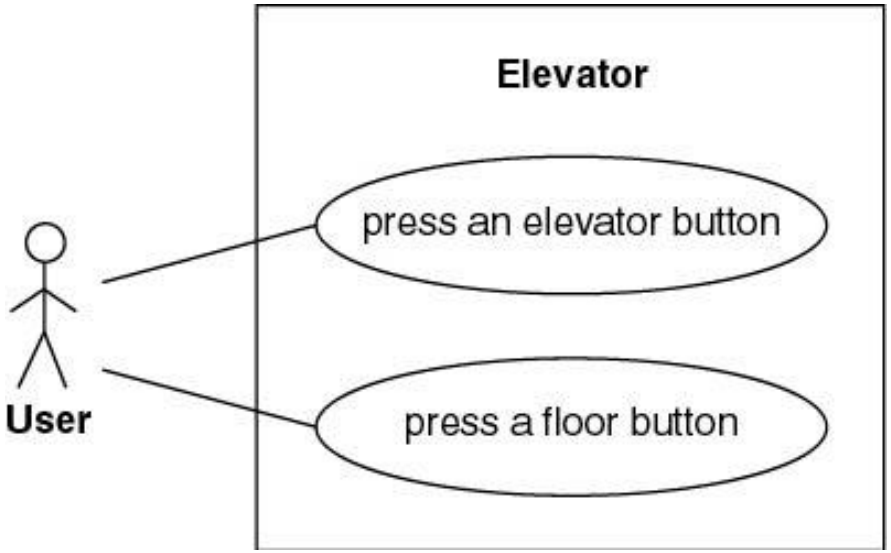
1. **Use-case modeling**
   - Determine how the various results are computed by the product (without regard to sequencing)
   - Largely action oriented
2. **Class modeling ("object modeling")**
   - Determine the classes and their attributes
   - Purely data-oriented
3. **Dynamic modeling**
   - Determine the actions performed by or to each class
   - Purely action-oriented

Iterative process

Elevator Problem: OOA

1. Use-Case Modeling

- Use case: Generic description of overall functionality



- Scenario: Instance of a use case

Get comprehensive insight into behavior of product

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.

Exception Scenario

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 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 1.
8. The elevator button for floor 1 is turned on.
9. The elevator doors close after a timeout.
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.

Class Modeling

Deduce the classes from use cases and their scenarios

Extract classes and their attributes

Represent them using an entity-relationship diagram

Often there are many scenarios

- Possible danger: too many candidate classes

Two Approaches to Class Modeling

Noun extraction

- Always works
- Process
  - Analyze text document : nouns are class candidates

CRC classes

- Need to have domain expertise
- Easy to implement (nearly no cost for tools)
- Team effort
- Need coach!

Noun Extraction

Stage 1. Concise Problem Definition

- Define product in single sentence
  - Buttons in elevators and on the floors control the motion of n elevators in a building with m floors.

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.

Stage 3. Formalize the Strategy

- Identify nouns in informal strategy. Use nouns as candidate classes

Noun Extraction (contd)

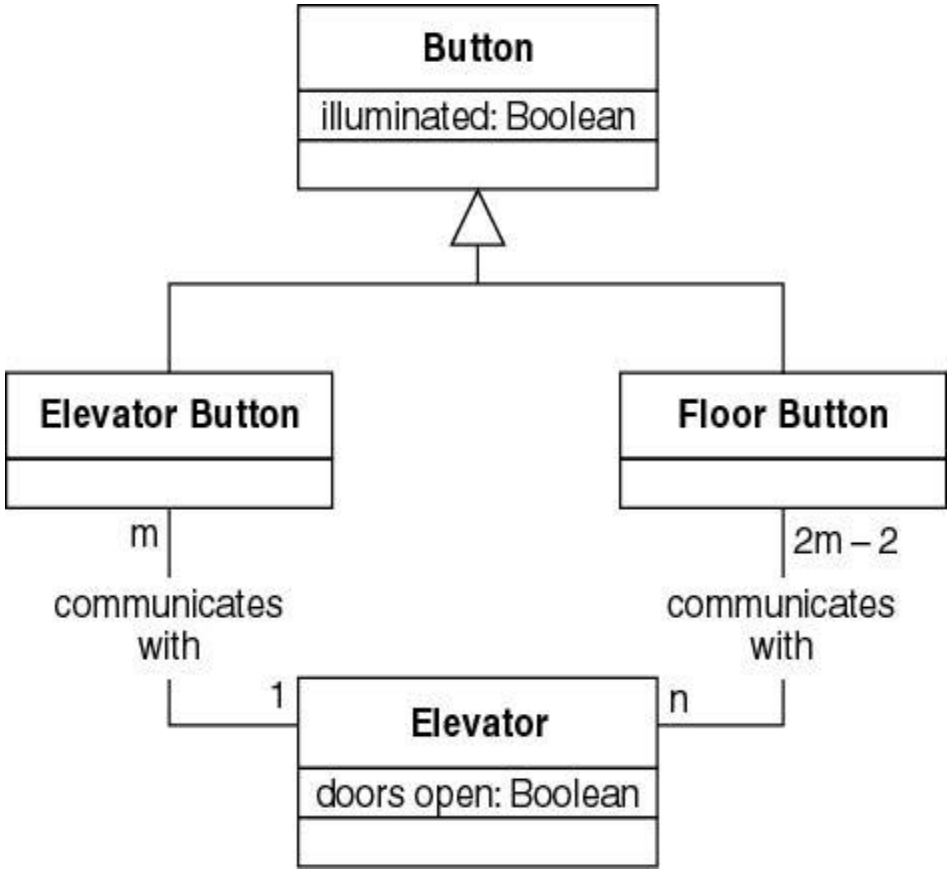
Nouns

- button, elevator, floor, movement, building, illumination, illumination, door
- floor, building, door are outside problem boundary — exclude
- movement, illumination, illumination are abstract nouns — exclude (may become attributes)

Candidate classes: Elevator **and** Button

Subclasses: Elevator Button **and** Floor Button

First Iteration of Class Diagram



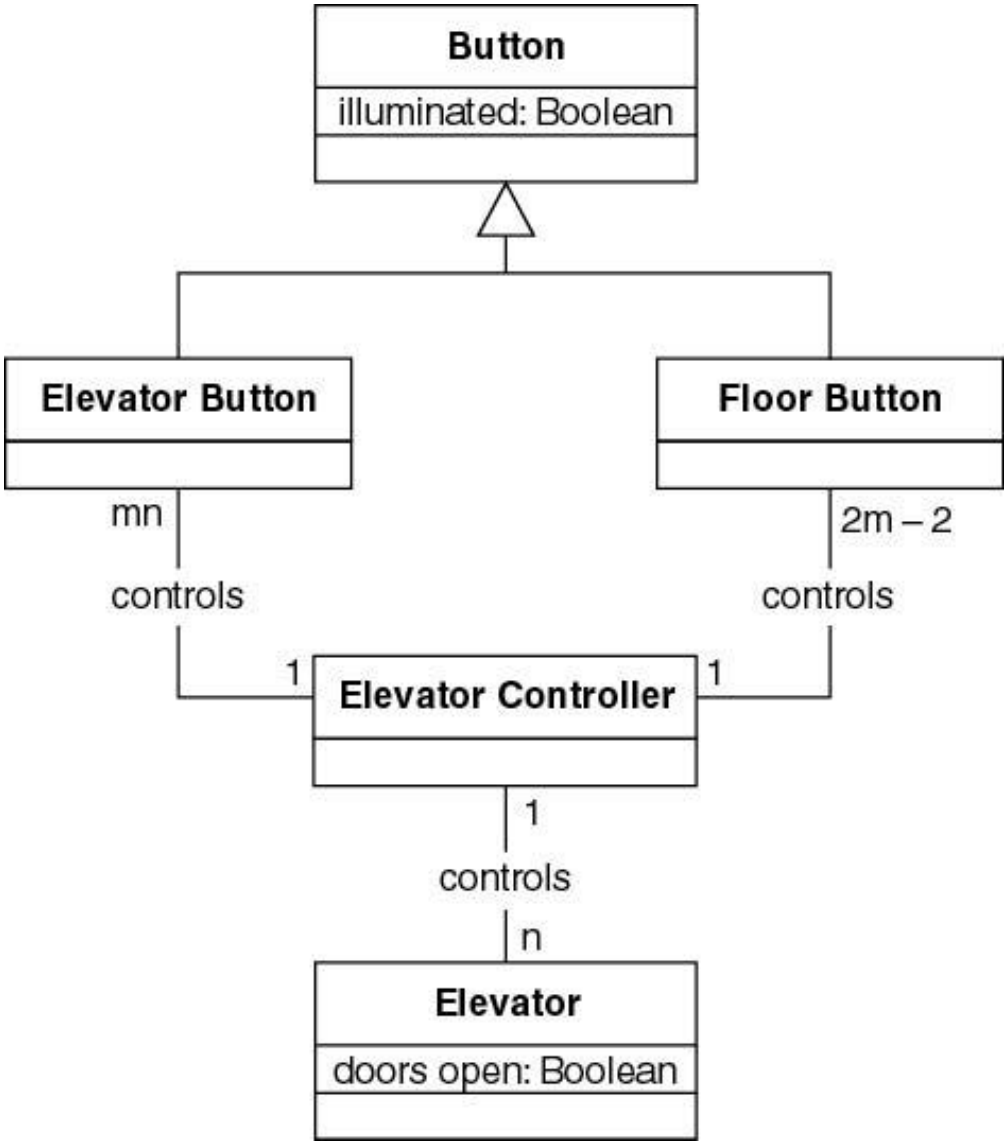
Problem

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

Second Iteration of Class Diagram

All relationships are now 1-to-n

- Makes design and implementation easier



CRC Cards

Used since 1989 for OOA

For each class, fill in card showing

- Name of class
- Functionality (responsibility)
- List of classes it invokes (collaboration)
- Now automated (CASE tool component)

Strength

- When acted out by team members, powerful tool for highlighting missing or incorrect items

Weakness

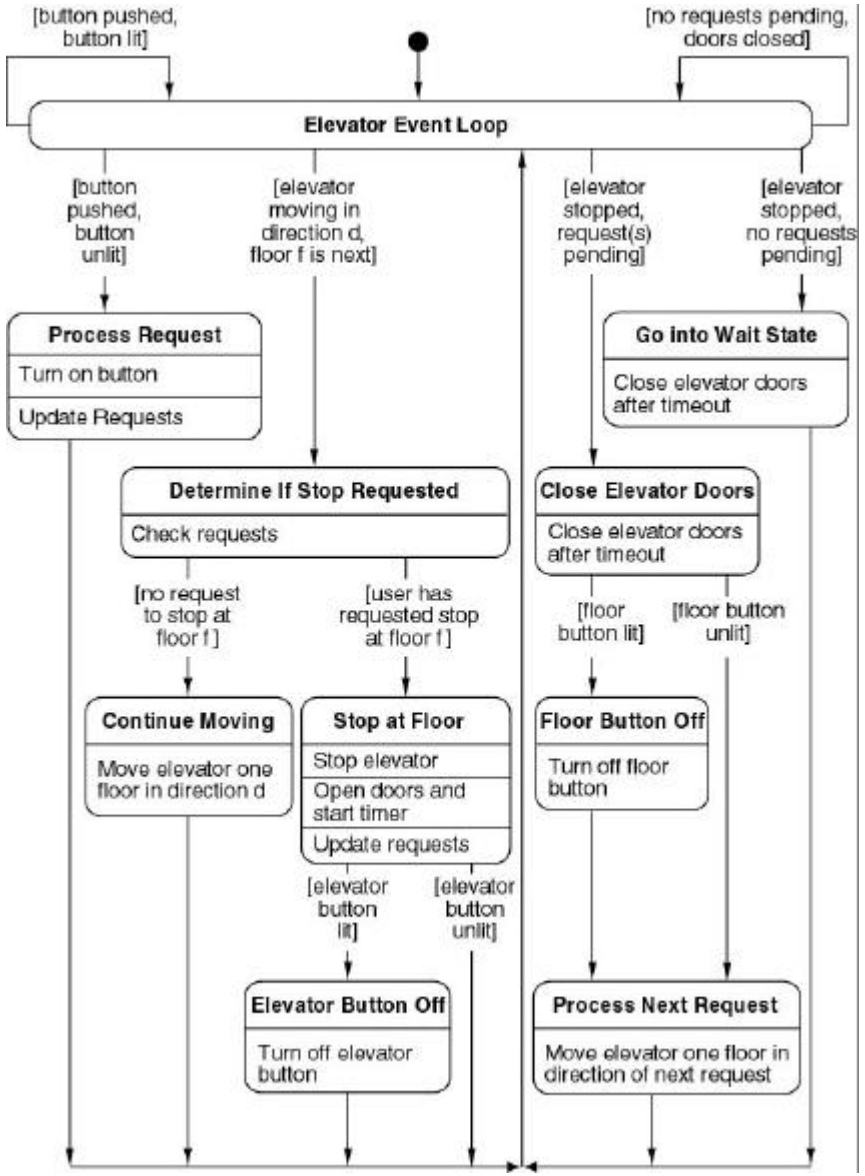
- Domain expertise is needed

3. Dynamic Modeling

Produce UML state diagram

State, event, predicate distributed over state diagram

UML "guards" are in brackets



CRC Cards

A class has been overlooked

- Elevator doors have a *state* that changes during execution (class characteristic)
- Add class **Elevator Doors**
- Safety considerations

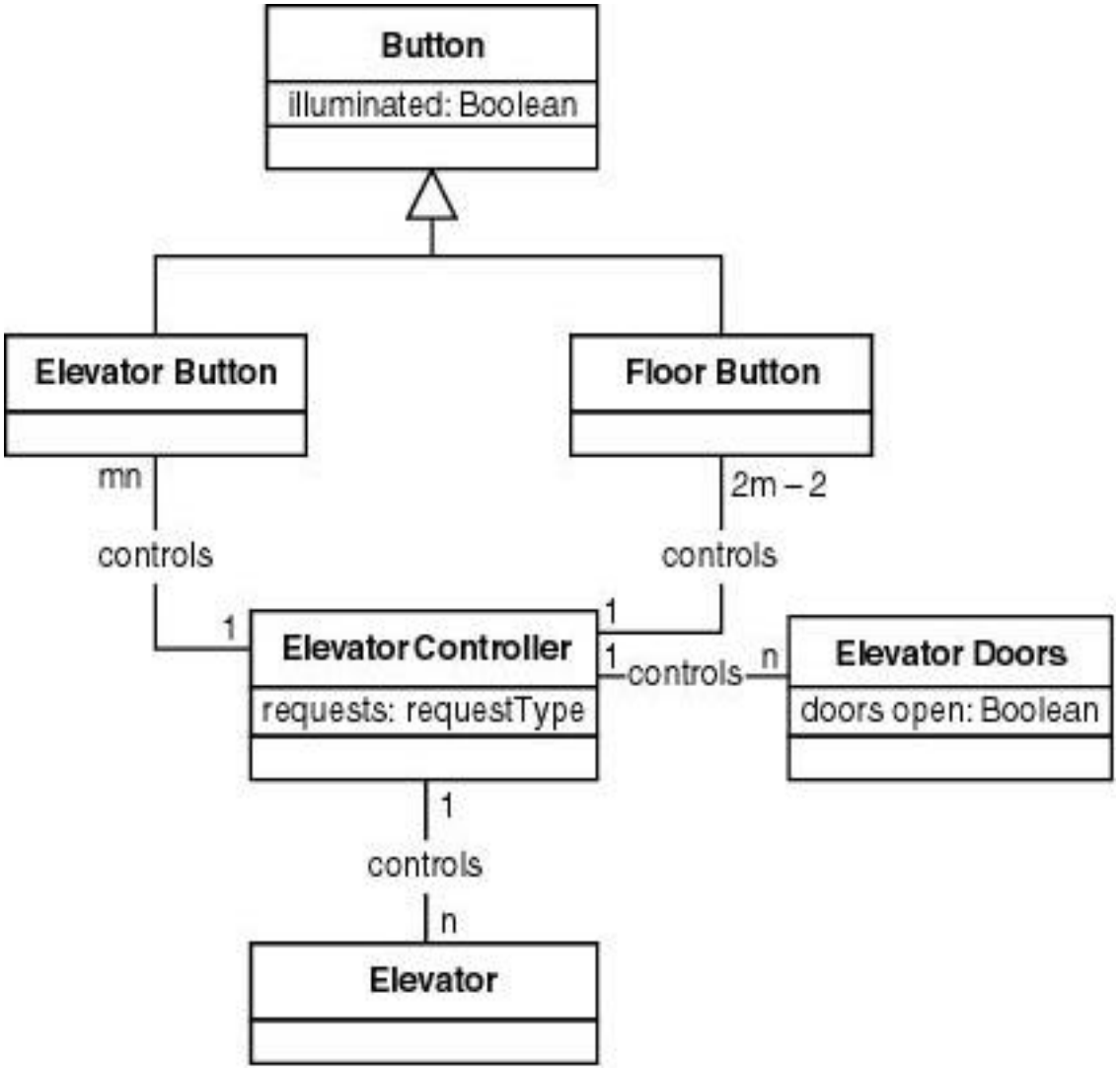
Reconsider class model

Then reconsider dynamic model, use-case model

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



Why Is All This Iteration Needed?

Perhaps the method is not yet mature?

- Waterfall model (explicit feedback loops)
- Rapid prototyping model (aim: to reduce iteration)
- Incremental model, and
- Spiral model

Latter two explicitly reflect iterative approach

Iteration is an intrinsic property of all software production

- Especially for medium- and large-scale products
- Expect iteration in the object-oriented paradigm

CASE tools for OOA phase

Diagrams play a major role

Diagrams often change

- Need a diagramming tool
- Many tools go further

All modern tools support UML

- Example
  - Rose
  - JTogether
  - …
  - Visio
  - …