Entwicklungsmethoden

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

OBJEKT-ORIENTIERTE ANALYSE PHASE

Übersicht

Objekt-orientierte Analyse

Use-case Modellierung

Klassen Modellierung

Dynamische Modellierung

Testen in der Objekt-orientierten Analyse Phase

CASE Tools für die Objekt-orientierte Analyse Phase

Objekt-Orientierte Analyse Phase

Objekt-orientiertes Paradigma

- ist eine Reaktion auf die Schwächen des strukturierten Paradigmas
  - Probleme in Grossprojekten
- bei Objekten werden Daten und die Dynamik gleichwertig betrachtet

Objekte bestehen aus

- Daten (Attribute, Zustandsvariablen, Instanzenvariablen, Feldern, Data Members : unterschiedliche Namen für das selbe) und
- Aktionen (Methoden, Member Functions : unterschiedliche Namen für das selbe)

Objekte sind unabhängige Grössen

- Objekte sind konzeptuell unabhängig
- Objekte sind physisch unabhängig

Objekt-Orientierte Analyse

Verwendet semi-formale Techniken

Eine Vielzahl von Techniken wurde definiert / entwickelt:

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

Alle sind ähnlich / äquivalent

Der heutige Standard zur Dokumentation ist die UML (unified modeling language)

Ein dazu passender Prozess ist der Rational Unified Process

Die drei Stufen der OOA

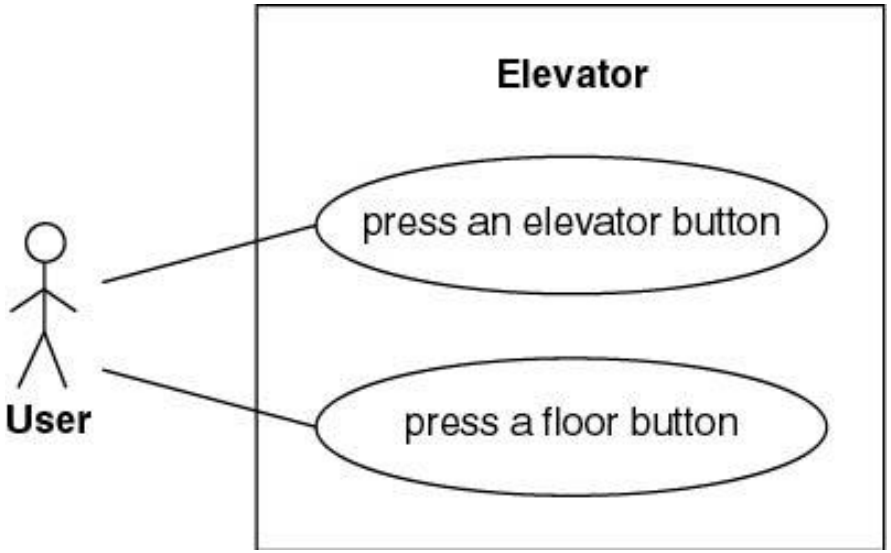
1. **Use-Case Modellierung**
   - Grobbeschreibung der Funktionen
   - im Wesentlichen Aktions-orientiert
2. **Class Modellierung ("object modeling")**
   - Bestimmen der Klassen und Attribute
   - möglichst bestimmen der Daten
3. **Dynamische Modellierung**
   - bestimmen der Methoden / Aktionen jedes Objekts, jeder Klasse
   - Aktions-orientiert

Iterativer Prozess

Lift: OOA

1. Use-Case Modeling

- Use case: generische Beschreibung der Funktionalität



- Scenario als Instanz des Use Cases

Ziel: eine Übersicht erhalten!

Primärszenario

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.

Sekundärszenario

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.

Klassen Modellierung

Herleiten der Klassen aus den Szenarios und Use Cases

Extraktion der Klassen und Attribute

Darstellung als ER Diagramm

Oft müssen viele Szenarios untersucht werden

- Gefahr: oft hat man zuviele Szenarios / Use Cases

Zwei Techniken

Dingworte

- funktioniert immer
- Prozess
  - analysieren des Textes
  - Dingwörter sind Klassen-Kandidaten

CRC Klassen

- benötigt Domänen-Expertise
- kostengünstige Implementierung ist möglich
- stellt einen Team Effort dar
- benötigt einen Coach (...siehe meine Adresse... [Spass bei Seite])

Dingwörter extrahieren

Stufe 1. Erstellen einer genauen Beschreibung

- Definieren des Produktes möglichst in einem einzigen Satz
  - Buttons in elevators and on the floors control the motion of n elevators in a building with m floors.

Stufe 2. Informale Strategie

- jetzt beschreiben Sie auch noch Sonderfälle
  - 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.

Stufe 3. Formalisieren the Strategie

- Dingwörter sind Klassenkadidaten

Dingwörter

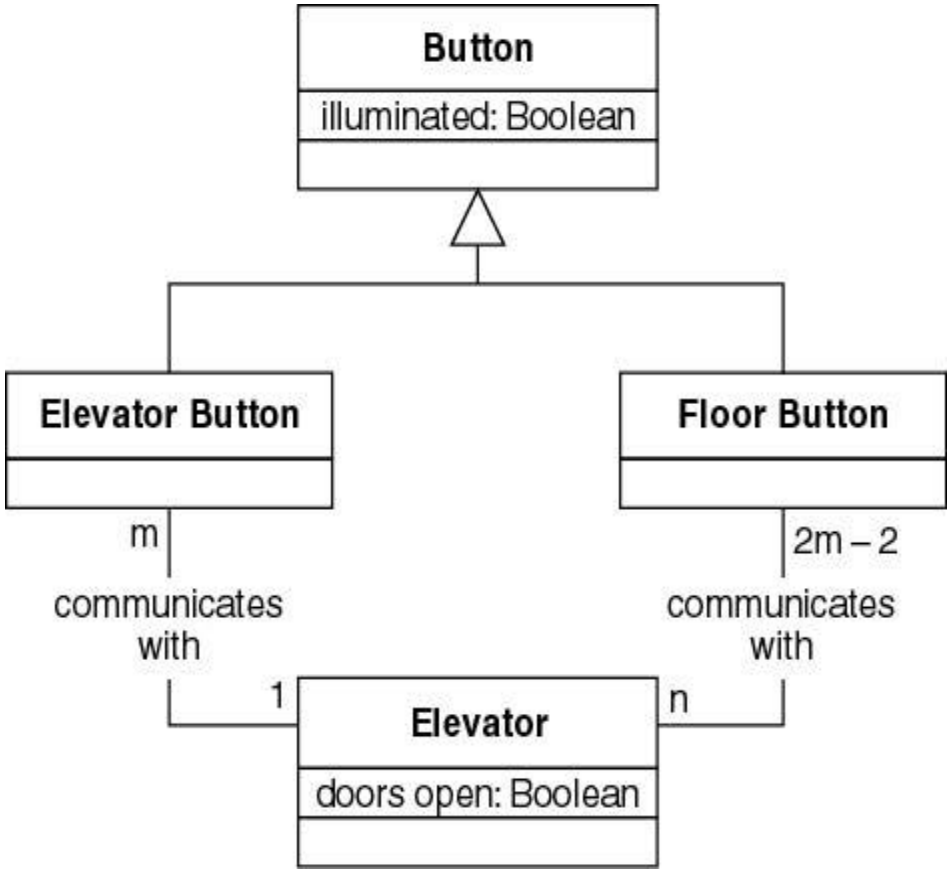
Dingwörter

- button, elevator, floor, movement, building, illumination, illumination, door
- floor, building, door
  gehören nicht zum System
- movement, illumination, illumination
  sind abstrakt, könnten eventuell Attribute werden

Klassenkandidaten: Elevator **und** Button

Subklasses: Elevator Button **und** Floor Button

Erste Version



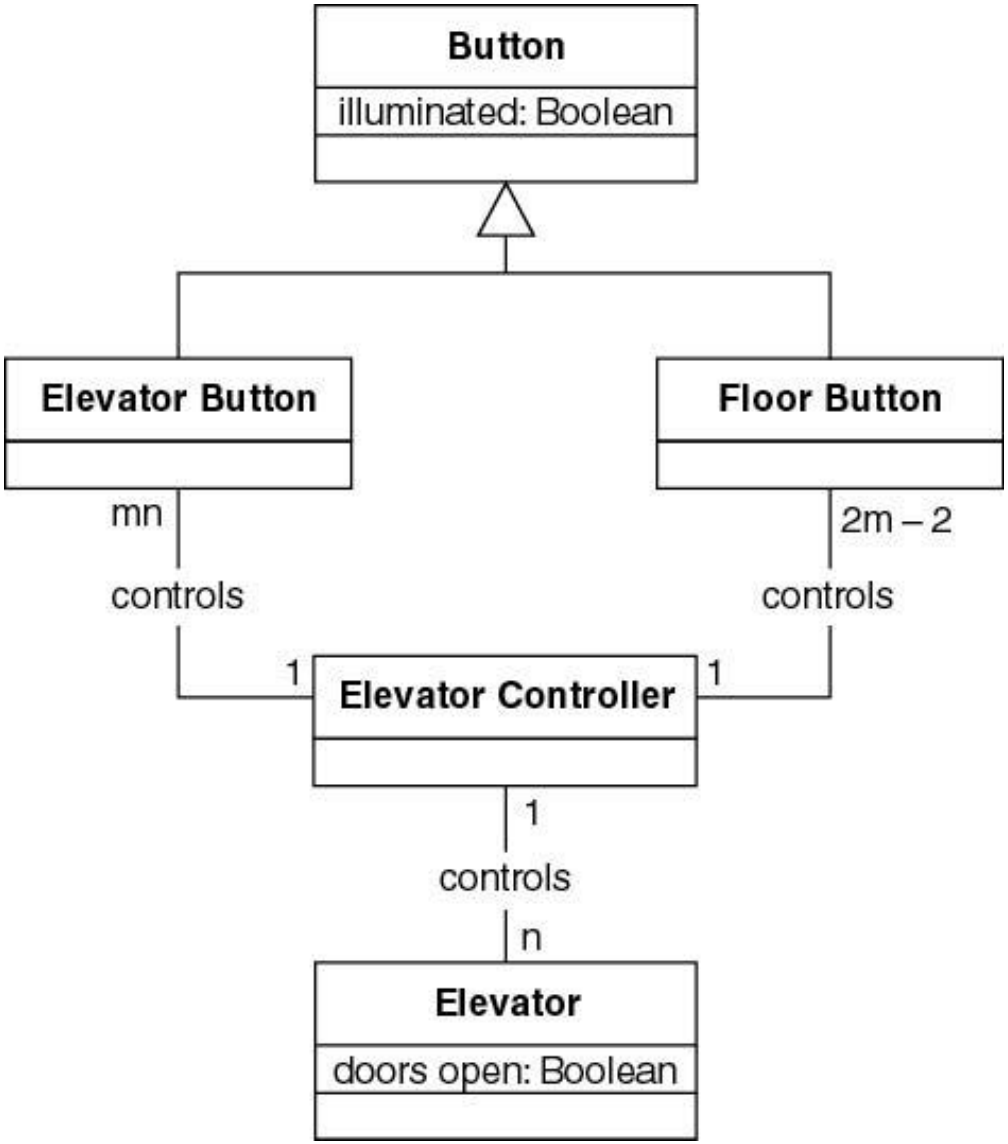
Problem

- wie kommunizieren die Buttons mit dem Lift?
- Wir benötigen einen **Elevator Controller**

Zweite Version

Alle Beziehungen sind 1-to-n

- dieses Design lässt sich besser implementieren



CRC Karten

Wurden 1989 in der OOA eingeführt

Auf jeder Karte steht eine Klasse

- Name der Class
- Funktionalität (Responsibility)
- Liste der Klassen, mit denen sie zusammenarbeitet (Collaboration)
- es gibt auch CASE Tool

Stärken

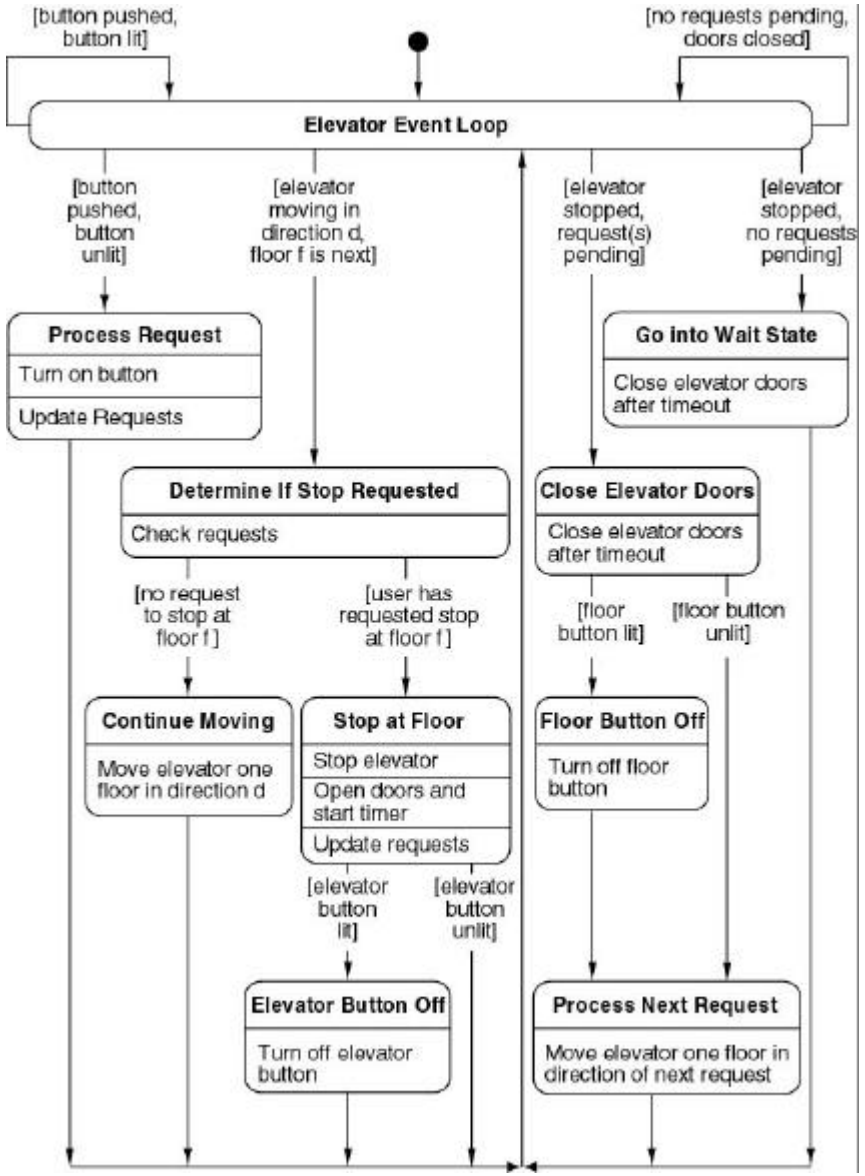
- einfach, übersichtlich, kommunikativ (Team)

Schwächen

- Expertise ist nötig

3. Dynamische Modellierung

UML Zustands- Diagramm



CRC Karten

Wir haben eine Klasse vergessen

- Lifttüren haben einen *Zustand*, der sich verändern kann
- neue Klasse **Elevator Doors**

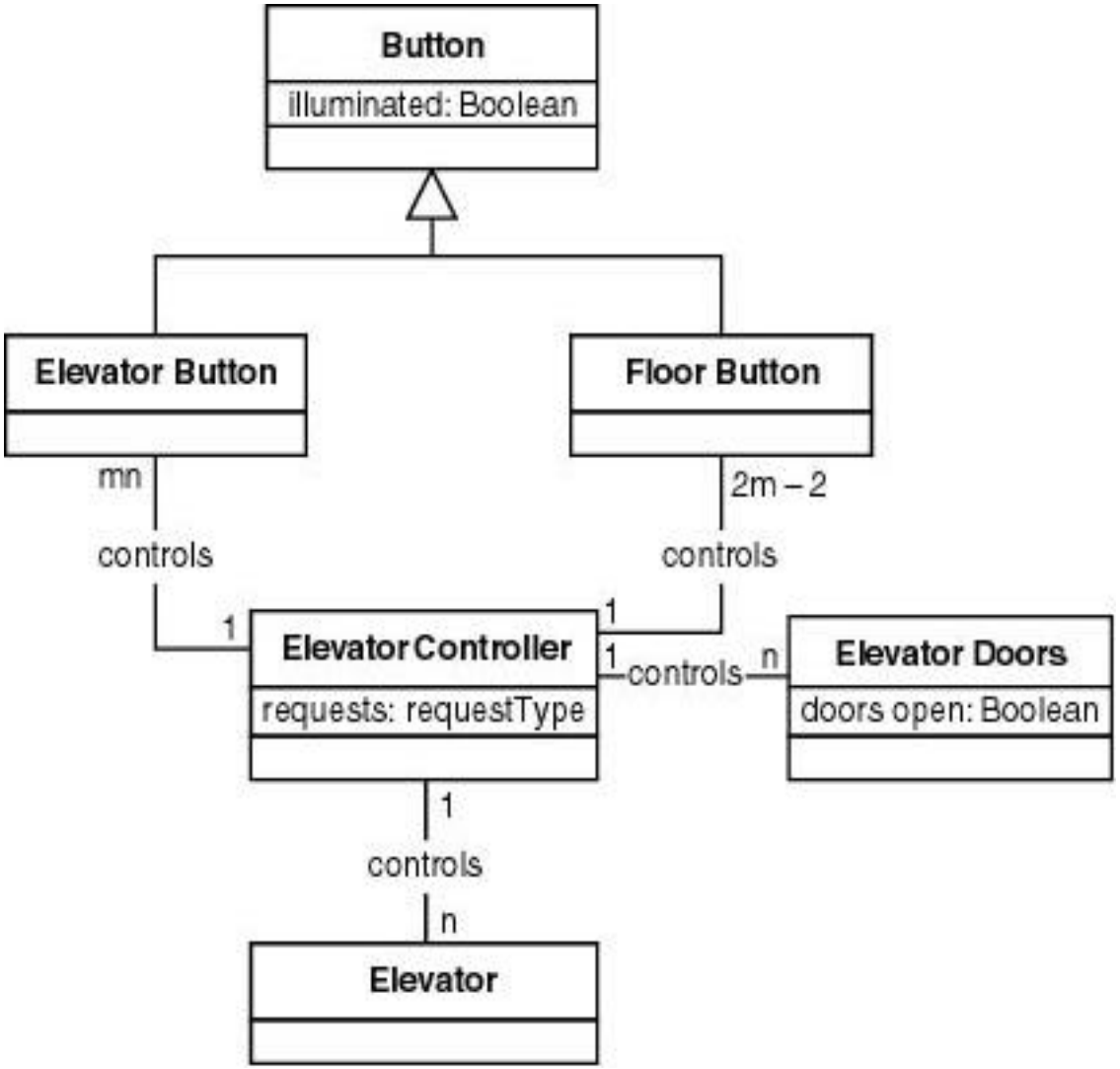
Wie sieht das Modell nun aus?

Anpassen der Modelle und Use Cases (= Anwendungsfall)

Zweite CRC Card Version

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**

Dritte Version des Klassendiagramms



Warum muss man iterieren?

Ist die Methode nicht sauber?

- Das Wasserfall Modell kennt Loops
- Das Rapid Prototyping Modell iteriert vor dem Entwicklungsprozess
- inkrementelle und das Spiral Modell iterieren auch

Iterationen sind offensichtlich eines der inhärenten Kennzeichen von erfolgreichen SW / IT Prozessmodellen

- speziell bei mittleren und grösseren Projekten
- auch in OO Projekten

CASE Tools für die OOA Phase

Diagramme

Diagramme ändern oft

- ein Zeichen-Werkzeug ist sehr hilfsreich
- viele Werkzeuge gehen weiter

UML wird von allen modernen Tools unterstützt

- Beispiele
  - Rose
  - JTogether
  - …

  - Visio
  - …