Elevator Example
Problem

GSU schedules to upgrade all the campus elevators in 6 months. Due to incompatibility with the new hardware, the current software that controls the elevators cannot be reused. You are asked to develop the software that controls the elevator so that the elevators will operate in the same way as they do now.
You formed a group with 5 group members.

What do you do now?
Management aspect
- schedule,
- budget,
- resources
- software life cycle
- development environment & tools

Development aspect
Software Development

Start with functional requirement
Why?
How?
  - observation
  - documentation
  - domain expert

How about nonfunctional requirement
  - performance
  - constraints
How to describe functional requirement?

“I take elevators everyday and I know how they work. There is no need to describe the requirements – I have everything in my mind”.

We need a “formal” or “semi-formal” way to document the requirements.
Elevators use case: Quickly using idle elevator
1. User enters elevator on ground floor
2. User presses floor 8 button
3. Doors close
4. Elevator moves up to desired floor
5. Doors open
6. User leaves
Note: Elevator was waiting on ground floor because it had been idle
Elevators use case: **Hold the door**
User1 enters elevator on ground floor
User1 presses floor 8 button
Doors start to close
User1 sees User2 approaching
User1 presses "open door" button
Doors open fully
User2 enters and sees floor 8 button already lit
Doors close after 10 seconds
Elevator moves up to desired floor
Doors open
User leaves
Note: User2 does not need to press anything
Use case example: Special conditions

1. A user enters into the elevator but does not press anything
2. Two users press floor buttons indicating opposite moving directions
3. One user (inside) presses a floor button but then changes his/her mind and then presses a button for an opposite direction.
   - Is there a “cancel” button available?
4. One user (outside) presses an up button but then changes his/her mind (not entering into the elevator) and press down button.
5. Emergency call function.
6. etc.
A use case diagram

Elevator system

Share a ride
Use idle elevator
Hold the door
Maintenance

user

user

administrator
Analysis

Analyze the requirement and identify major objects.
A high-level class diagram

- **Building**
  - name: String

- **Floor**
  - number: int

- **Elevator**
  - defaultFloor: int
  - currentFloor: int
  - desiredFloors: int

- **FloorButton**
  - number: int

- **Passenger**
  - passengerLimit: int

- **Freight**
  - weightLimit: int

Relationships:
- Building: 1 floor
- Floor: 1..* currentFloor
- Elevator: 1 desiredFloors
- FloorButton: 1..* passengerLimit
- Freight: 1 weightLimit

A use case diagram at a lower level

- Detect if (E,F) button is pressed
- Button illumination is turned on/off
- Move / stop Elevator
- Open / close Doors
A more detailed level

- Elevator controller
  - Elevator button
  - Floor button
  - Elevator motor
  - Door motor
  - Move up
  - Move down
  - Open door
  - close door
Another class diagram

Elevator — control — Elevator_Controller — control — Door

communicate with

1

Button

* 

Elevator_Button

Floor_Button
Sequence Diagram for Serving Elevator Button
Sequence Diagram for Serving Door Button
Detailed class diagram

Note: this diagram ignores all the sensors that detect person in door way, overloaded, etc.
## Detail Operation Description

<table>
<thead>
<tr>
<th>Module Name</th>
<th>Elevator_Control::Elevator_control_loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module Type</td>
<td>Method</td>
</tr>
<tr>
<td>Input Argument</td>
<td>None</td>
</tr>
<tr>
<td>Output Argument</td>
<td>None</td>
</tr>
<tr>
<td>Error Message</td>
<td>None</td>
</tr>
<tr>
<td>File Access</td>
<td>None</td>
</tr>
<tr>
<td>File Change</td>
<td>None</td>
</tr>
<tr>
<td>Method Invoke</td>
<td>button::illuminate,</td>
</tr>
<tr>
<td></td>
<td>button::cancel_illumination,</td>
</tr>
<tr>
<td></td>
<td>door::open,</td>
</tr>
<tr>
<td></td>
<td>door::close,</td>
</tr>
<tr>
<td></td>
<td>elevator::move,</td>
</tr>
<tr>
<td></td>
<td>elevator::stop</td>
</tr>
</tbody>
</table>
void elevator_control (void) {
    while a button has been pressed
        if button not on
            {button::illuminate;
             update request list;
            }
        else if elevator is moving up
            {
                if there is no request to stop at floor f
                    Elevator::move one floor up;
                else
                    (to be continue…)
            }
}
New Requirements

Your group spends 5 months on this project and starts to implement and test the software. Then one day you receive a phone call from GSU saying that all elevators should support the function of displaying text messages from the building manager......

What do you do now?