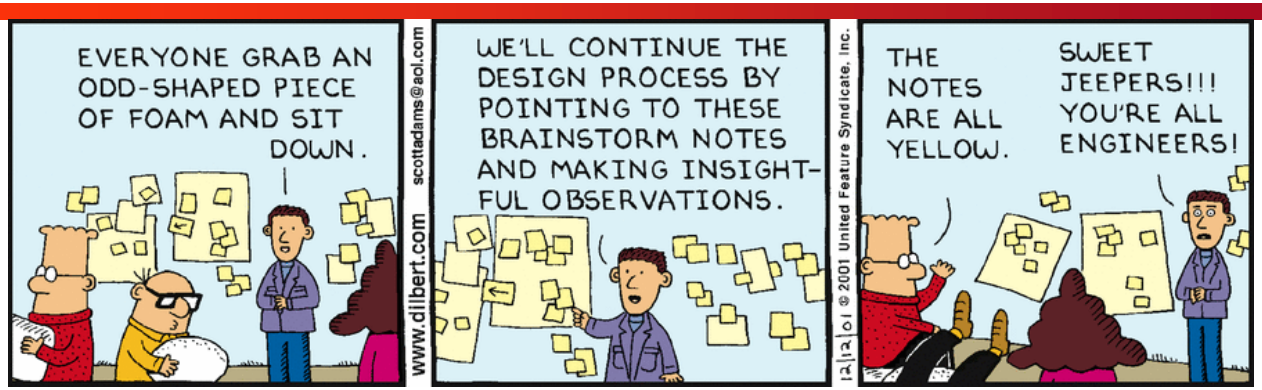
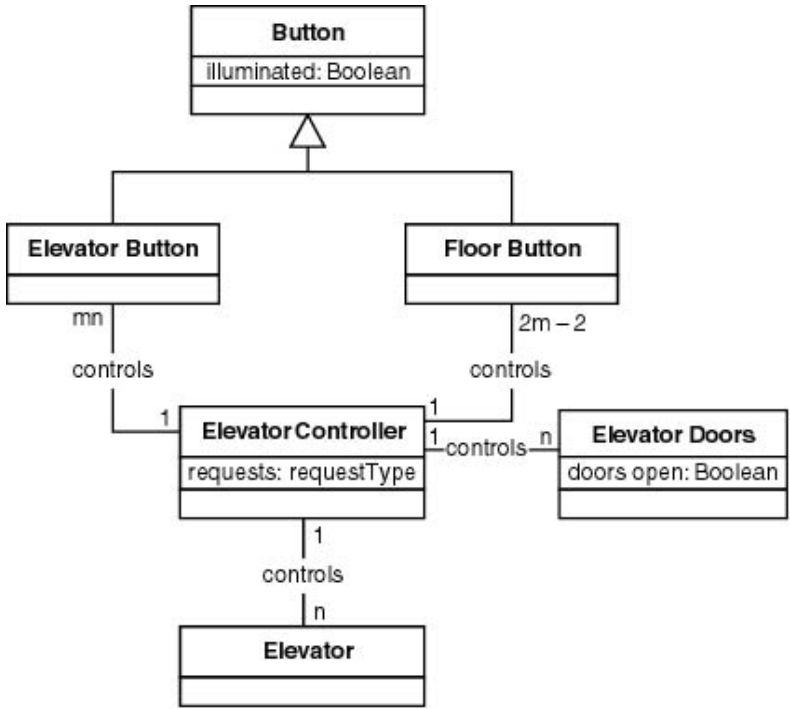
Object-Oriented Design (Schach Chap 14)



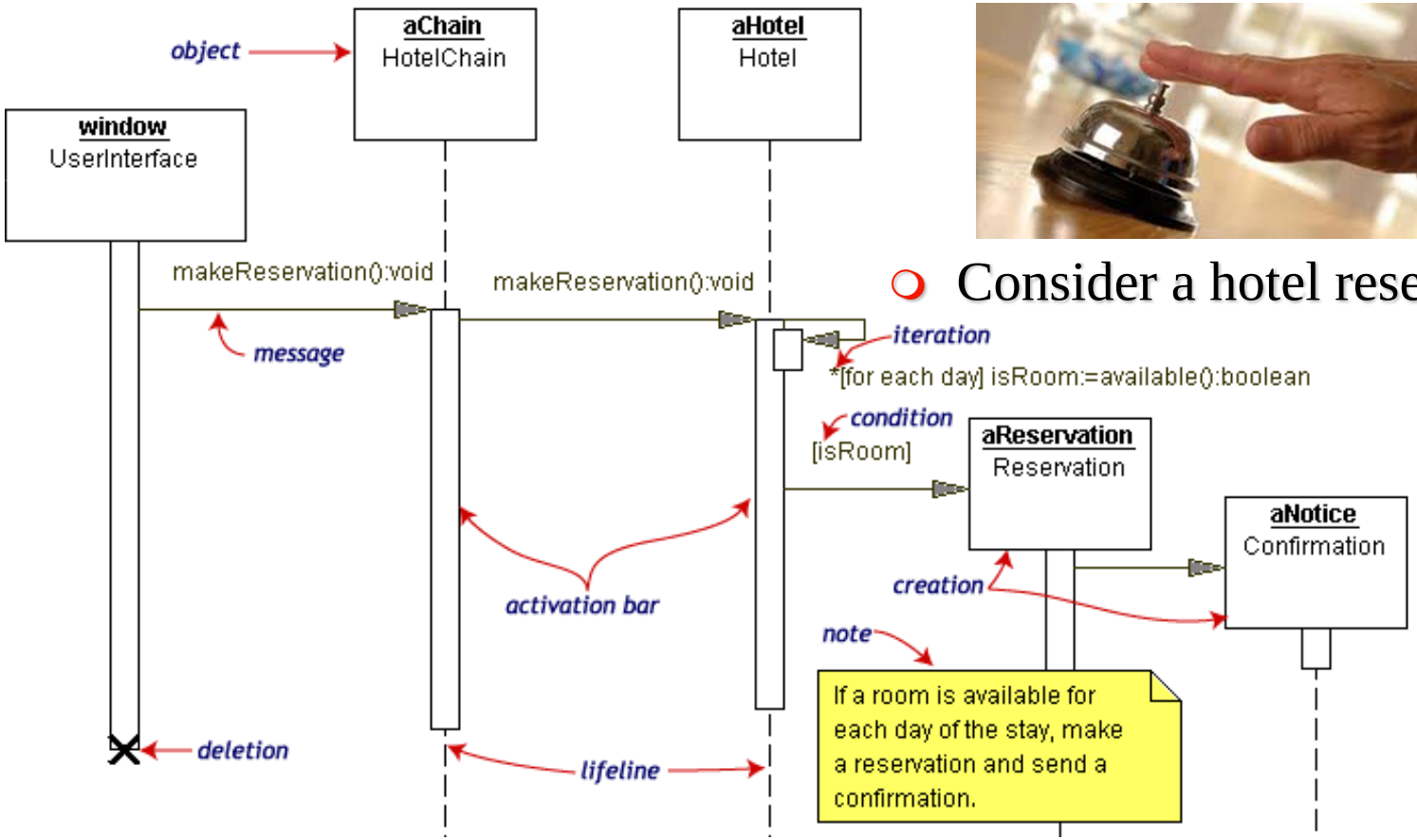
- **Goal:** Populate OOA's UML Class Diagram with the methods that belong to each class
- OOD uses UML Sequence diagrams for each scenario
- Develop UML ***detailed*** class diagram (id each class's methods)

Result of our Elevator System OOA: A UML ***Class Diagram***



Sequence Diagrams: *what* happens *when*

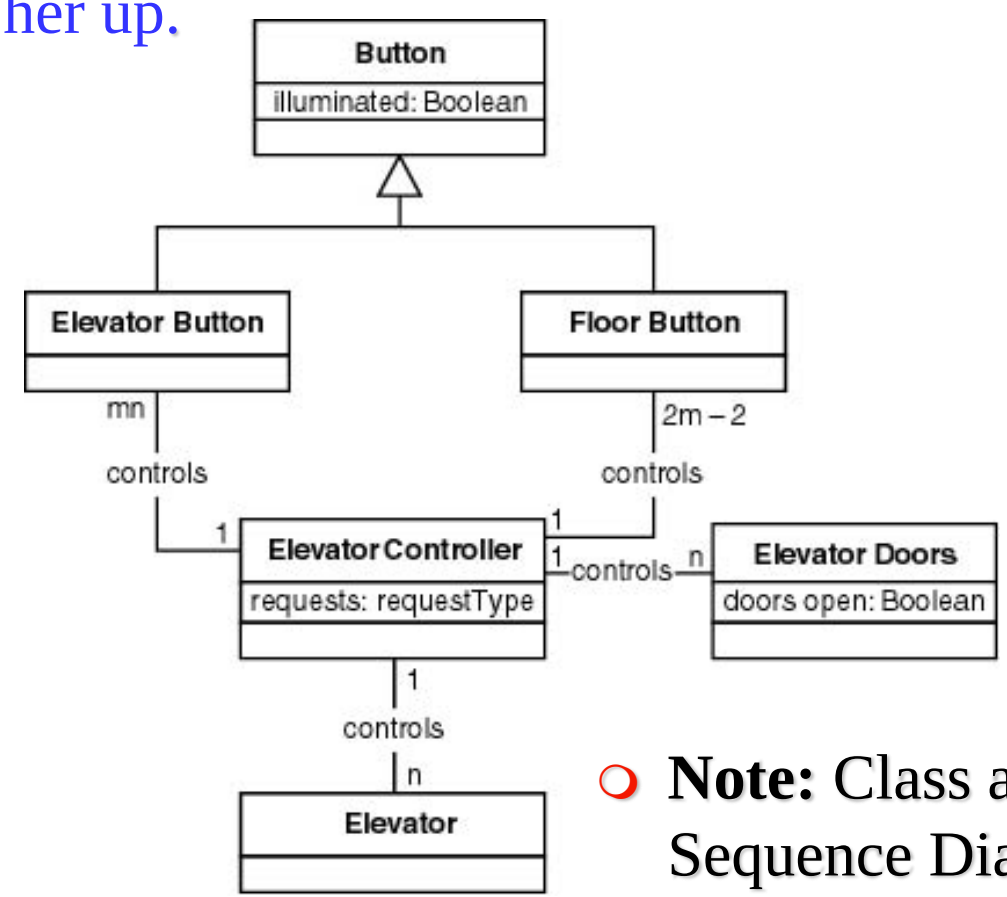
- Focus is on chronological interactions between objects. These become a class's methods
  - An object is represented by a rectangle and a vertical bar.
  - Message sending order indicated by position on axis.



- Consider a hotel reservation

Example: Elevator *Sequence* Diagram

- What would a UML Sequence Diagram look like for the following scenario => Sally is on the 10th floor and presses a floor button. The elevator arrives from the ground floor to pick her up.

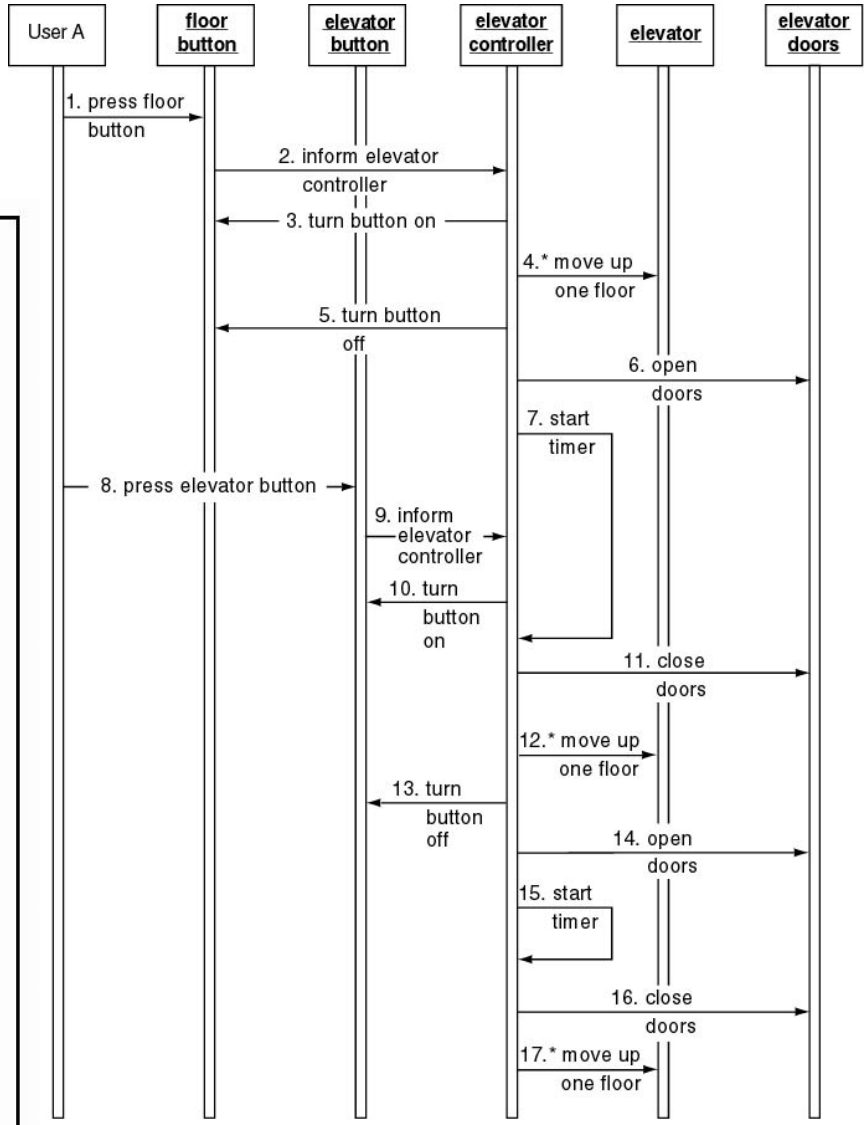


- **Note:** Class at head of arrow on Sequence Diagram gets tasked with implementing message as a method.

Connection between *Scenarios* & *Sequence* Diagrams

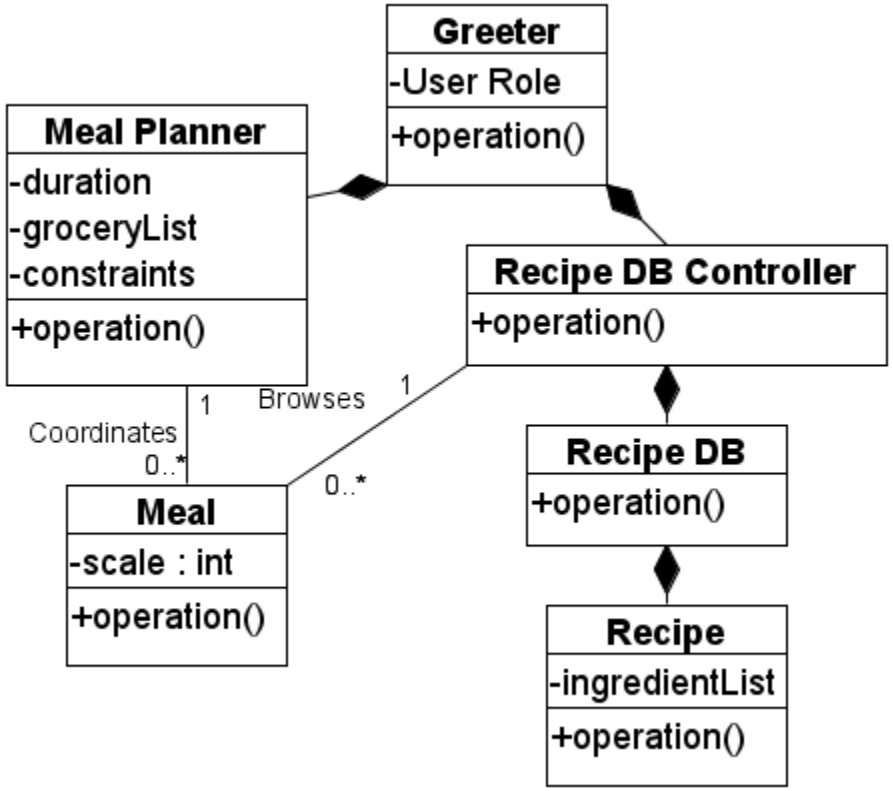
Example of a Scenario + resultant UML Sequence Diagram

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 floor button informs the elevator controller that the floor button has been pushed.
3. The elevator controller sends a message to the Up floor button to turn itself on.
4. The elevator controller sends a series of messages to the elevator to move itself up to floor 3. The elevator contains User B, who has entered the elevator at floor 1 and pressed the elevator button for floor 9.
5. The elevator controller sends a message to the Up floor button to turn itself off.
6. The elevator controller sends a message to the elevator doors to open themselves.
7. The elevator control starts the timer.
   User A enters the elevator.
8. User A presses elevator button for floor 7.
9. The elevator button informs the elevator controller that the elevator button has been pushed.
10. The elevator controller sends a message to the elevator button for floor 7 to turn itself on.
11. The elevator controller sends a message to the elevator doors to close themselves after a timeout.
12. The elevator controller sends a series of messages to the elevator to move itself up to floor 7.
13. The elevator controller sends a message to the elevator button for floor 7 to turn itself off.
14. The elevator controller sends a message to the elevator doors to open themselves to allow User A to exit from the elevator.
15. The elevator controller starts the timer.
    User A exits from the elevator.
16. The elevator controller sends a message to the elevator doors to close themselves after a timeout.
17. The elevator controller sends a series of messages to the elevator to move itself up to floor 9 with User B.

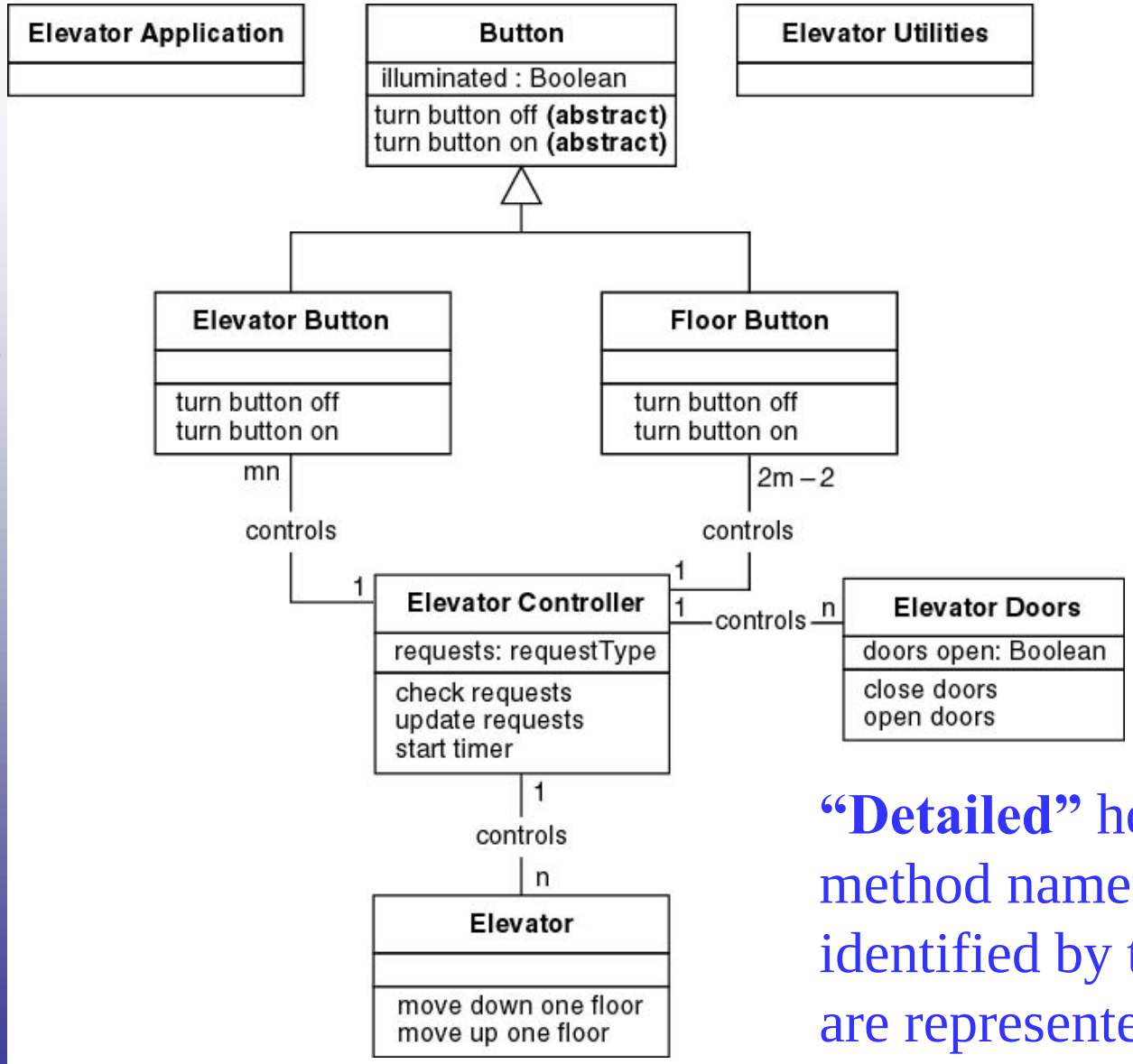


MFKA UML Sequence Diagram

- ***ICE:*** Produce a UML *Sequence Diagram* for this Much-o Fantastic-o Kitchen Assistant scenario: Find a recipe for a meal of chili that includes chipotle powder, scale it, and generate a grocery list of the ingredients needed for a week's worth of chili (to be eaten 3 times a day) by 200 Marines.



Construct UML *Detailed* Class Diagram



"Detailed" here means that all method names and attributes identified by the design team, are represented on the diagram.

Perform Detailed Design

- Detailed design of method elevator controller loop, draws heavily from prior UML diagrams.

Design Team:

- Should not do too much:
  - Detailed design should not become complete code
- Should not do too little:
  - It is essential for detailed design to fully indicate how all functionality is met.
- So, how to verify we are done with detailed design?

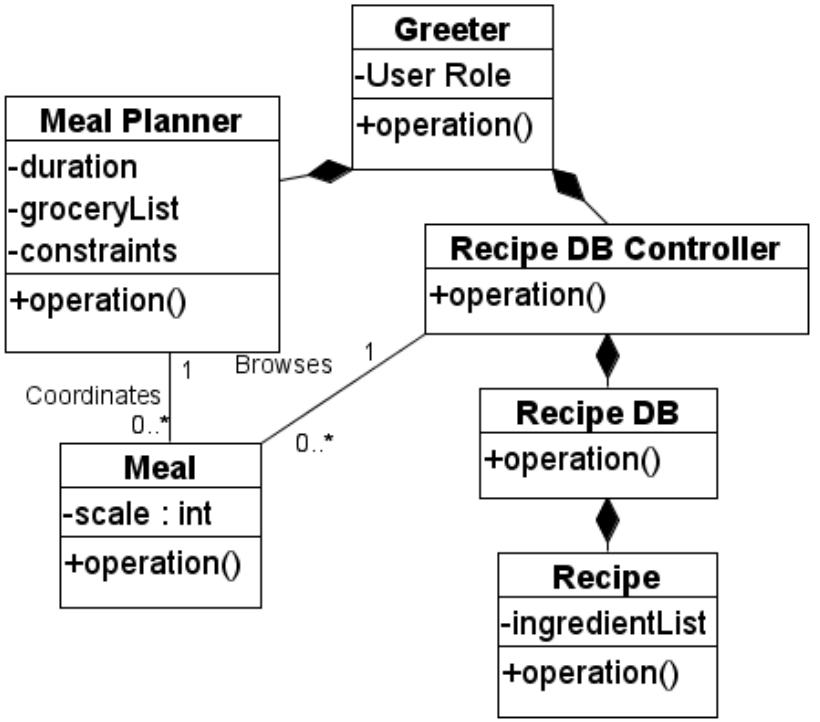
```
void elevator event loop (void)
{
  while (TRUE)
  {
    if (a button has been pressed)
      if (button is not on)
      {
        update requests;
        button::turn button on;
      }
    else if (elevator is moving up)
    {
      if (there is no request to stop at floor f)
        elevator::move one floor up;
      else
      {
        stop elevator by not sending a message to move;
        elevator doors::open doors;
        start timer;
        if (elevator button is on)
          elevator button::turn button off;
        update requests;
      }
    }
    else if (elevator is moving down)
      [similar to up case]
    else if (elevator is stopped and request is pending)
    {
```

ICE: Detailed Design

- ICE: Provide *pseudo-code* for a method in class Meal Planner called `planMeal()`
  1. Identify what arguments `planMeal()` will need and what methods, including those defined in other classes, that `planMeal()` will use
  2. Show how `planMeal()` fulfills the below scenario.

MFKA Meal Plan Scenario: Find a recipe for a meal of chili that includes chipotle powder, scale it, and generate a grocery list of the ingredients needed for a week's worth of chili (to be eaten 3 times a day) by 200 Marines.

```
void elevator event loop (void)
{
  while (TRUE)
  {
    if (a button has been pressed)
      if (button is not on)
      {
        update requests;
        button::turn button on;
      }
    else if (elevator is moving up)
    {
```



Testing During the Design Phase

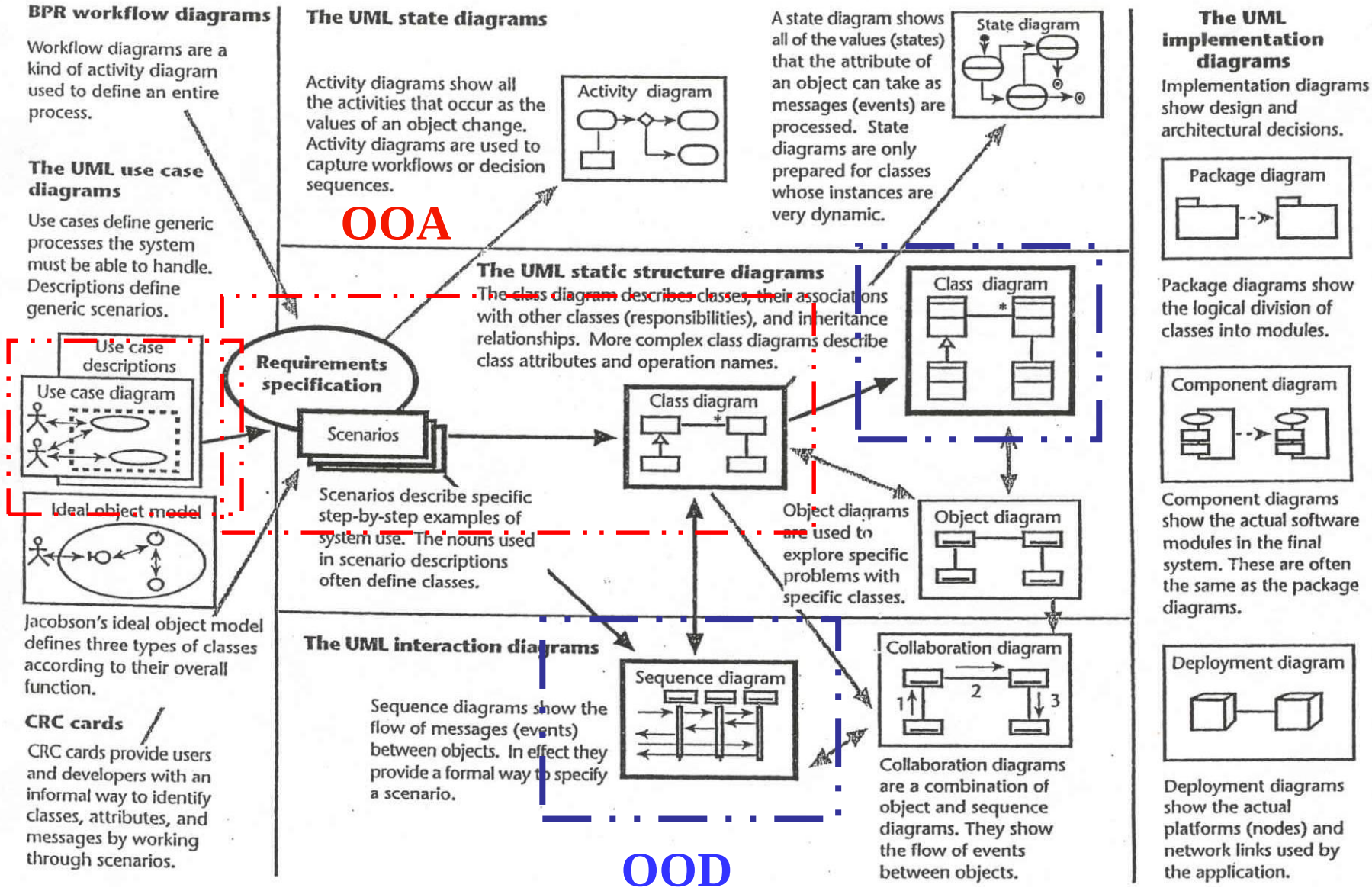
- Design itself must be correct
  - No logic faults
  - Fully defined Class interface.
- Design reviews. Purpose is to show that Design correctly reflects Specification. How can this be shown?

  - Show how each normal/abnormal scenario is met by the design.
  - Critical that a proper set of scenarios was developed during OOA.

Metrics for OOD

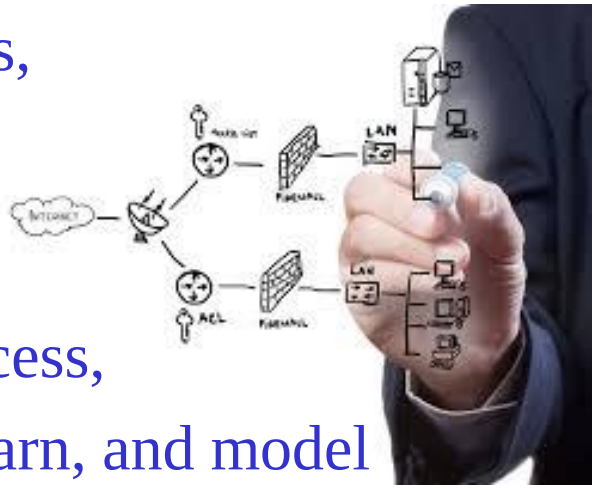
- What Metrics can be used to describe various aspects of the Design?
- One is McCabe's Cyclomatic complexity. Measure based on # of controlflow decisions, can apply to pseudo-code.
  - the number of binary decisions plus 1,
  - a metric of design quality, the lower the M the better
    - **Advantage:** Easy to compute, can be automated
    - **Disadvantage**: Measures *control* complexity only, does not measure *data* complexity.
- What other aspects of a Design can be measured?

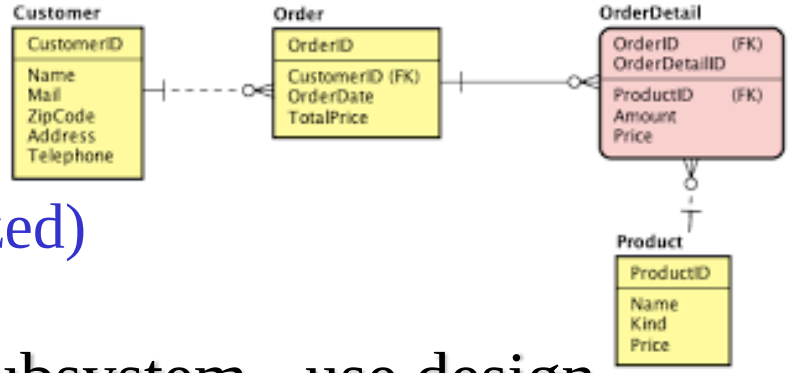
Recap: Relationships Between UML Diagrams



Design Artifacts for your Capstone Project

- You likely have seen other (non-UML) design artifacts depending on the courses you have taken (some examples):
  - **network diagrams** (routers, switches, hosts, IP ranges, device links, etc)
  - **machine learning design**:
    - data description, data creation process,
    - description of what you want to learn, and model you intend to use (Bayesian net, neural net, plan for using model output).
  - **entity relationship model**
    - database schema (normalized)





- Your Capstone Project: for <u>each</u> subsystem - use design artifacts that make sense in the context of your project