System Modeling, review

- The object model describes the structure of the system (objects, attributes, and operations)

- The dynamic model describes the behavior of the system: how the objects change state (how the attributes change) and in which order the state changes can take place
  - Interaction diagrams
  - Activity diagrams
  - State Diagrams
Different Types of Interaction Diagrams

- An Interaction Diagram typically captures a use-case
  - A sequence of user interactions

- **Sequence diagrams**
  - Highlight the sequencing of the interactions between objects

- Collaboration diagrams
  - Highlight the structure of the components (objects) involved in the interaction
Home Heating Use-Case

Use case:  Power Up
Actors:    Home Owner (initiator)
Type:      Primary and essential
Description:  The Home Owner turns the power on. Each room is temperature checked. If a room is below the desired temperature the valve for the room is opened, the water pump started, the fuel valve opened, and the burner ignited. If the temperature in all rooms is above the desired temperature, no actions are taken.

Cross Ref.:  Requirements XX, YY, and ZZ
Use-Cases:   None
Sequence Diagrams

- **A Homeowner**
- **OnOffSwitch**
- **the On-Off Switch**
- **the Controller: Controller**
- **a Room**
- **WaterPump**

Use the left column to offer comments about the messages.

**Synchronous message**

**Response to synchronous message**

**Guard for message**

Use * to denote iteration.

- **System On**
- **powerOn()**
- **tempStatus=checkTemp()**
- **[tempStatus == low]**
- **pumpOn()**
- **[tempStatus == low]**
- **openValve()**
- **[tempStatus == low]**
- **startBurner()**

* [for all rooms]
When the owner turns the system on, the on switch notifies the controller. The controller creates a room object for each room in the building. The rooms sample the temperature in the room every 5 s. When a low temp is detected, the room notifies the controller.
When to Use Interaction Diagrams

- When you want to clarify and explore single use-cases involving several objects
  - Quickly becomes unruly if you do not watch it

- If you are interested in one object over many use-cases -- state transition diagrams

- If you are interested in many objects over many use cases -- activity diagrams
State Diagrams
Events, Conditions, and States

- **Event**: something that happens at a point in time
  - Operator presses self-test button
  - The alarm goes off

- **Condition**: something that has a duration
  - The fuel level is high
  - The alarm is on

- **State**: an abstraction of the attributes and links of an object (or entire system)
  - The controller is in the state self-test after the self-test button has been pressed and the reset-button has not yet been pressed
  - The tank is in the state too-low when the fuel level has been below level-low for alarm-threshold seconds
To make a call, the caller lifts receiver. The caller gets a dial tone and the caller dials digit (x). The dial tone ends. The caller completes dialing the number. The callee phone begins ringing at the same time a ringing begins in caller phone. When the callee answers the called phone stops ringing and ringing ends in caller phone. The phones are now connected. The caller hangs up and the phones are disconnected. The callee hangs up.
A State Machine

Idle
  ──── on-hook
     ├─── off-hook
     │     ▼
     │     Dial tone
     │         ──── dial(x)
     │         ▼
     │         Connecting
     │             ──── valid-number
     │             ▼
     │             Connected
     │                 ──── called-phone-answers
     │                 ▼
     │                Disconnected
     │                     ──── called-phone-hangs-up
     │                     ▲
     │                     Ringing
     │                         ──── number-busy
     │                         ▲
     │                         Busy tone
     │                             ──── routed
     │                             ▲
     │                             Dialing
     │                                 ──── digit(x)
Hierarchical State Machines

- Group states with similar characteristics
- Enables information hiding
- Simplifies the diagrams

Diagram:

- **Idle**
  - off-hook
  - dial(x) [x is a digit]
  - dial(x) [x = *]

- **Make Call**
  - Establish call
  - Dialing
    - dial(x)
    - valid-number
  - Connecting
    - do/ find connection
    - routed
  - Busy tone
    - do/ busy tone
  - Ringing
    - do/ring bell
  - Connected
    - called-phone-answers / connect line
    - called-phone-hangs-up / disconnect line
  - Disconnected
    - on-hook
    - on-hook / disconnect line

Voice Mail
Information Hiding

Make Call

Establish call
- on-hook
- on-hook / disconnect line

Disconnected
- on-hook
- on-hook / disconnect line

Dial tone
- off-hook
- dial(x) [x is a digit]
- dial(x) [x = *]

Dialing
- dial(x)
- valid-number
- number-busy
- connected

Connecting
- find connection
- routed

Busy tone
- do/ busy tone

Ringing
- do/ ring bell

Establish call
- called-phone-answers /
  - connect line
- called-phone-hangs-up /
  - disconnect line

Voice Mail
- on-hook

Idle
- on-hook
When to use State Machines

- When you want to describe the behavior of one object for all (or at least many) scenarios that affect that object

- Not good at showing the interaction between objects
  - Use interaction diagrams or activity diagrams

- Do not use them for all classes
Activity Diagrams
Activity Diagrams

- Shows how activities are connected together
  - Shows the order of processing
  - Captures parallelism

- Mechanisms to express
  - Processing
  - Synchronization
  - Conditional selection of processing
Coffee Example

1. Find Beverage
   - If found coffee, go to Put Coffee in Filter
   - If no coffee, go to Decision

2. Decision
   - If no soda, go to [no coffee]
   - If found soda, go to Get can of soda

3. Get can of soda
   - Get Cups
   - Put Filter in Machine
   - Put Coffee in Filter
   - Add Water to Reservoir

4. Turn On Machine
   - coffePot.TurnOn
   - Brew Coffee

5. Drink Beverage
   - Pour Coffee
When to Use Activity Diagrams

- Useful when
  - Analyzing a use case (or collection of use cases)
  - Understanding workflow in an organization
  - Working with multi-threaded applications
    - For instance, process control applications
  - Do not use activity diagrams
    - To figure out how objects collaborate
    - See how objects behave over time