Dynamic Modeling - Finite State Machines

SWE 321
Fall 2014
Rob Pettit
Finite State Machines

- Finite number of states
  - Only in one state at a time
- Transition
  - Change of state
  - Caused by event
  - Transition to same or different state
  - Action may result from state transition
    - As entry into a state
    - As exit from a state
    - Activity while in a state
- Representation
  - State transition table
  - State machine diagram (UML)
State Machine Diagrams

- State machine diagram
  - Graphical representation of finite state machine
  - Relates events and states
  - States are rounded boxes
  - Transitions are arcs
    - May include actions
- Event
  - May cause change of state (by triggering a transition)
- State
  - A recognizable situation
  - Exists over an interval of time
  - Represents an interval between successive events
UML State Machine Notation

SuperState

entry / action
do / activity
exit / action

State 1

entry / action
do / activity
exit / action

Event [condition] / action

State 2

Event
Events

• Event
  – A discrete signal that happens at a point in time
  – Also known as a stimulus
  – Input to an object
• Events may cause a change in state
• Events may trigger actions
  – Actions can be internal or external
• Events may have associated conditions
• Signal events can be used to communicate between state machines
Events and Conditions

- State transition label
  - Event [Condition]
- Condition is a Boolean function
  - Conditions are optional on state machines
  - Condition is true for finite period of time
- When event occurs, condition must be true for state transition to occur.
- If condition is false, state transition does not occur
- Condition may be used to indicate that event has occurred
  - E.g., Closedown Was Requested
Actions

- State transition label
  - Event / action(s)
  - Event [condition] / action(s)

- Action
  - Executed as a result of state transition
  - Executes instantaneously at state transition
  - Terminates itself
  - Is optional

- Entry/exit actions
Entry and Exit Actions

- Entry action
  - Action executed on entry into state
    - Entry / action
- Exit action
  - Action executed on exit from state
    - Exit / action
- Entry and Exit actions specified in the state
  - Must be true for every entry / exit occurrence
  - If not, then you must use actions on the individual transition arcs
Activities

- Activity
  - Not instantaneous
  - Executes for duration of state
    - do / Activity in state
- Examples of activities
  - Increase Speed
    - Executes for duration of Accelerating state
  - Maintain Speed
    - Executes for duration of Cruising state
  - Resume Cruising
    - Executes for duration of Resuming state
Case Study: Automated Coffee Maker

- Person fills with coffee grounds & water filled
- Person puts carafe in place
- Person presses the Start button
- System starts brewing coffee
  - Heating element and carafe warmer turned on
  - When brewing temp reached, pump is started
  - Stop pump, heating element, and sound beeper when water level is empty
- Automated timer begins when brewing is complete
- When timer elapses, turn off carafe warmer.
Hierarchical State Machines

- Early versions of state machines
  - Flat structure (no composite states)
  - With many / complex states, diagrams became very cluttered

- UML State Machines
  - Based on Harel Statecharts
  - Allow hierarchical composition to manage complexity
    - Superstate decomposed into substates
    - Default entry states
    - Transition out of superstate corresponds to transition out of every substate
Hierarchical State Machines

- **OR decomposition**
  - When object is in superstate
    - It is in one and only one of substates
  - Transition into superstate
    - Must be to one and only one of substates
- **Aggregation of state transitions**
  - If same event causes transition out of every substate
  - Then aggregate into transition out of superstate
Hierarchical State Machines

- Orthogonal state machine
  - Used to depict states of different aspects of object
- AND decomposition
  - Object is in one substate on each lower level state machine
  - Object’s state is union of all substates
- Same event
  - May cause transitions on more than one state machine
- Output action on one state machine
  - May be input event on other state machine
- Substate on one state machine
  - May be condition on other state machine
State Machine Guidelines

- Create a state machine for each non-«entity» class
  - Some will only be simplistic, single-state machines
    - During analysis, we’ll focus on «control» classes
      - This is a deviation from the COMET guidelines, where state machines were only created for «state dependent» classes
- State name should be active not passive
  - Represents time period when something is happenING, e.g., Elevator Moving
    - Identifiable situation, e.g., Elevator Idle, Initial
- State names must be unique
- Must be able to exit from every state
- Flat state machine
  - State machine is only in one state at a time
- Hierarchical state machine
  - State machine is only in one substate at a time
    - Orthogonal state machines – one substate of each orthogonal region
State Machine Guidelines - 2

- Event is the cause of the state transition
  - Event happens at a moment in time
  - Event name indicates something has just happened
    - e.g., Received Up Request, Door Closed
- Action is the result of the state transition
  - Action is a command, e.g., Stop, Close Door
  - Action executes instantaneously
  - Activity executes throughout a given state
- More than one action possible with a state transition
  - Executed in the order they appear on the transition
- Condition is a Boolean value
  - Event [Condition]
  - State transition only occurs if
    - Event happens & Condition is True
    - Condition is True over some interval of time
- Events, Actions, Activities and Conditions are optional
Developing State Machines

• From the Use Case Model:
  – Look for distinct *modes* or *states*
    • Are there certain situations where inputs are processed different?
    • Are there certain situations where inputs are allowed and other where they are not?
    • Do inputs cause a change in the state?

• From the Interaction Diagrams
  – Look at incoming messages (events) and their corresponding results (actions)
  – Do events get processed differently depending on the state of the object?
Example

Consistency Between Statecharts and Sequence Diagrams
- Temperature Control System Case Study -