Steps in Using COMET/UML

1 Develop Software Requirements Model

2 Develop Software Analysis Model
   - Develop static model of problem domain (Chapter 7)
   - Structure system into objects (Chapter 8)
   - Develop statecharts for state dependent objects (Chapter 10)
   - Develop object interaction diagrams for each use case (Chapter 9, 11)

3 Develop Software Design Model
Finite State Machines and Statecharts

5. Section FSM

Hassan Gomaa

H. Gomaa, Chapter 10 - Designing Concurrent, Distributed, and Real-Time Applications with UML, Addison Wesley Object Technology Series, July, 2000

Copyright © 2011 Hassan Gomaa
All rights reserved. No part of this document may be reproduced in any form or by any means, without the prior written permission of the author.

Finite State Machines and Statecharts

- Many information and real-time systems are state dependent
  - Action depends not only on input event
  - Also depends on state of system
- Finite State Machine
  - Finite number of states
  - Only in one state at a time
- Statechart
  - Graphical representation of finite state machine
  - States are rounded boxes
  - Transitions are arcs
- Statechart relates events and states
States and Events

- **State**
  - A recognizable situation
  - Exists over an interval of time
  - Represents an interval between successive events

- **Event**
  - A discrete signal that happens at a point in time
  - Also known as a stimulus
  - Has no duration
  - Causes change of state
    - Referred to as state transition
Events and Conditions

- State transition label
  - Event [Condition]
- Condition is a Boolean function
  - Conditions are optional on statecharts
  - 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
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
Figure 10.8 Example of actions

Entry and Exit Actions

- Entry action
  - Action executed on entry into state
    - Entry / action
    - E.g., Start Cooking
- Exit action
  - Action executed on exit from state
    - Exit / action
    - E.g., Stop Cooking
Figure 10.10 Example of entry action

Fig. 10.11a: Actions on state transitions

Fig. 10.11b: Entry action

Figure 10.11 Example of exit action

Fig. 10.11a Actions on state transitions

Fig. 10.11b: Exit action
Hierarchical Statecharts

- Disadvantages of State Transition Diagrams and Flat Statecharts
  - Complex State Transition Diagrams get very cluttered
  - Limited capability for managing complexity
- Hierarchical Statecharts
  - Based on Harel Statecharts
  - Notation for hierarchical decomposition of state transition diagrams
    - Composite state decomposed into substates
    - Default entry states
    - Transition out of composite state corresponds to transition out of every substate

Figure 10.9 Example of same event and action on different state transitions
Hierarchical Statecharts

- **Sequential** decomposition
  - When object is in composite state
    - It is in one and only one of substates
  - Transition into composite state
    - 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 composite state
Figure 10.12 Example of hierarchical statechart showing composite state without substates

Guidelines on Statecharts

- State name must be passive not active
  - Represents time period when something is happening, e.g., Waiting for PIN
  - Identifiable situation, e.g., Idle, Initial
- State names must be unique
- Must be able to exit from every state
- Flat statechart
  - Statechart is only in one state at a time
- Hierarchical statechart
  - **sequential** decomposition
    - Statechart is only in one substate at a time
Guidelines on Statecharts

- Event is the cause of the state transition
  - Event happens at a moment in time
  - Event name indicates something has just happened
    - e.g., Card Inserted, Door Closed
- Action is the result of the state transition
  - Action is a command, e.g., Dispense Cash, Start Cooking
  - Action executes instantaneously
  - Activity executes throughout a given state
- More than one action possible with a state transition
  - No sequential dependency between actions
- 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
- Actions, Activities and Conditions are optional

Developing Statechart from Use Case

- Develop state dependent use case
- Start with scenario (one path through use case)
  - Consider sequence of interactions between actor and system
- Consider sequence of external events
  - Input event from external environment
  - Causes state transition to new state
  - Action may result from state transition
- Initially develop flat statechart
Figure 19.1 Banking System use case model

Figure 10.14 Statechart for ATM Control - Validate PIN use case
Developing Statechart from Use Case (continued)

- Consider alternative external events
  - Could result in additional states
  - Could result in additional state transitions
- Develop hierarchical statechart
  - States that can be aggregated to form composite state
  - Event causing transition from several states
    - Create composite state with one transition out of composite state
    - Instead of many transitions out of substates
Example of integrated statechart

Example of hierarchical statechart

Copyright 2011 H. Gomaa
Figure 10.18 Top level statechart for ATM Control

Figure 10.19 Statechart for ATM Control - Processing Customer Input composite state
Figure 10.20 Statechart for ATM - Processing Transaction composite state

Figure 10.21 Statechart for ATM Control - Terminating Transaction composite state
**Steps in Using COMET/UML**

1. Develop Software Requirements Model
2. **Develop Software Analysis Model**
   - Develop static model of problem domain (Chapter 7)
   - Structure system into objects (Chapter 8)
   - **Develop statecharts for state dependent objects (Chapter 10)**
   - Develop object interaction diagrams for each use case (Chapter 9, 11)
3. Develop Software Design Model