ICS 121 Lecture on UML State Machines

Jason E. Robbins jrobbins@ics.uci.edu

Information and Computer Science Dept.
University of California, Irvine

Overview

- What are State Machines?
  - States
  - Transitions
  - State machine diagrams
  - Advanced state machine features
- Example 1:Telephone Dialing
- Example 2: Traffic Intersection Signals
- Using Argo/UML for State Machines

What are States?

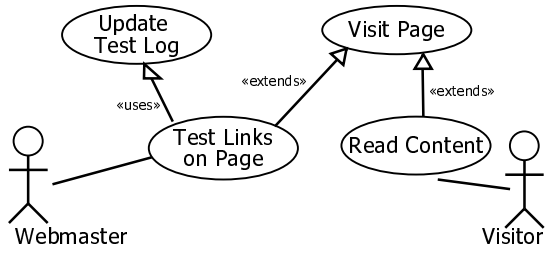
- A UML State is a mode of operation for an instance of a class
  - Some classes have state machines, others do not
  - Each instance of a class with a state machine is "in" a given state
- Example:
  - Class WristWatch has states: Date, Time, StopWatch, SetDate, SetTime
  - Each instance of WristWatch is in one of those states at any given moment
- States influence the way the object acts
  - E.g., the state of a WristWatch determines what is displayed

What are Transitions?

- Transitions are possible state changes
  - Start state, end state, trigger, guard condition, action
- Guard conditions must evaluate to true for the transition to be taken
- Transitions can have actions that execute when the transition is taken
- Example:
  - A WristWatch can go from state Date to state Time when the user presses the ModeButton
  - The WristWatch will beep on the transition into SetDate

State Machine Diagrams

- Shows all states and transitions for a class
- Start state is where new instances start
- Final state is where an instance is deallocated



Advanced Feature: Entry and Exit Actions

- A state can have an entry action that is executed whenever the instance enters that state.
  - This can be better than putting actions on all incoming transitions
- A state can have an exit action that is executed whenever the instance exits that state.

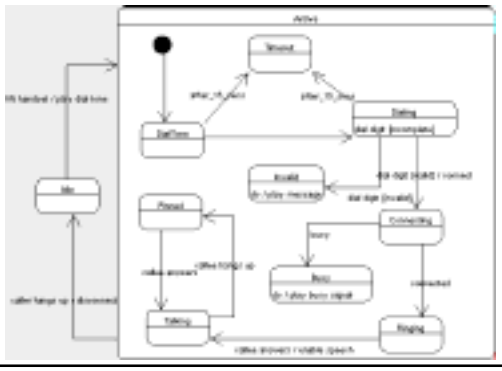
Advanced Feature: Composite States

- Composite states model commonalities between states
  - E.g., SetDate and SetTime might be in a Setting composite state
  - Composite states can have their own initial and final states

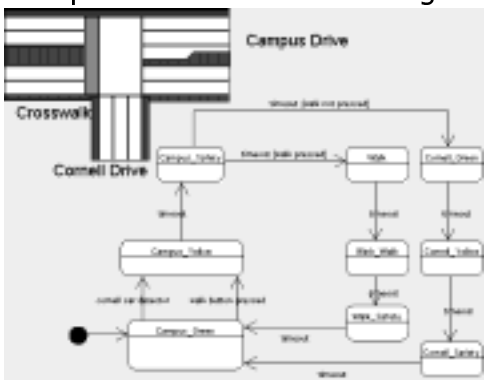
Advances Feature: Concurrent States

- Concurrent states allow an instance to do two things at once
  - e.g., a WristWatch is in one display state, and one battery state

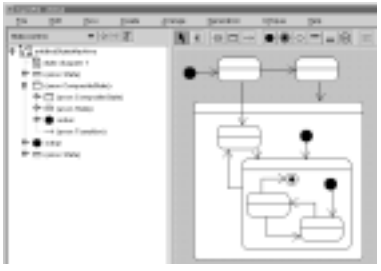
Example: Telephone Dialing



Example: Traffic Intersection Signals



- When you launch Argo/UML it automatically makes a class diagram and a use case diagram
- Place a class in the class diagram and name it
- Use the "Create" menu to make a State Machine



State
Composite State
Transition

Initial State
Final State
Decision
Branch
Join
History

Add Internal Transition

Tree view of nested states

Using Argo/UML

- Click and type to name states or enter transition labels
- Use toolbar buttons to make states, composite states, and transitions
- Transition labels: event [guard] / action
- Use property sheets if you are unsure on syntax