Modeling with UML
Chapter 2, part 3

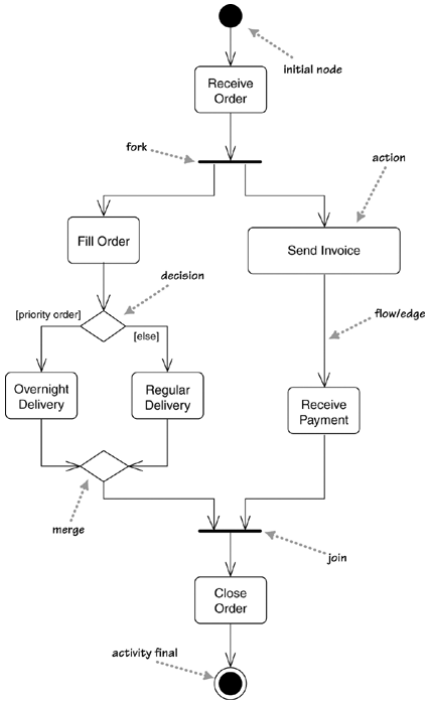
CS 4354
Summer II 2014

Jill Seaman

Activity Diagrams

- Describe the behavior of a system in terms of activities
- Represent the sequencing and coordination of actions or steps.

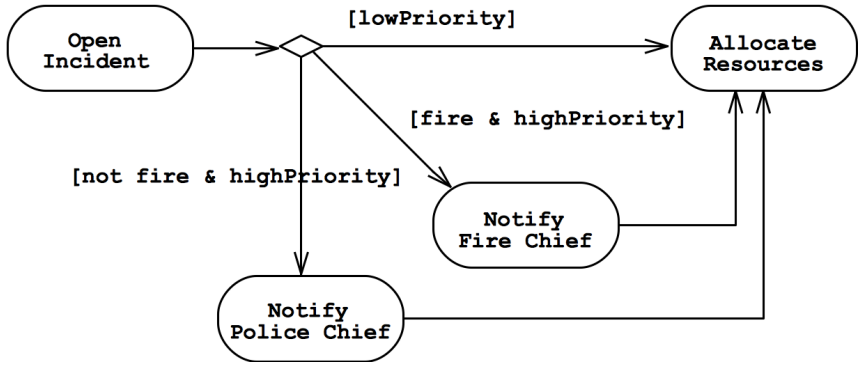
- Rounded rectangles represent actions and activities.
- Edges between activities represent control flow.
  - ✦branching, looping, concurrency
- Activity diagrams can be hierarchical:
  - ✦A given activity in a rounded rectangle could be further detailed in its own separate activity diagram.



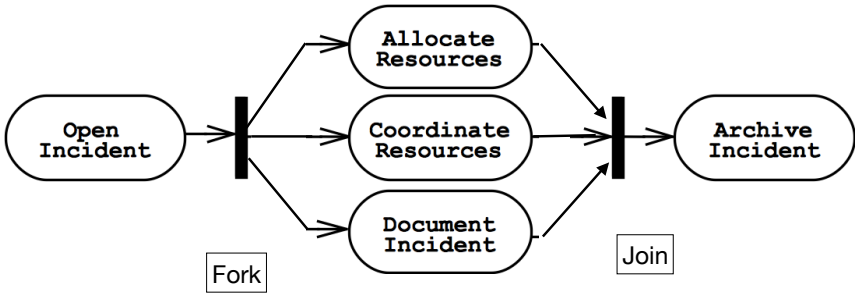
Activity Diagrams: control nodes

- Decisions (branches, alternates)
  - ✦Diamond with one incoming arrow two or more outgoing arrows.
  - ✦Outgoing edges labeled with guards (conditions) that select that arrow.
  - ✦Merge nodes (diamond with many incoming, one outgoing arrow) to mark the end of the branching, are often omitted.
- Fork nodes and Join nodes (concurrency)
  - ✦Fork: denotes splitting control into multiple threads
  - ✦Join: denotes synchronizing threads back into one
  - ✦Denotes activities that may be done in any order (they are not **required** to be done concurrently).

Decision in the Handle Incident process.



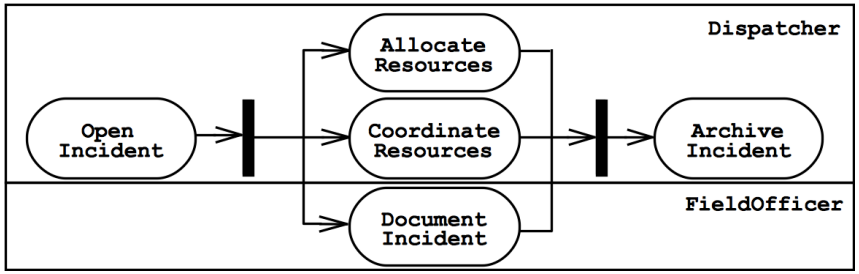
Concurrency in incident management process.



Activity Diagrams: swimlanes

- Swimlanes (activity partitions)
  - ✦Rectangles enclosing a group of activities
  - ✦Denotes object of subsystem that implements the activities
  - ✦Edges may cross swimlane boundaries

Swimlanes in incident management process.



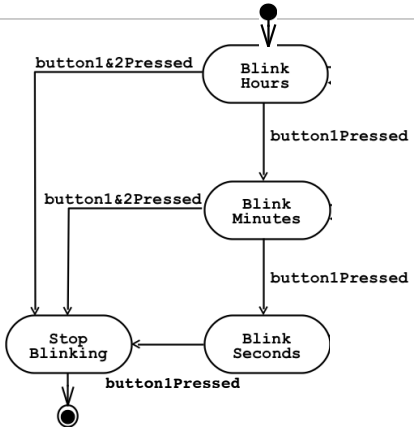
When and how to use Activity Diagrams

- When developing use cases
  - ✦activity diagrams are good at capturing business (and other) processes (also called workflows).
- During Object-Oriented design
  - ✦deciding what objects perform which activities (once you already have an activity diagram).
- When designing complicated operations/methods.
  - ✦use to model the control flow through a single method (like a flowchart or control flow diagram).

State diagrams

- Describe the dynamic behavior of an individual object
- Describes the sequence of states an object goes through in response to external events
  - ✦A graph: states are nodes, transitions are edges
- Transitions from one state to another occur as a result of external events

State diagram for the watch display

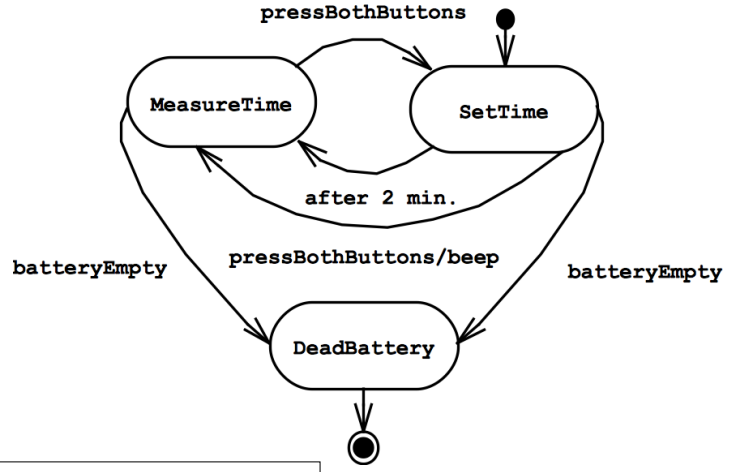


- small black circle: start state
- small black circle inside another circle: finish state

States and Transitions

- A state is a value of an attribute of an object that is changed by an external event.
  - ✦An Incident can exist in four states: Active, Inactive, Closed and Archived
  - ✦These are nodes in the graph
  - ✦A node can have some activity that is performed when the node is entered.
- A transition represents a change of state triggered by events, conditions, or time.
  - ✦Transitions are directed edges in the graph
  - ✦labelled by the event causing the transition:
    Event [Guard] / Action
    Each part is optional, Guard must be true to transition, Action is performed when transition occurs.

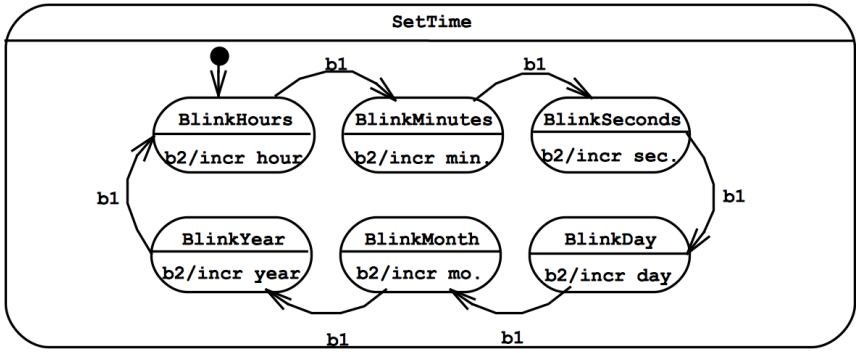
State Machine diagram for 2Bwatch



/beep is the action that happens when both buttons are pressed

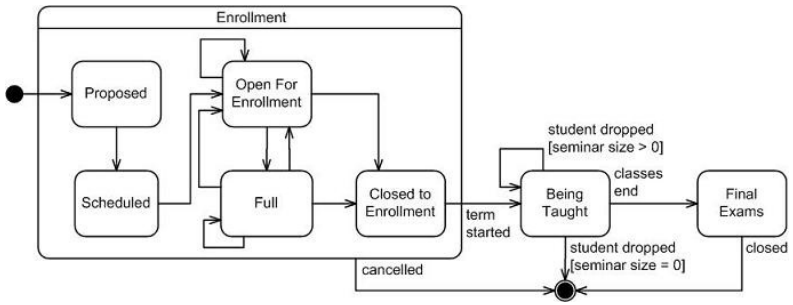
Nested State Machine example: SetTime state

a separate state diagram to describe setTime of previous slide



b1 = pressButton1 b2 = pressButton2

State diagram with nested state and guards



When and how to use State Diagrams

- When designing a class that has an attribute that responds to external events (and determining which state the object is in is not trivial)
  - ✦Use the state diagram to document the transitioning behavior
- During testing
  - ✦If you have a state diagram, you can develop tests that perform a sequence of events and then verify that the object is in the correct state with respect to the diagram
- If your object (or system) does not have an attribute that responds to external events, do not use state diagrams!
- User Interface objects often have behavior that is useful to depict with a state diagram