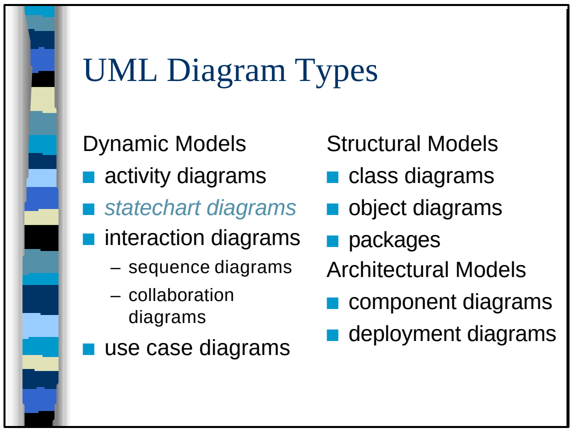


UML Diagram Types

Dynamic Models

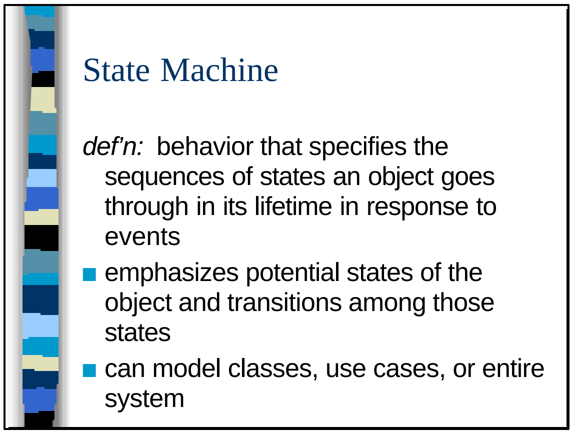
- activity diagrams
- *statechart diagrams*
- interaction diagrams
  - sequence diagrams
  - collaboration diagrams
- use case diagrams

Structural Models

- class diagrams
- object diagrams
- packages

Architectural Models

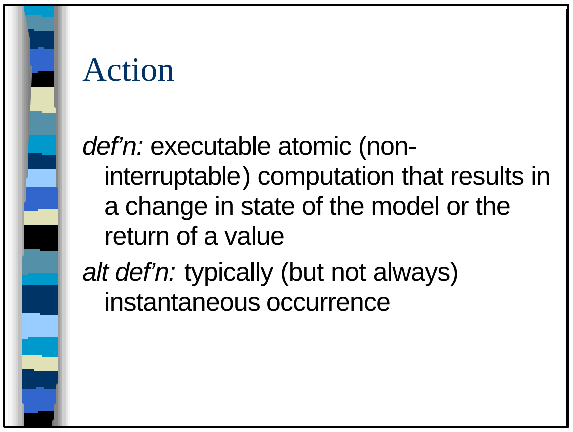
- component diagrams
- deployment diagrams



State Machine

def'n: behavior that specifies the sequences of states an object goes through in its lifetime in response to events

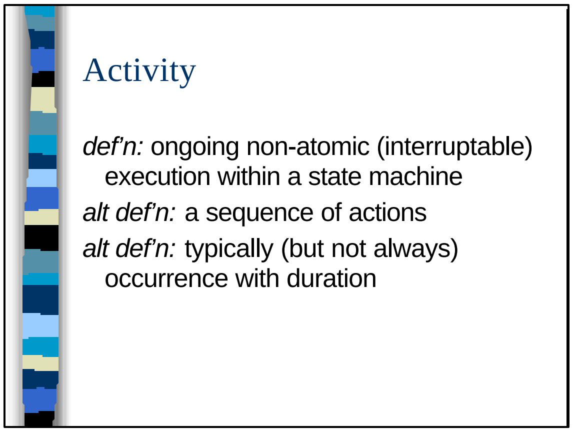
- emphasizes potential states of the object and transitions among those states
- can model classes, use cases, or entire system



Action

def'n: executable atomic (non-interruptable) computation that results in a change in state of the model or the return of a value

alt def'n: typically (but not always) instantaneous occurrence

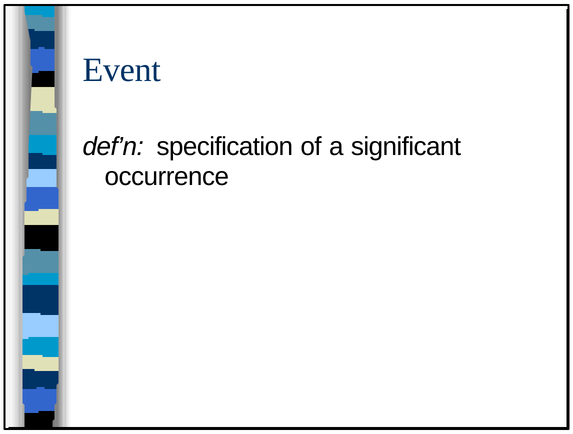


Activity

def'n: ongoing non-atomic (interruptable) execution within a state machine

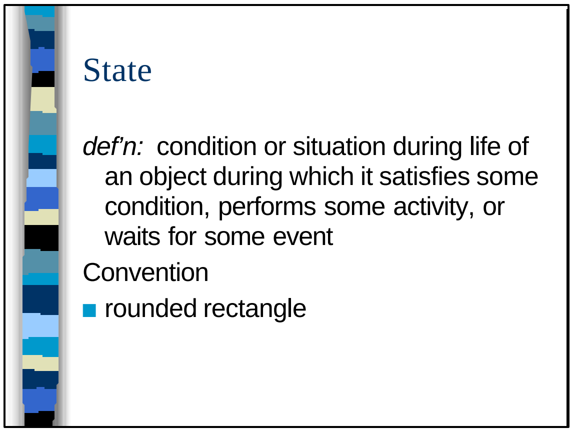
alt def'n: a sequence of actions

alt def'n: typically (but not always) occurrence with duration



Event

def'n: specification of a significant occurrence

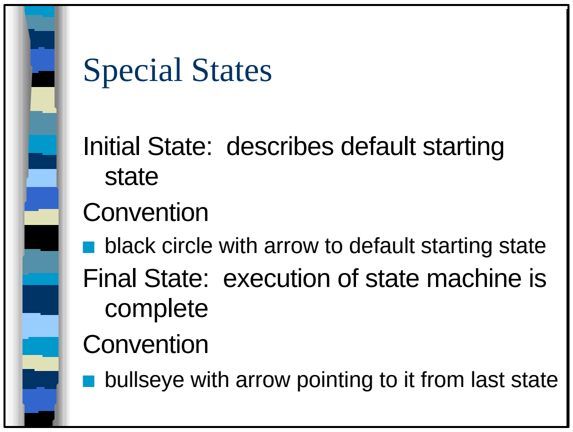


State

def'n: condition or situation during life of an object during which it satisfies some condition, performs some activity, or waits for some event

Convention

- rounded rectangle



Special States

Initial State: describes default starting state

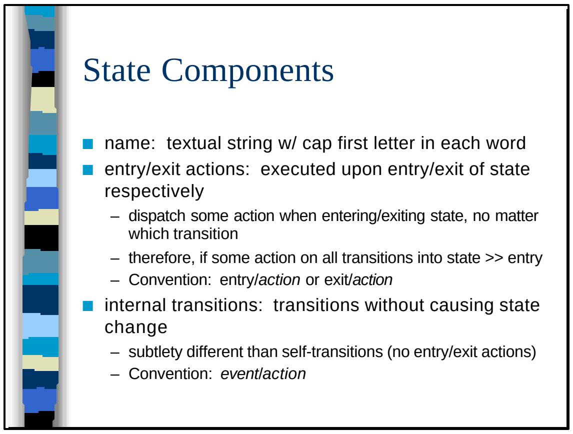
Convention

- black circle with arrow to default starting state

Final State: execution of state machine is complete

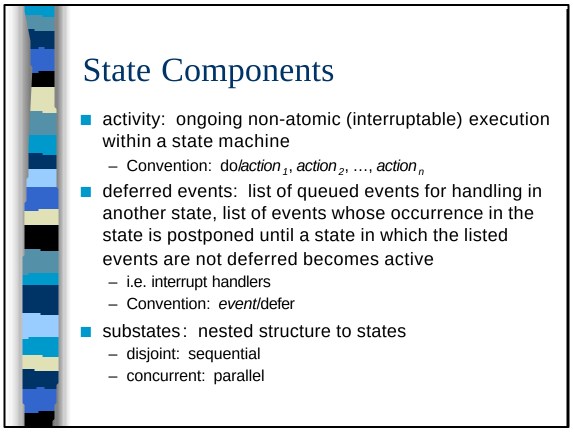
Convention

- bullseye with arrow pointing to it from last state



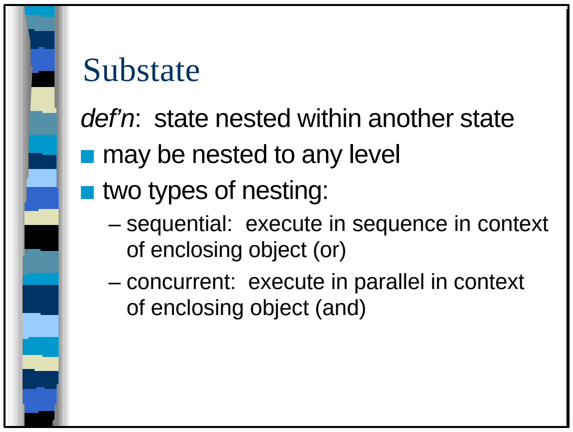
State Components

- name: textual string w/ cap first letter in each word
- entry/exit actions: executed upon entry/exit of state respectively
  - dispatch some action when entering/exiting state, no matter which transition
  - therefore, if some action on all transitions into state >> entry
  - Convention: entry/*action* or exit/*action*
- internal transitions: transitions without causing state change
  - subtlety different than self-transitions (no entry/exit actions)
  - Convention: *event*/*action*



State Components

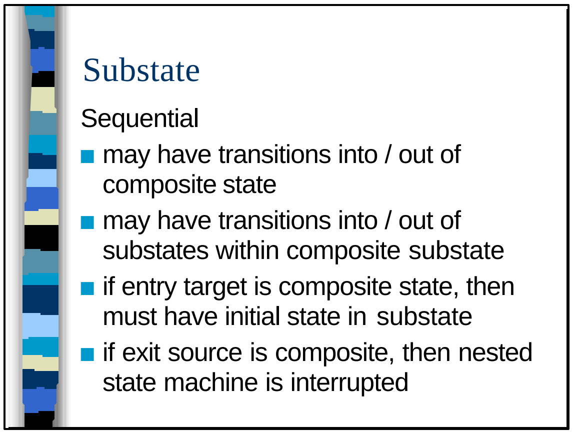
- activity: ongoing non-atomic (interruptable) execution within a state machine
  - Convention: do/*action*$_1$, *action*$_2$, …, *action*$_n$
- deferred events: list of queued events for handling in another state, list of events whose occurrence in the state is postponed until a state in which the listed events are not deferred becomes active
  - i.e. interrupt handlers
  - Convention: *event*/defer
- substates: nested structure to states
  - disjoint: sequential
  - concurrent: parallel



Substate

def'n: state nested within another state

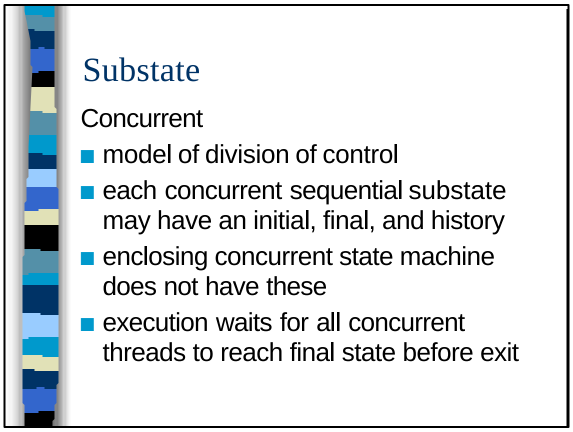
- may be nested to any level
- two types of nesting:
  - sequential: execute in sequence in context of enclosing object (or)
  - concurrent: execute in parallel in context of enclosing object (and)



Substate

Sequential

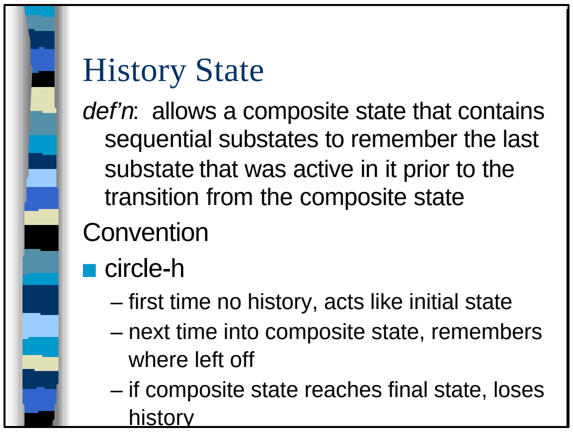
- may have transitions into / out of composite state
- may have transitions into / out of substates within composite substate
- if entry target is composite state, then must have initial state in substate
- if exit source is composite, then nested state machine is interrupted



Substate

Concurrent

- model of division of control
- each concurrent sequential substate may have an initial, final, and history
- enclosing concurrent state machine does not have these
- execution waits for all concurrent threads to reach final state before exit

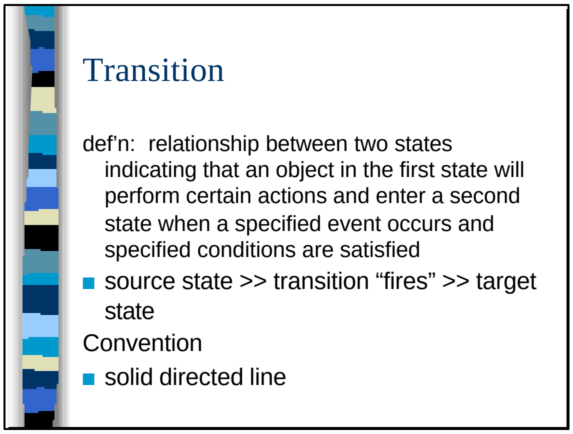


History State

def'n: allows a composite state that contains sequential substates to remember the last substate that was active in it prior to the transition from the composite state

Convention

- circle-h
  - first time no history, acts like initial state
  - next time into composite state, remembers where left off
  - if composite state reaches final state, loses history



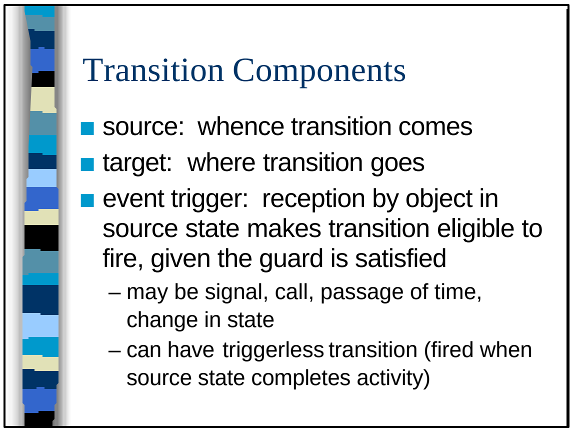
Transition

def'n: relationship between two states indicating that an object in the first state will perform certain actions and enter a second state when a specified event occurs and specified conditions are satisfied

- source state >> transition "fires" >> target state

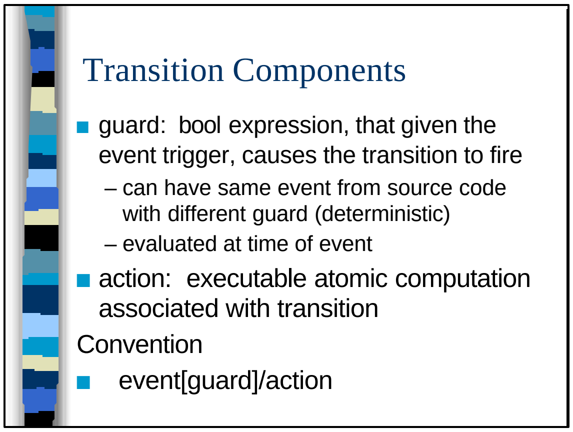
Convention

- solid directed line



Transition Components

- source: whence transition comes
- target: where transition goes
- event trigger: reception by object in source state makes transition eligible to fire, given the guard is satisfied
  - may be signal, call, passage of time, change in state
  - can have triggerless transition (fired when source state completes activity)

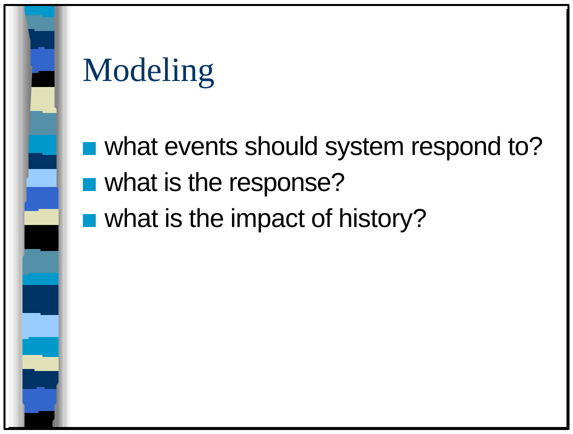


Transition Components

- guard: bool expression, that given the event trigger, causes the transition to fire
  - – can have same event from source code with different guard (deterministic)
  - – evaluated at time of event
- action: executable atomic computation associated with transition

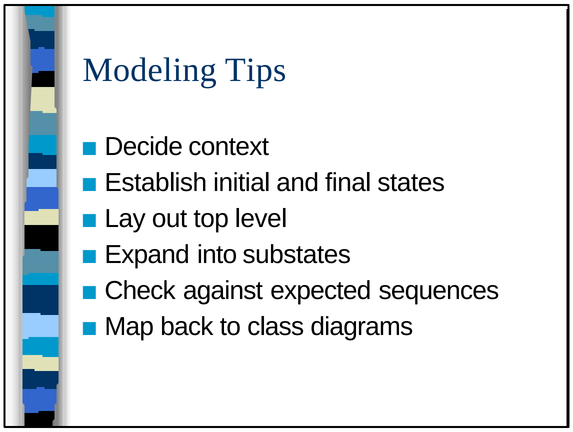
Convention

- event[guard]/action



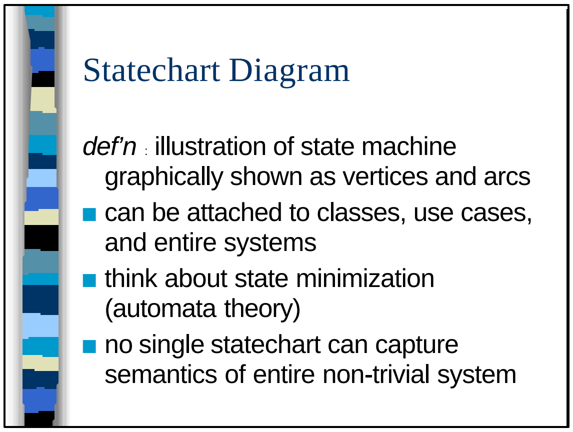
Modeling

- what events should system respond to?
- what is the response?
- what is the impact of history?



Modeling Tips

- Decide context
- Establish initial and final states
- Lay out top level
- Expand into substates
- Check against expected sequences
- Map back to class diagrams



Statechart Diagram

def'n : illustration of state machine graphically shown as vertices and arcs

- can be attached to classes, use cases, and entire systems
- think about state minimization (automata theory)
- no single statechart can capture semantics of entire non-trivial system