UML 2.0

State Diagrams

UML 2.0

- Class diagrams (+ OCL constraints)
- Package diagrams
- Component diagrams
- Deployment diagrams
- Use case diagrams
- State diagrams
- Activity diagrams
- Interaction diagrams

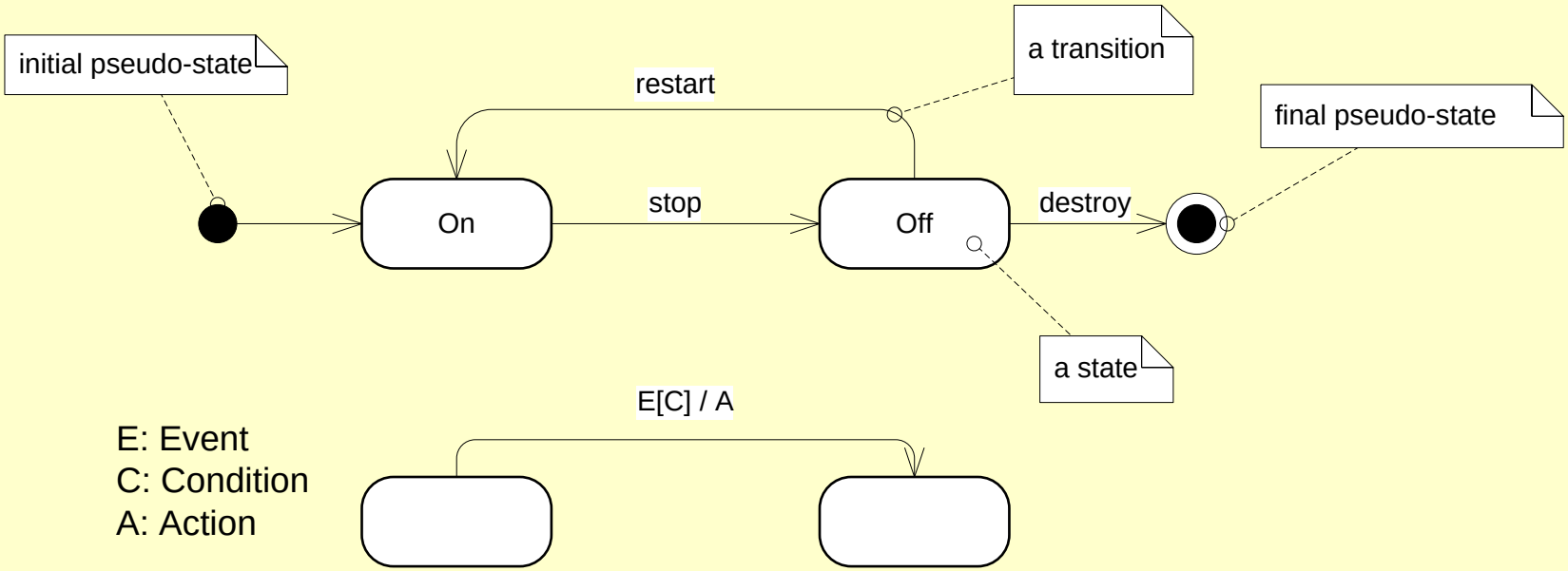
UML 2.0

- ✓ Class diagrams (+ OCL constraints)
- Package diagrams
- Component diagrams
- Deployment diagrams
- ✓ Use case diagrams
- ✓ State diagrams: today!
- Activity diagrams
- ✓ Interaction diagrams

UML State Machines

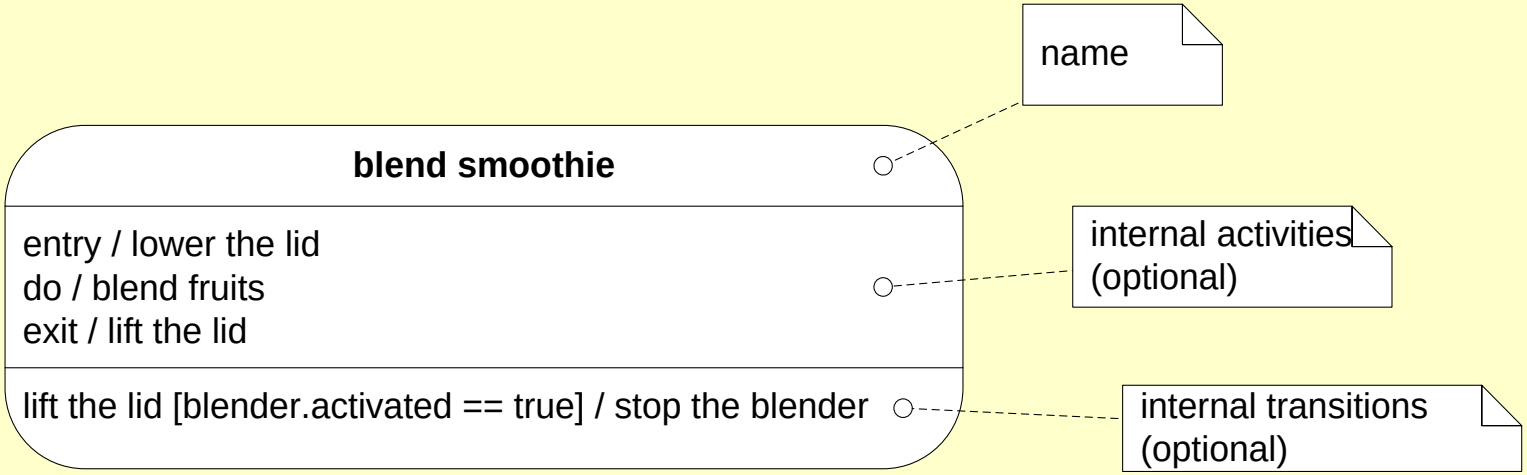
- Each UML diagram models a different point of view over the same system
- UML State (Machines) Diagrams
  - Goal: to describe the dynamic behavior of a class, an interface, a component, a use case…
  - Specifications: definition of states, transitions, guards, events, activities
  - State: a condition of being of a given entity
  - Transitions: a path between two states

State Diagrams



States

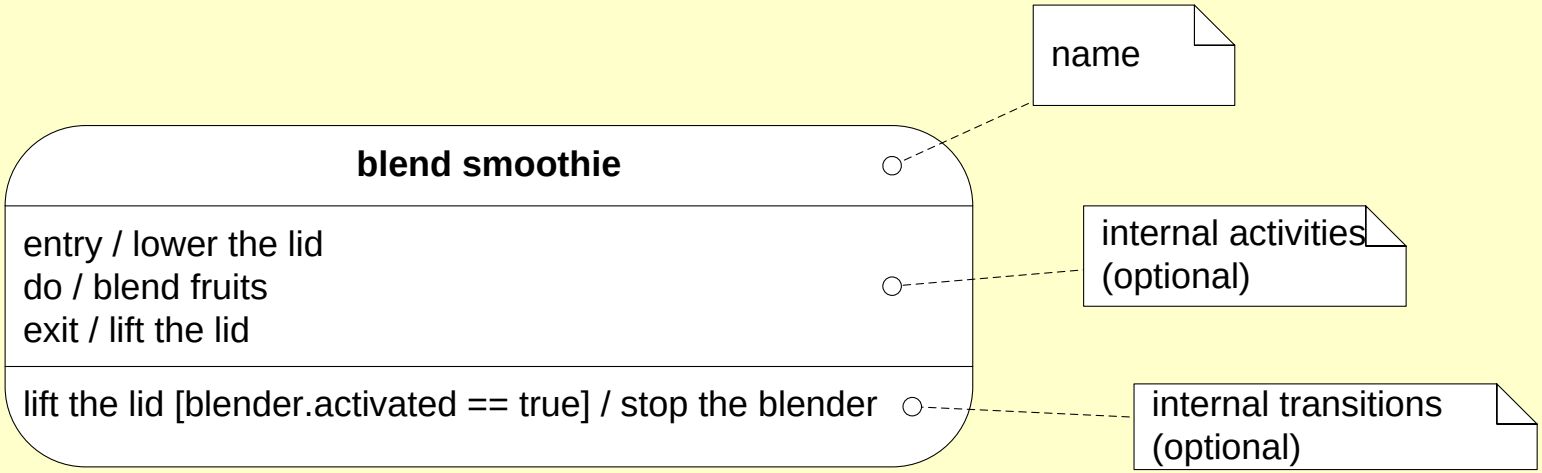
- A state is a condition of being at a certain time
- Can be *active* or *inactive*
- Characterized by stability and duration
- Example of a state in UML:



Internal Activities (1)

- An activity is a functionality executed by a system
- In UML, three kinds of activity are possible for a state:
  - **Entry**: triggered when entering a state. It is executed before anything else.
  - **Exit**: triggered when leaving a state. It is the last thing executed, just before the outgoing transition.
  - **Do**: executed as long as the state is active, after the entry activity.

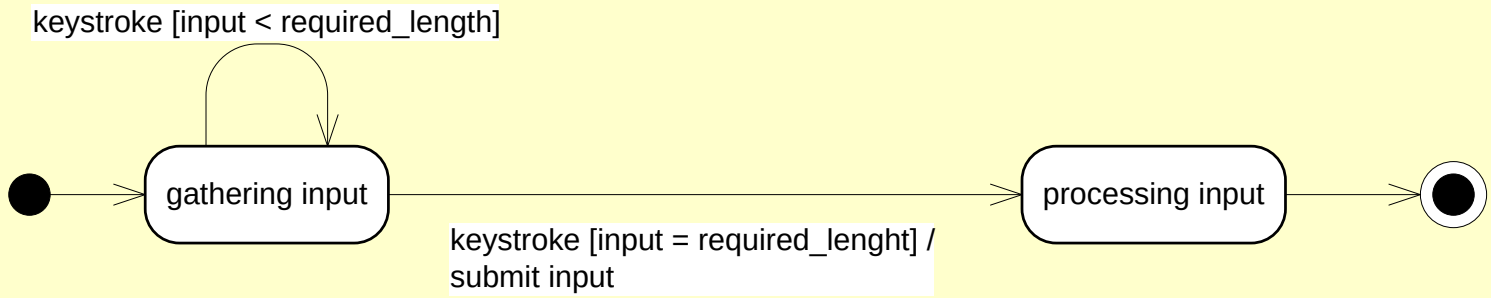
Internal Activities (2)



Transitions

- A transition is a relation/path between two states
- It represents a change of states from a *source state* to a *target state*
- Transitions are considered to take place in no time, that is, they are instantaneous
- Therefore, they cannot be interrupted

Triggers, guards, behaviors



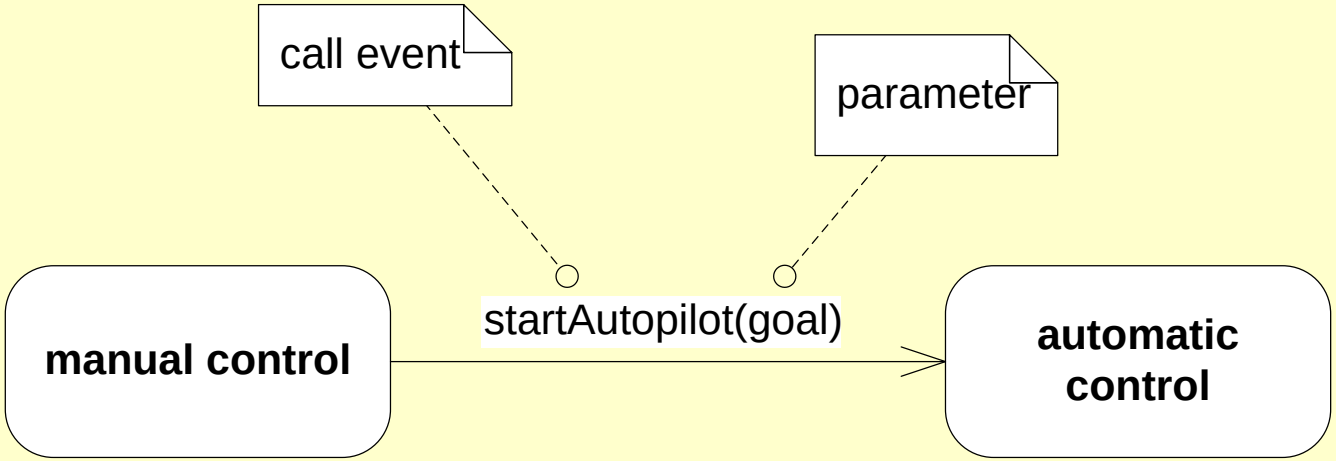
transition ::= trigger[guard] / behavior

- trigger: an event that may cause a transition
- guard: a boolean condition that permits or blocks the transition
- behavior: an uninterruptable activity that executes while the transition takes place

Events

- Transitions are triggered by events
- Events can be of 4 different types:
  - Operation calls
  - Signals (e.g. mouse clicks, external signals, etc.)
  - Conditions (**when**[x > 10])
  - Time events (**after** 10 minutes)
- When no event is specified: the transition is taken immediately after the source's internal behavior is complete

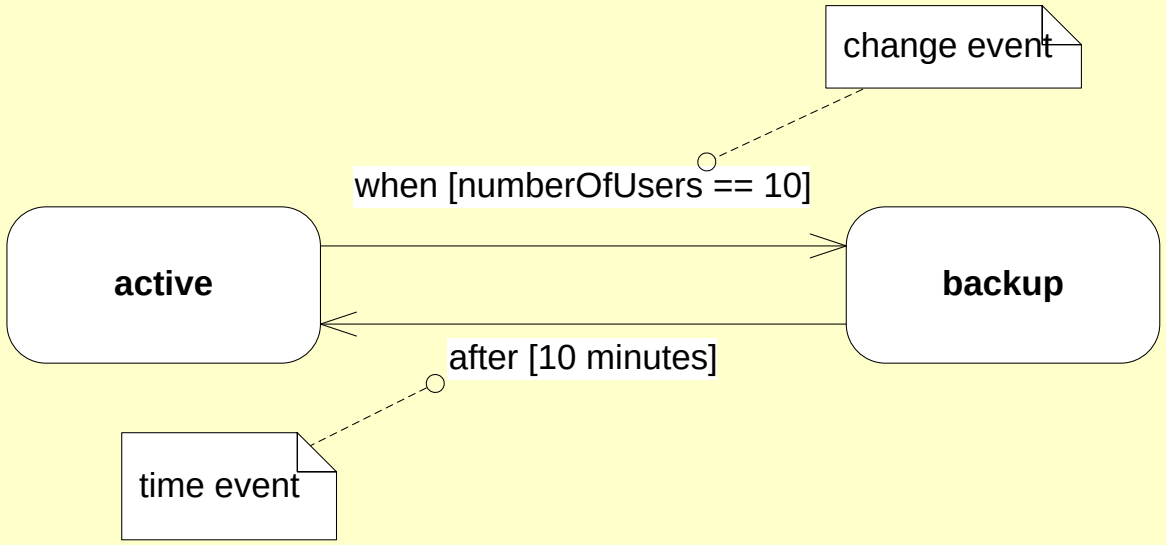
Call Events



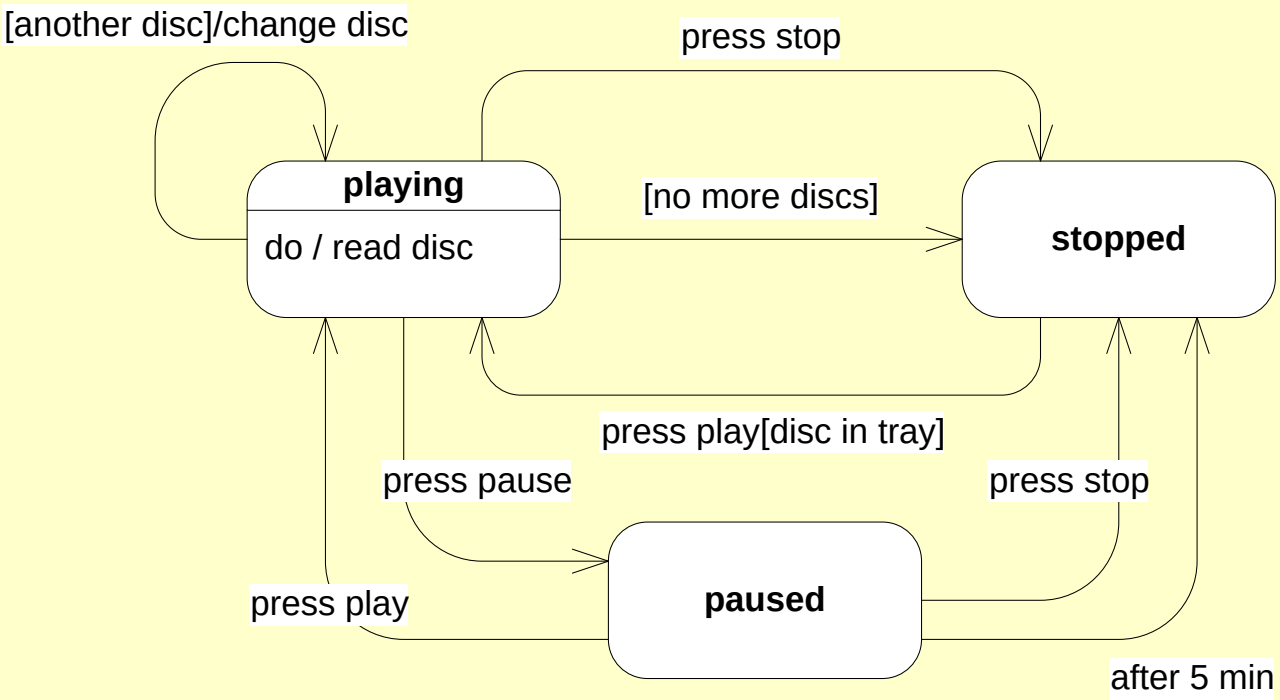
- Name and parameter of the event must be compatible to the methods of the class

Time and Change Events

- A time event occurs after the expiration of a time period
- A change event occurs if a specific constraint is fulfilled. The constraint is a boolean expression on the attributes of the actual object

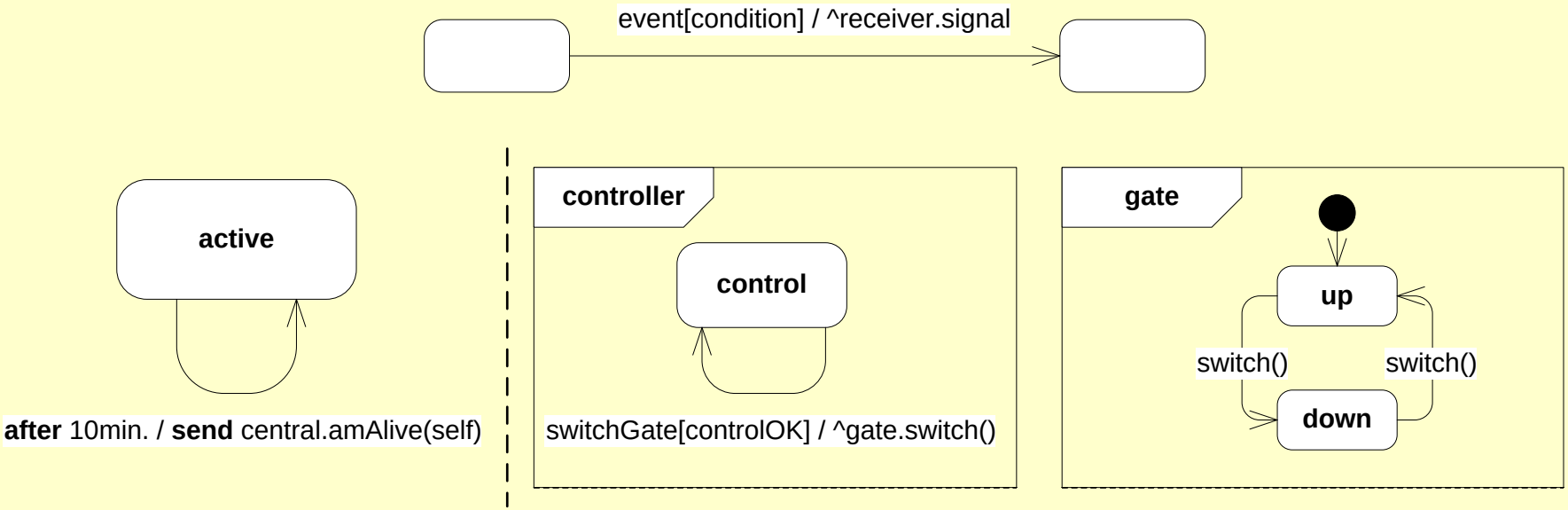


Events : example



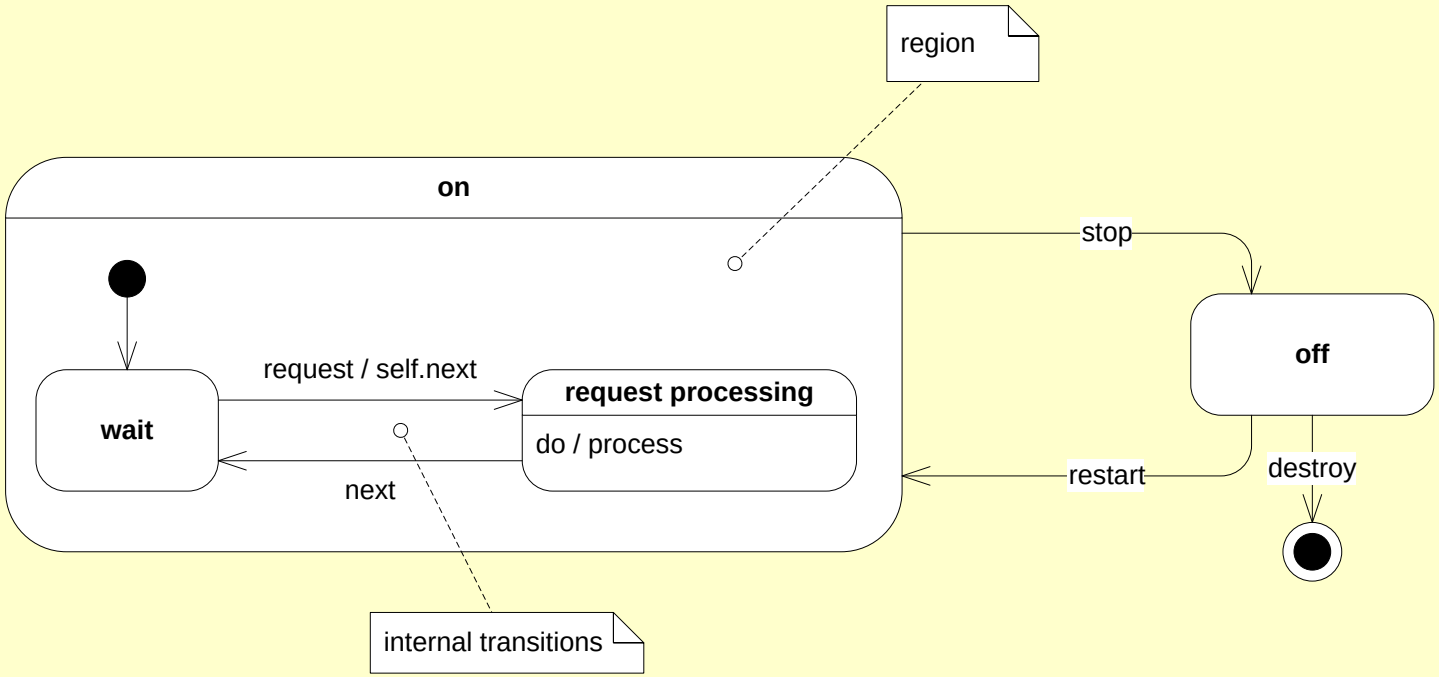
Sending Signals (Broadcasting)

- Signals can be sent to other objects during transitions: **send, ^**



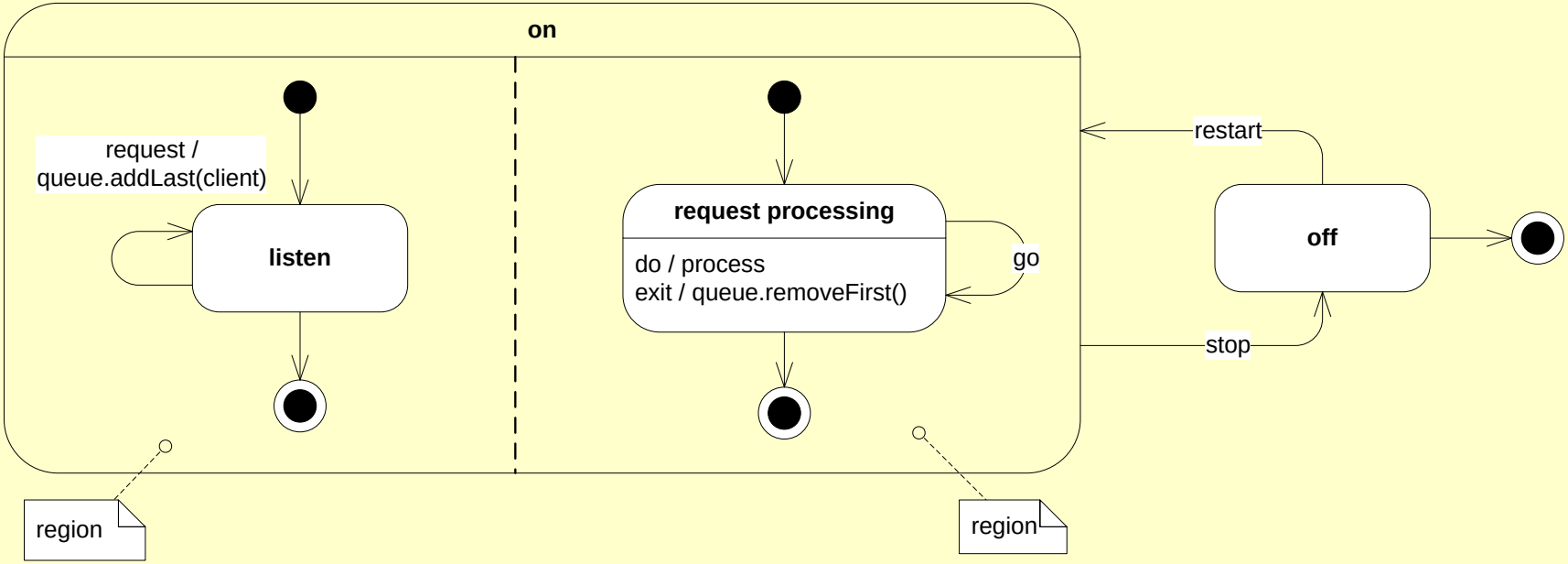
Composite States

- States can have one or more regions
- Regions describe substates

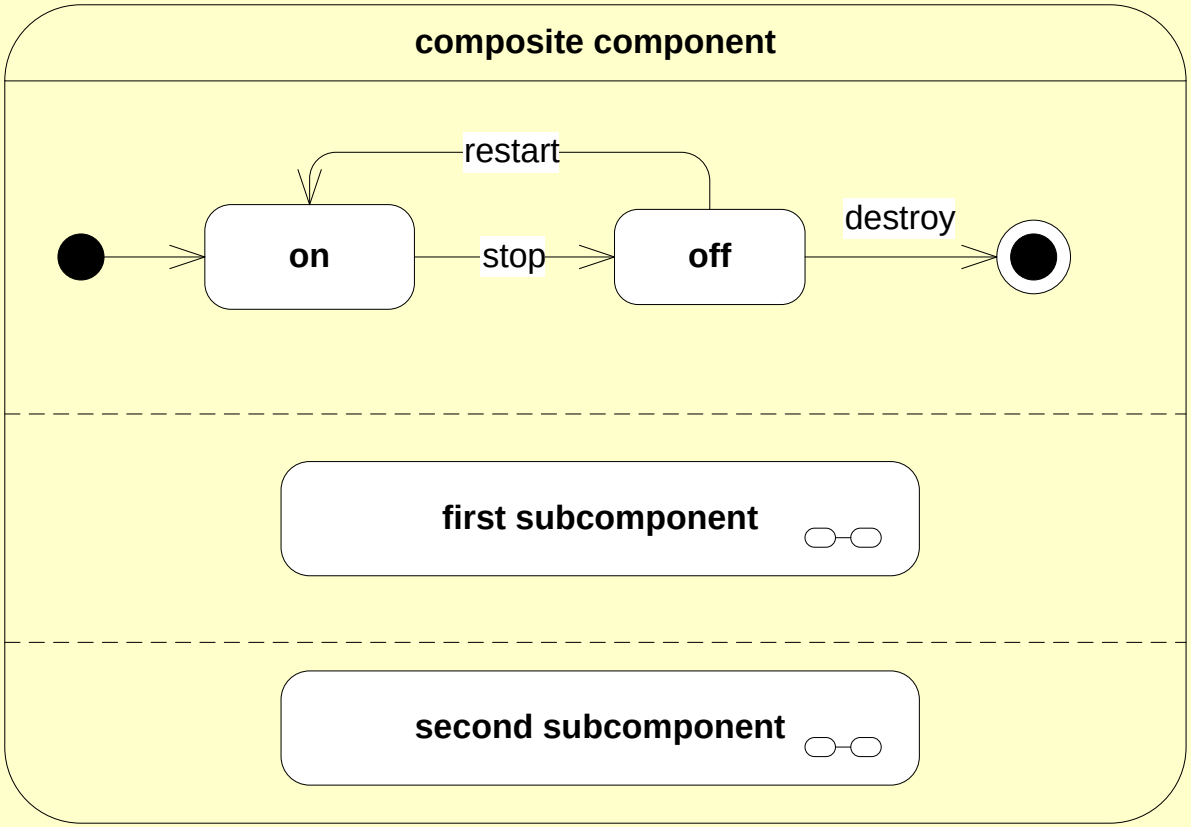


Regions & Orthogonality

- States can have one or more regions
- Regions describe substates
- A state with two or more regions is called orthogonal
- Each region is executed in parallel

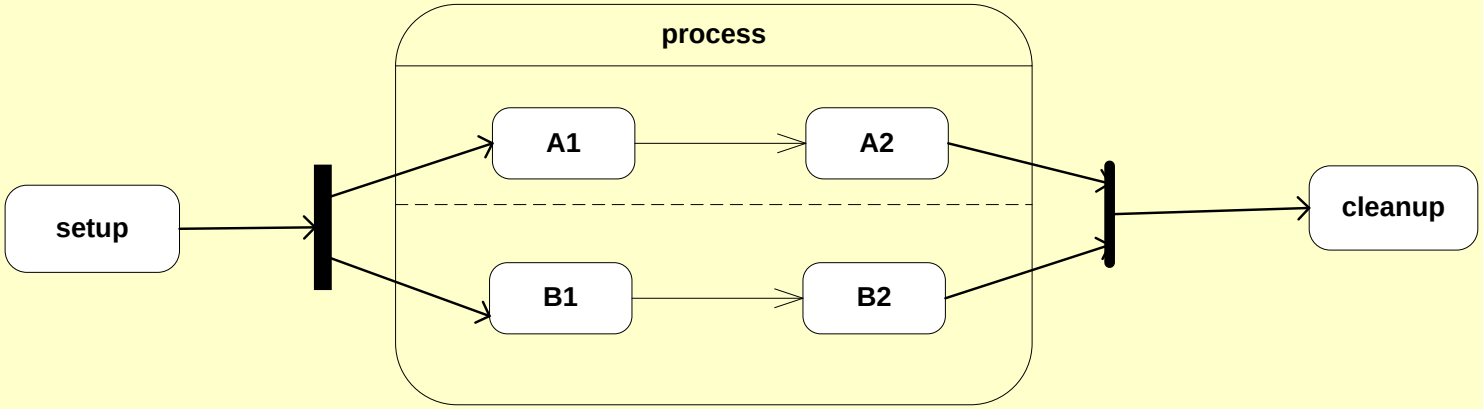


State Composition



Fork and Join

- Fork and join symbols are used to synchronize two regions
- Fork: transitions leave simultaneously
- Join: outgoing transition is activated only when all entering transitions are activated



History Pseudo-States

- Used to remember the previous state of a machine when it was interrupted