Unified Modeling Language (UML)

Sources:

- UML Tutorials fromOMG, see http://www.omg.org/uml/
- UML Specification Document v1.4, see http://www.omg.org/uml/
- The Unified Modeling Language User Guide" by GradyBooch, James Rumbaugh, Ivar Jacobson (AddisonWesley, 1999)
- Martin Fowler : UML Distilled - A Brief Guide to the Standard Object Modeling Language(Addison-Wesley, 2004）

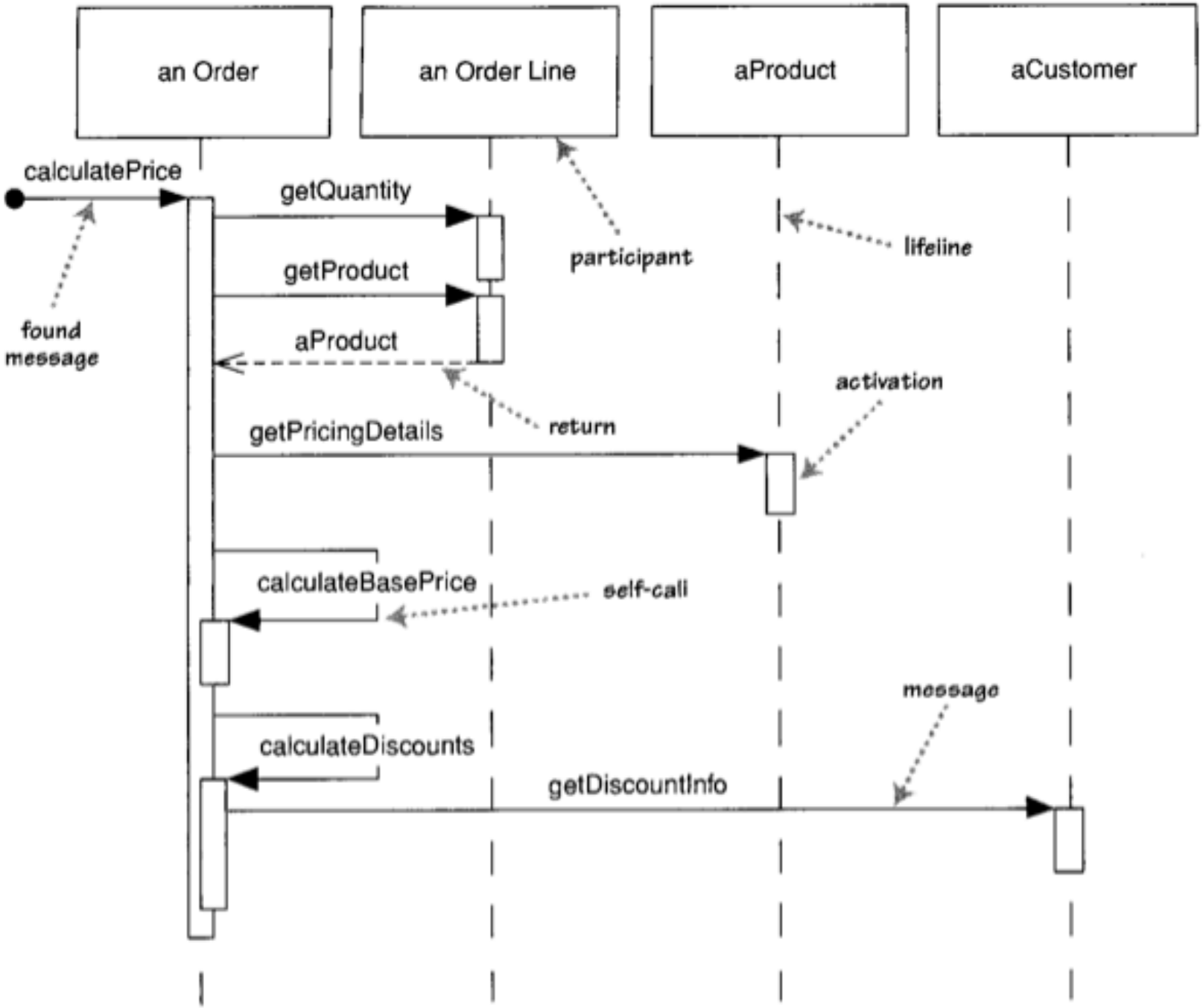
UML - Outline

- Introduction
- Structural modeling
- Behavioral modeling
  - **Use case diagrams**
  - **Interaction diagrams**
  - **State diagrams**
  - **Activity diagrams**

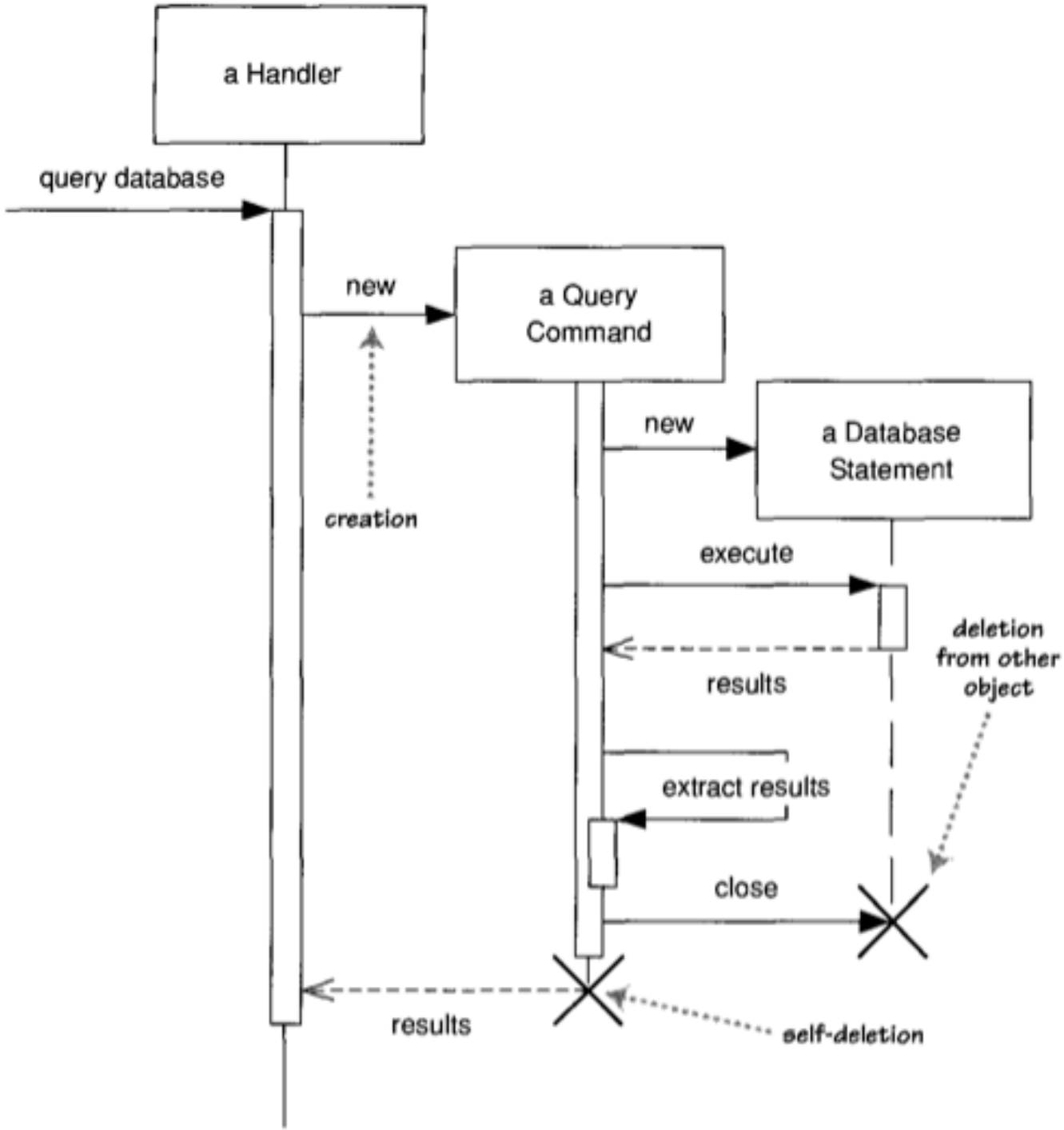
Sequence Diagram

Interaction diagrams describe how groups of objects collaborate in some behavior. The UML defines several forms of interaction diagram, of which the most common is the sequence diagram.

Sequence Diagram : Basic Elements



Sequence Diagram :Create and Delete



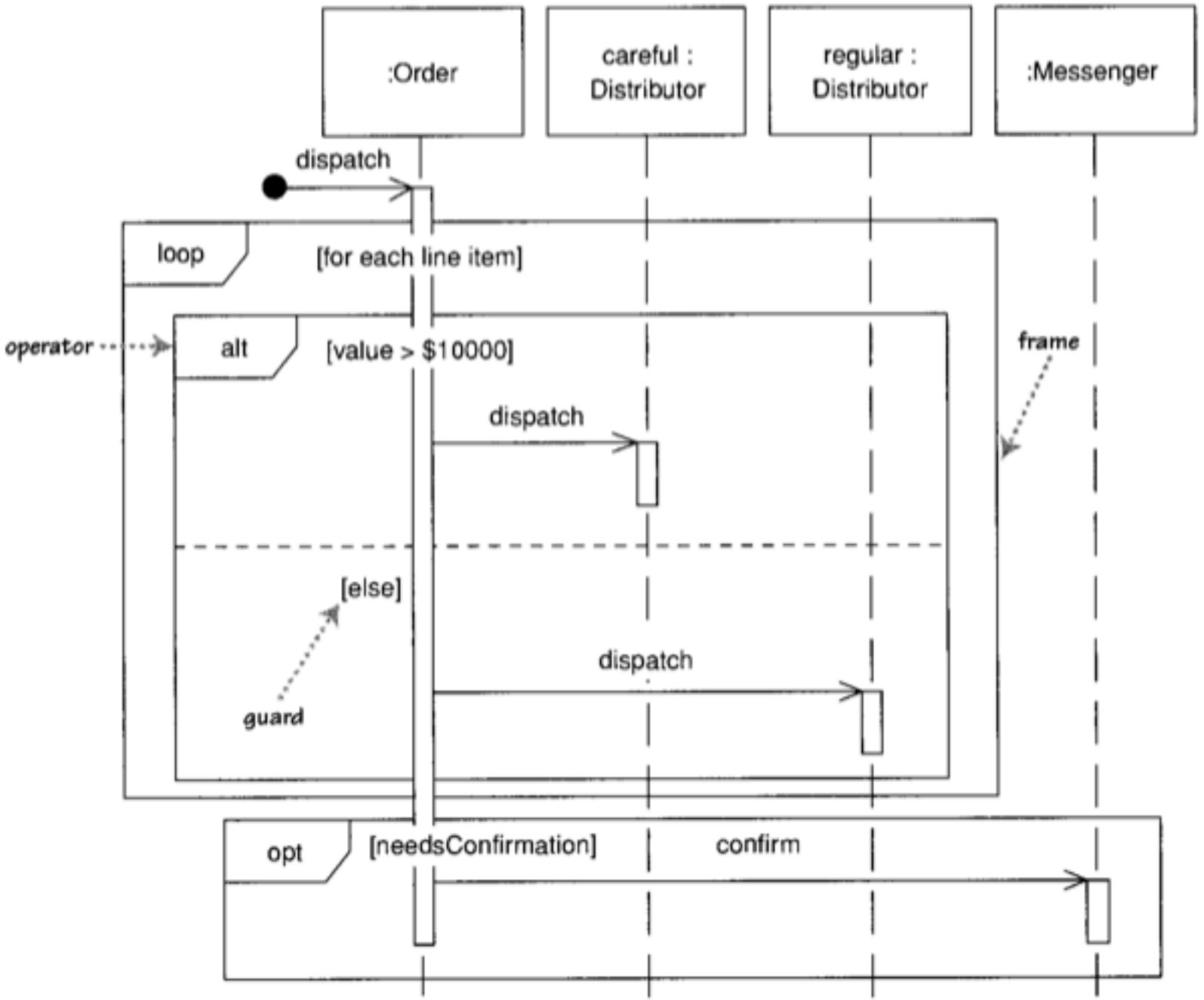
Sequence Diagram : Loops, Conditionals

Note:

- This isn't what sequence diagrams are good at.
- If you want to show control structures like this, you are better off with an activity diagram or indeed with code itself.

```
procedure dispatch
  foreach (lineitem)
    if (product.value > $10K)
      careful.dispatch
    else
      regular.dispatch
    end if
  end for
  if (needsConfirmation) messenger.confirm
end procedure
```

Sequence Diagram : Loops, Conditionals



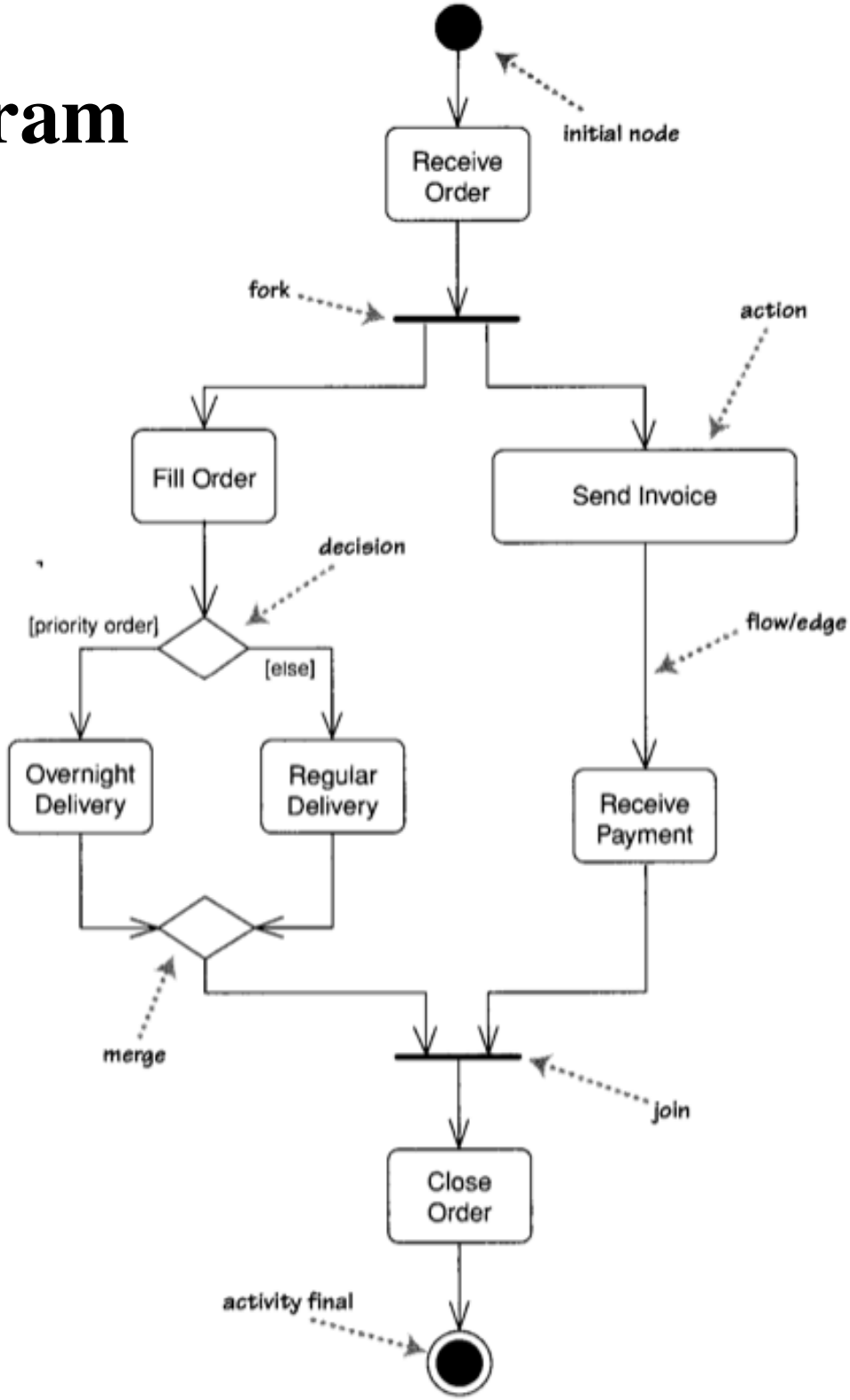
Sequence Diagram : When to Use

You should use sequence diagrams when you want to look at the behavior of several objects within a single use case. Sequence diagrams are good at showing collaborations among the objects; they are not so good at precise definition of the behavior.

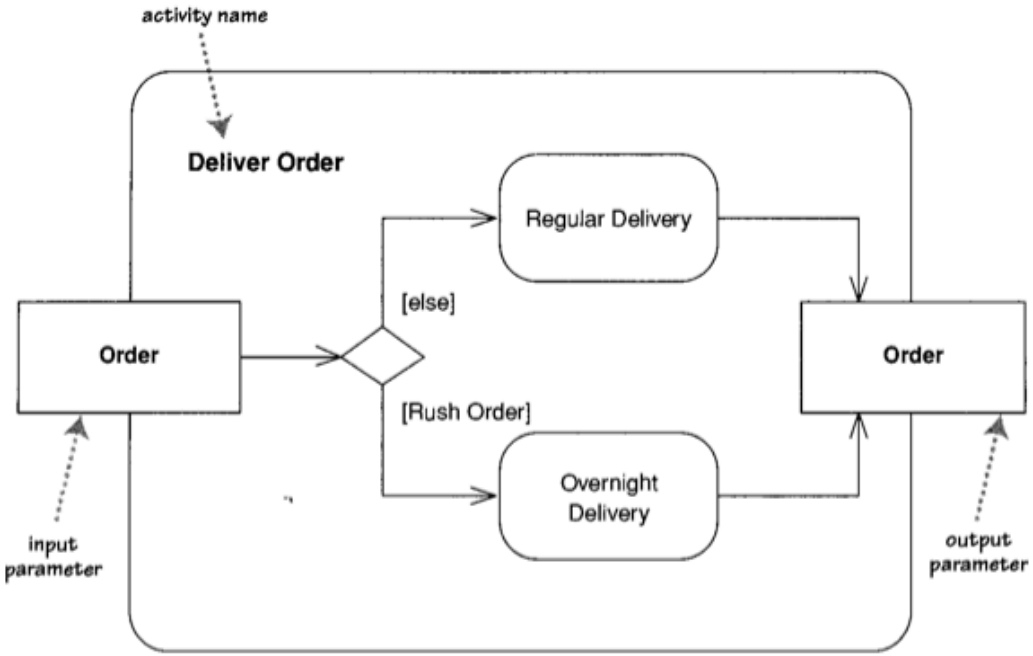
Activity Diagrams

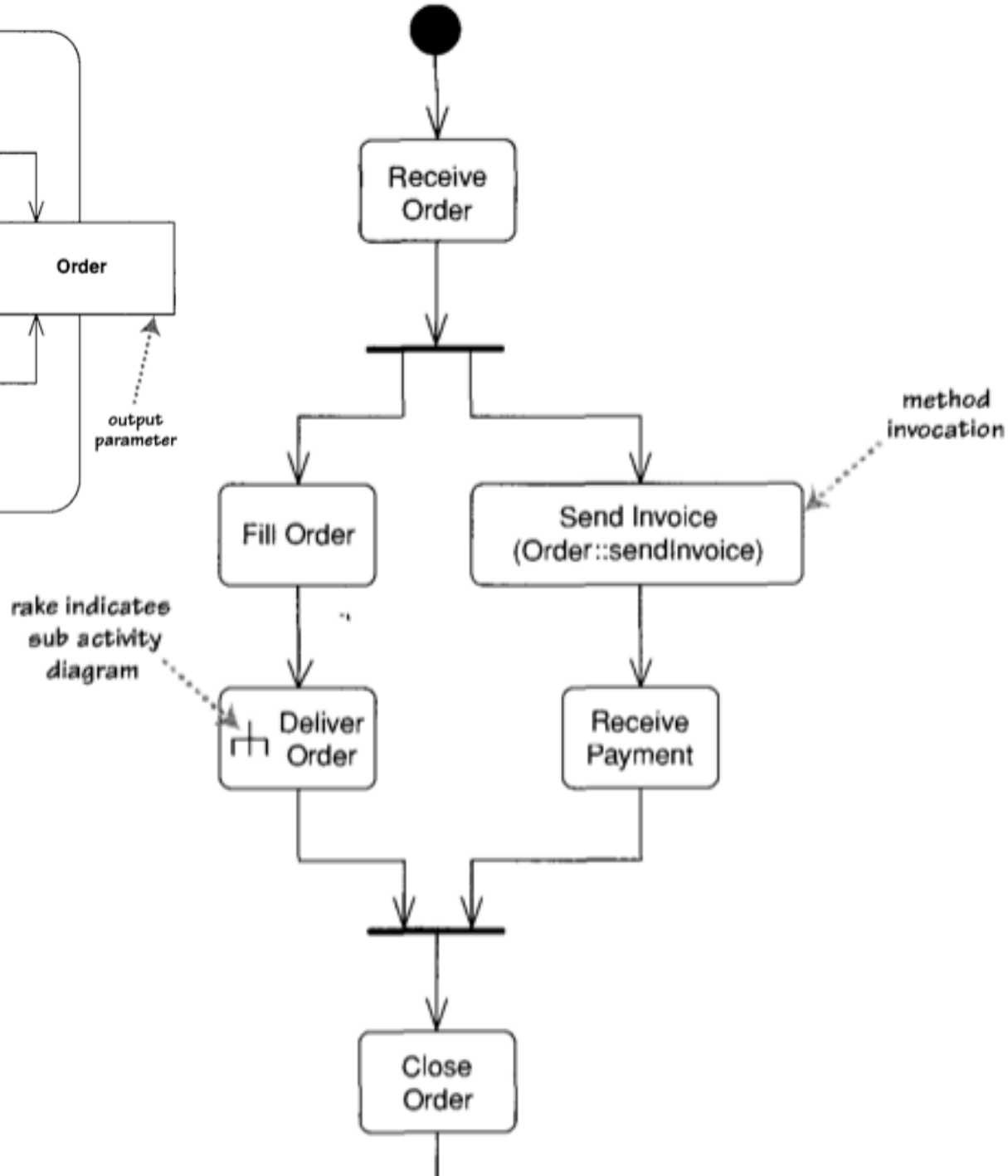
- Activity diagrams are a technique to describe procedural logic, business process, and work flow
- In many ways, they play a role similar to flowcharts, but the principal difference between them and flowchart notation is that they support parallel behavior.

A Simple Activity Diagram



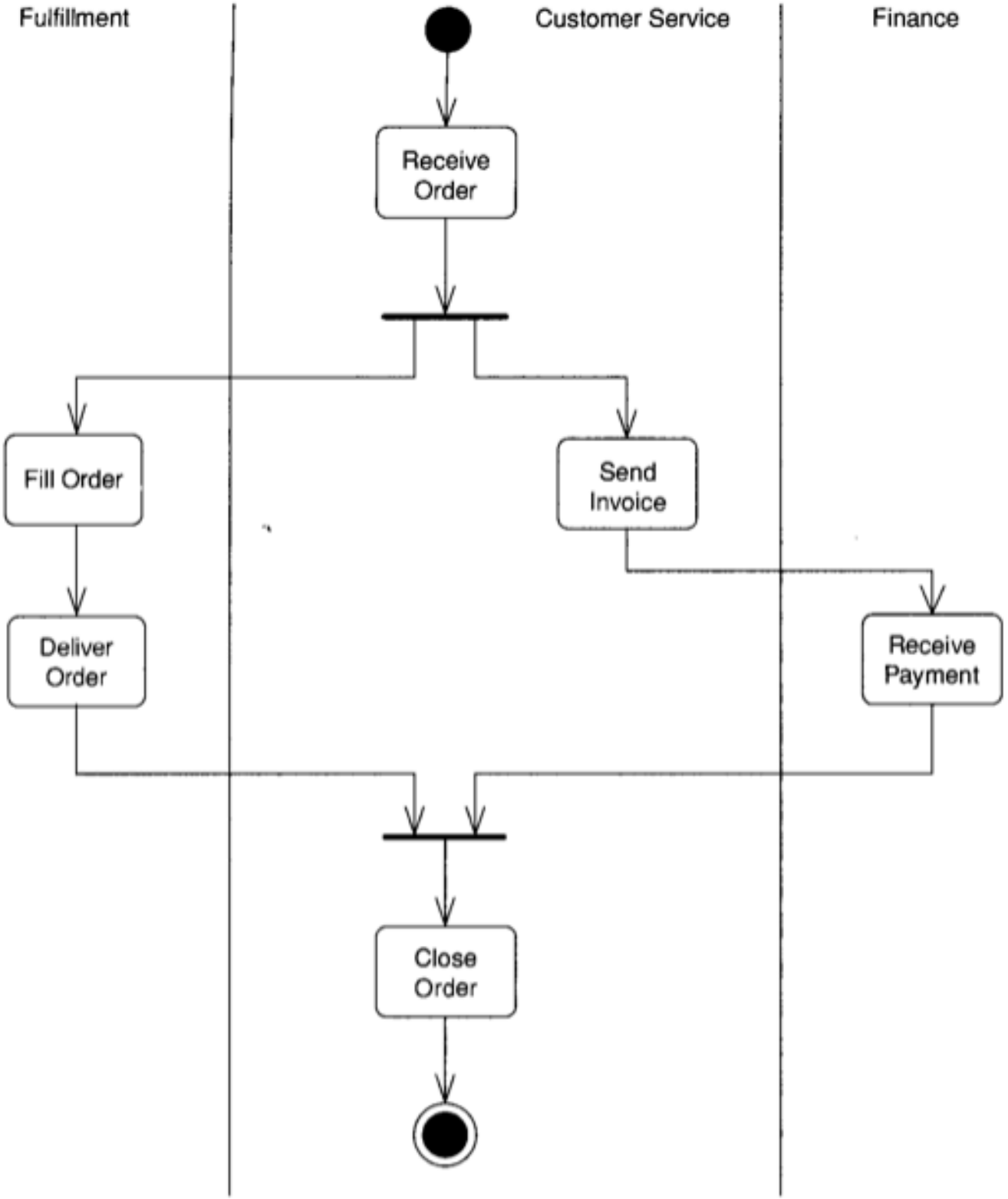
Decomposing an Action





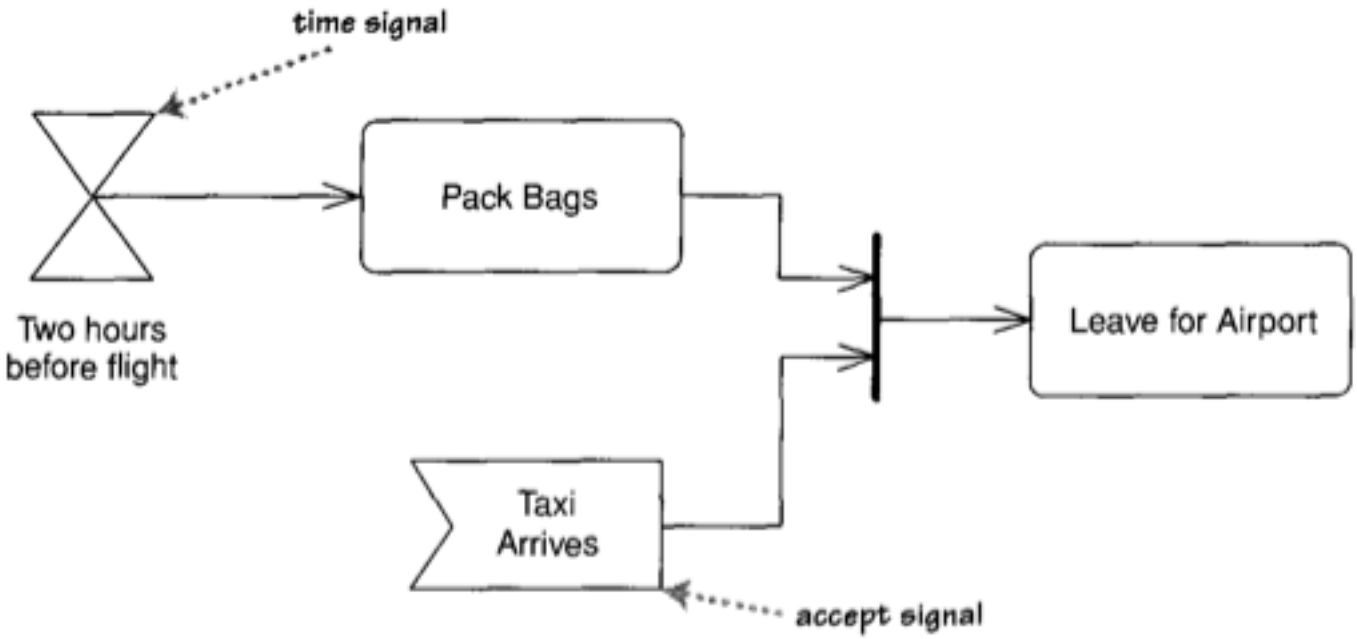
Partitions

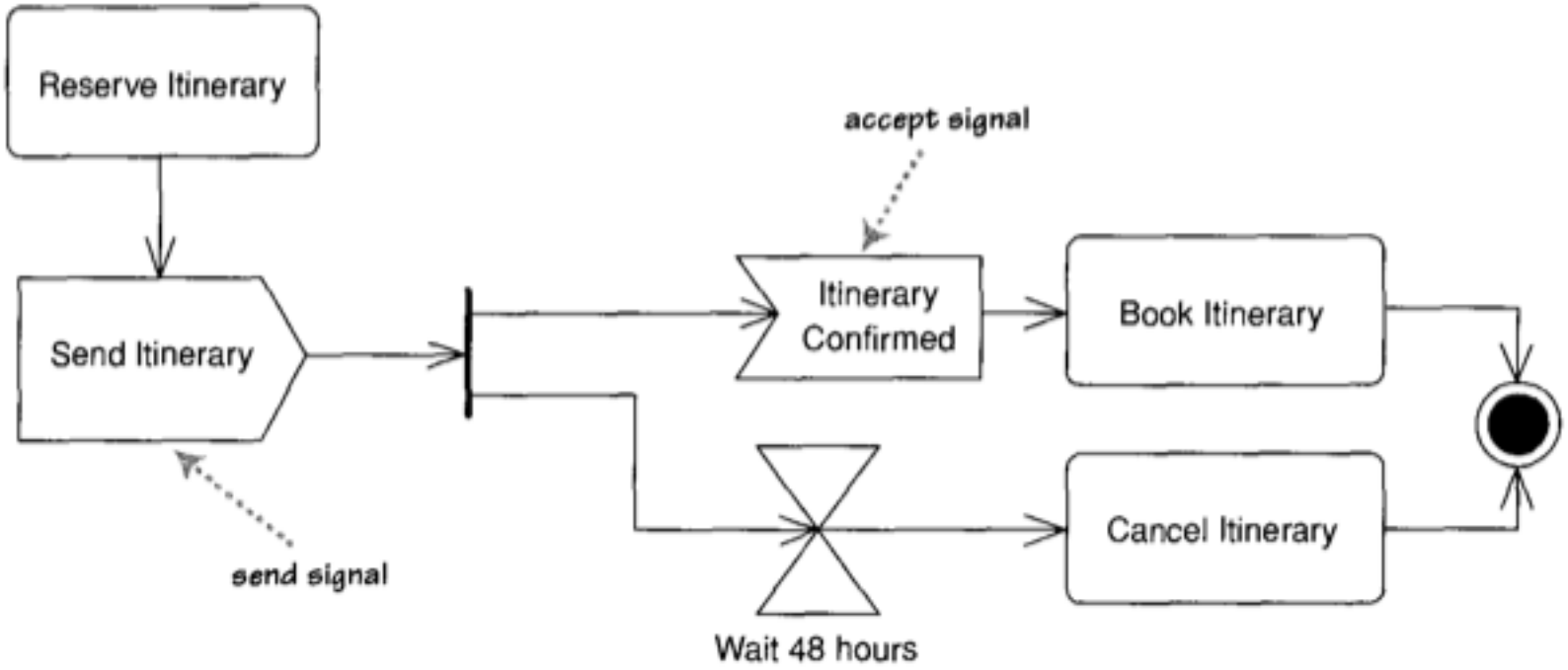
If you want to show who does what, you can divide an activity diagram into partitions



Signals

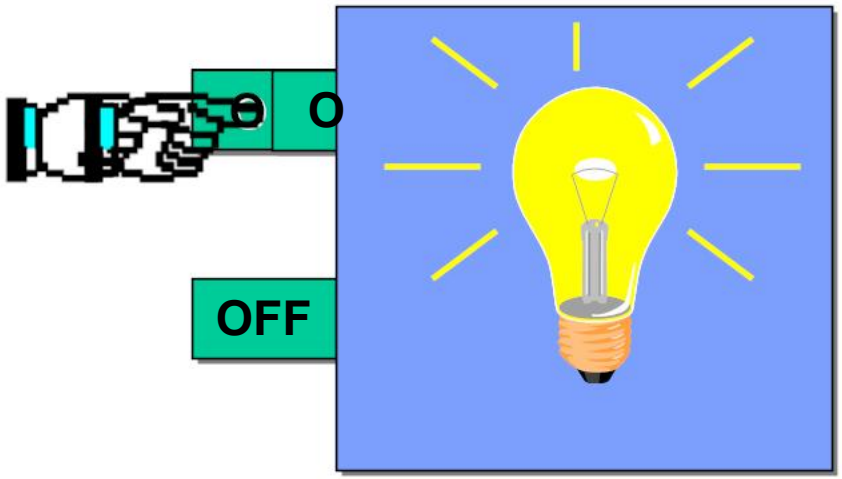
Actions can also respond to signals.





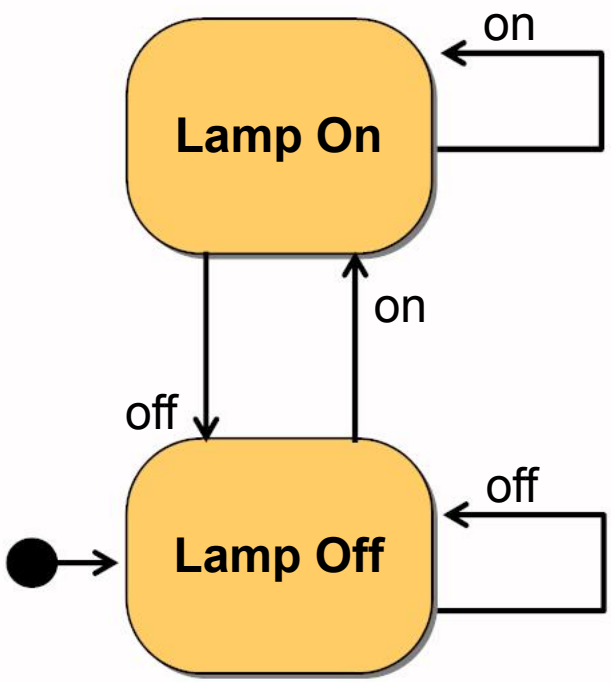
Automata

- A machine whose output behavior is not only a direct consequence of the current input, but of some past history of its inputs
- Characterized by an internal state which represents this past experience



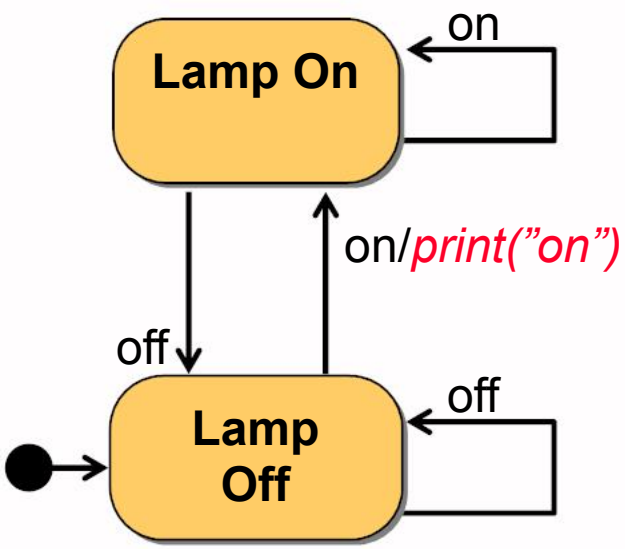
State Machine (Automaton) Diagram

- Graphical rendering of automata behavior

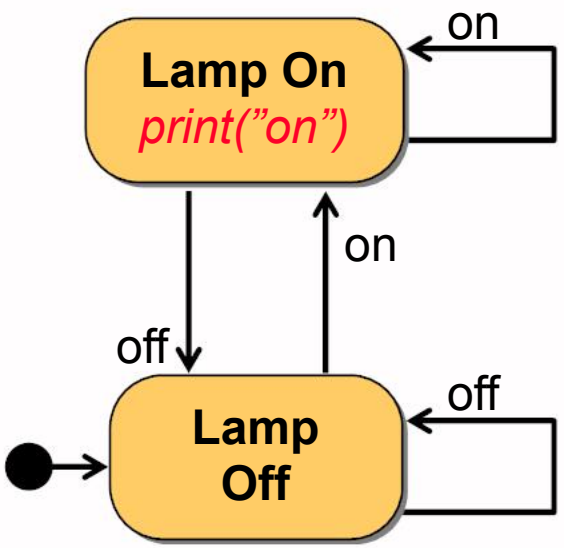


Outputs and Actions

- As the automaton changes state it can generate outputs:



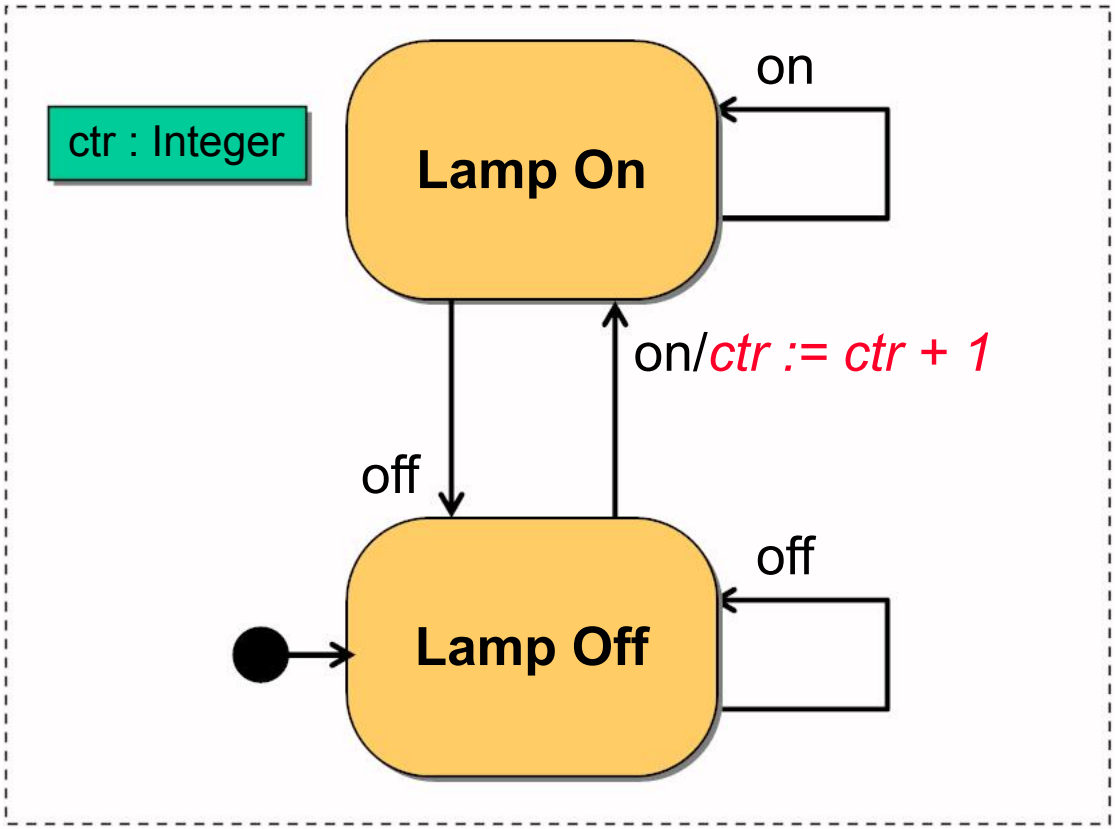
Mealy automaton



Moore automaton

Extended State Machines

- Addition of variables ("extended state")

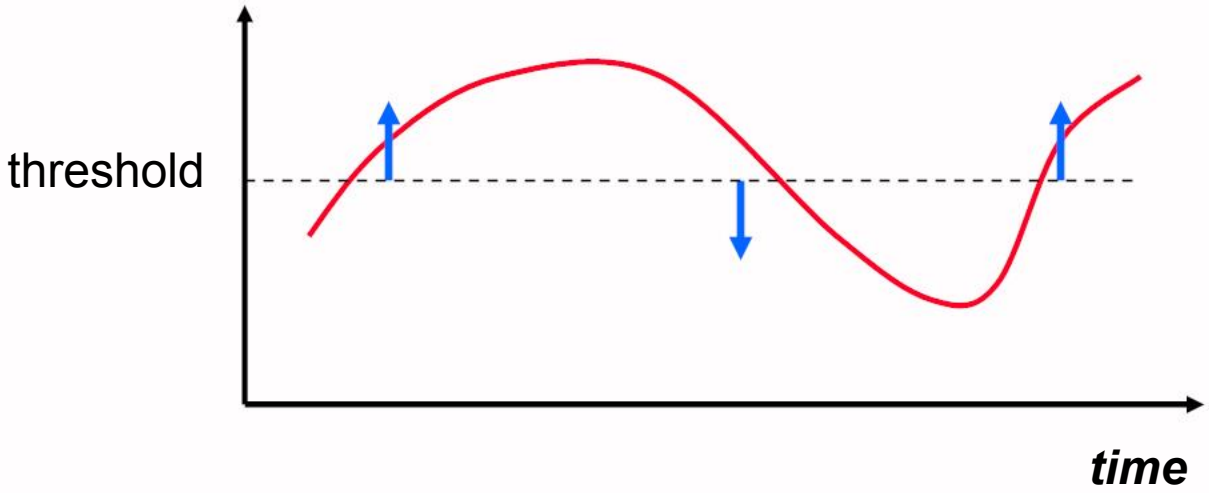


A Bit of Theory

- An extended (Mealy) state machine is defined by:
  - a set of input signals (input alphabet)
  - a set of output signals (output alphabet)
  - a set of states
  - a set of transitions
    - triggering signal
    - action
  - a set of extended state variables
  - an initial state designation
  - a set of final states (if terminating automaton)

What Kind of Behavior?

- In general, state machines are suitable for describing event-driven, discrete behavior
  - inappropriate for modeling continuous behavior



Event-Driven Behavior

- Event = a type of observable occurrence
  - interactions:
    - synchronous object operation invocation (call event)
    - asynchronous signal reception (signal event)
  - occurrence of time instants (time event)
    - interval expiry
    - calendar/clock time
  - change in value of some entity (change event)
- Event Instance = an instance of an event (type)
  - occurs at a particular time instant and has no duration

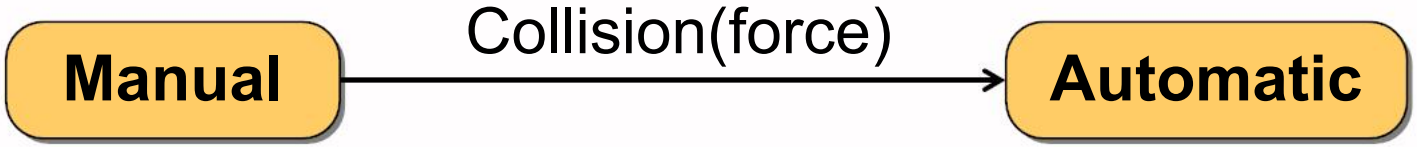
The Behavior of What?

- In principle, anything that manifests event-driven behavior
  - There is no support currently in UML for modeling continuous behavior
- In practice:
  - the behavior of individual objects
  - object interactions
- The dynamic semantics of UML state machines are currently mainly specified for the case of active objects

Events

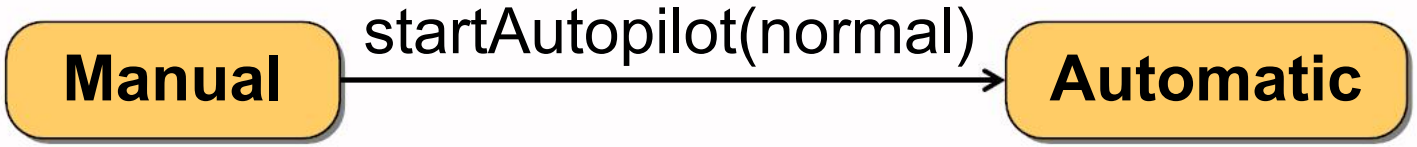
- **Event**: Spec of a significant occurrence that has a location in time and space.
  - State machine: occurrence of a stimulus that triggers a state transition.
- **Four** kinds of events
  - Signals
  - Calls
  - Passing of time
  - Change in state

Signals as Triggers



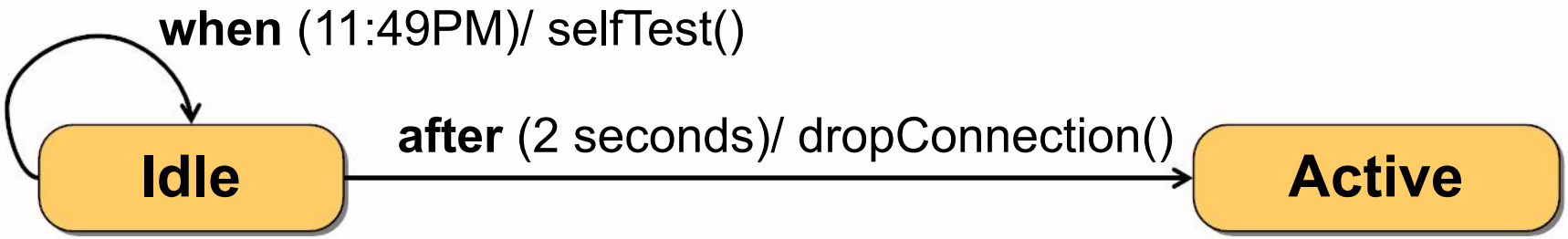
- Sending of a signal
- Asynchronous

Call Events as Triggers



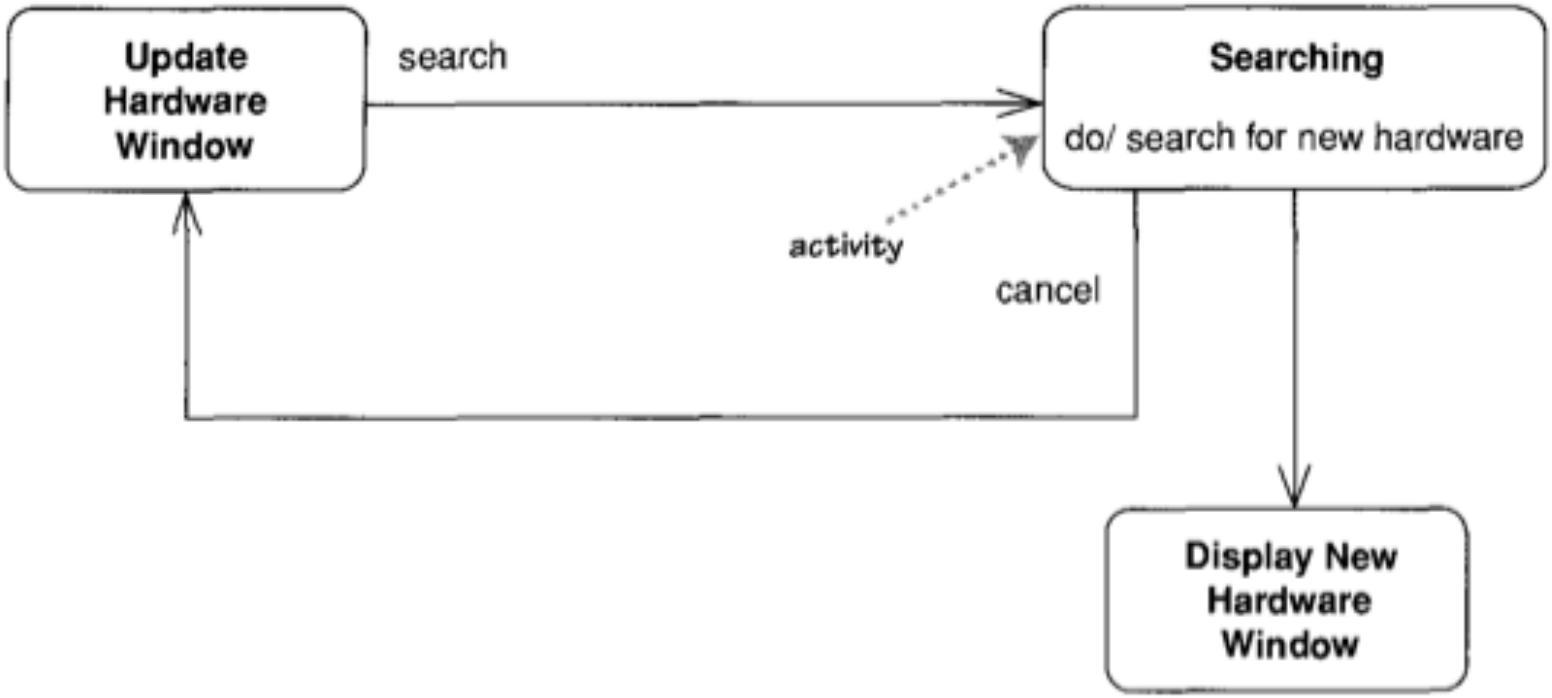
- Dispatch of an operation
- Synchronous

Time and Change events



- **Time event**: Represents passage of time (keyword: **after**)
- **Change event**: Change in state or satisfaction of some condition (keyword: **when**)

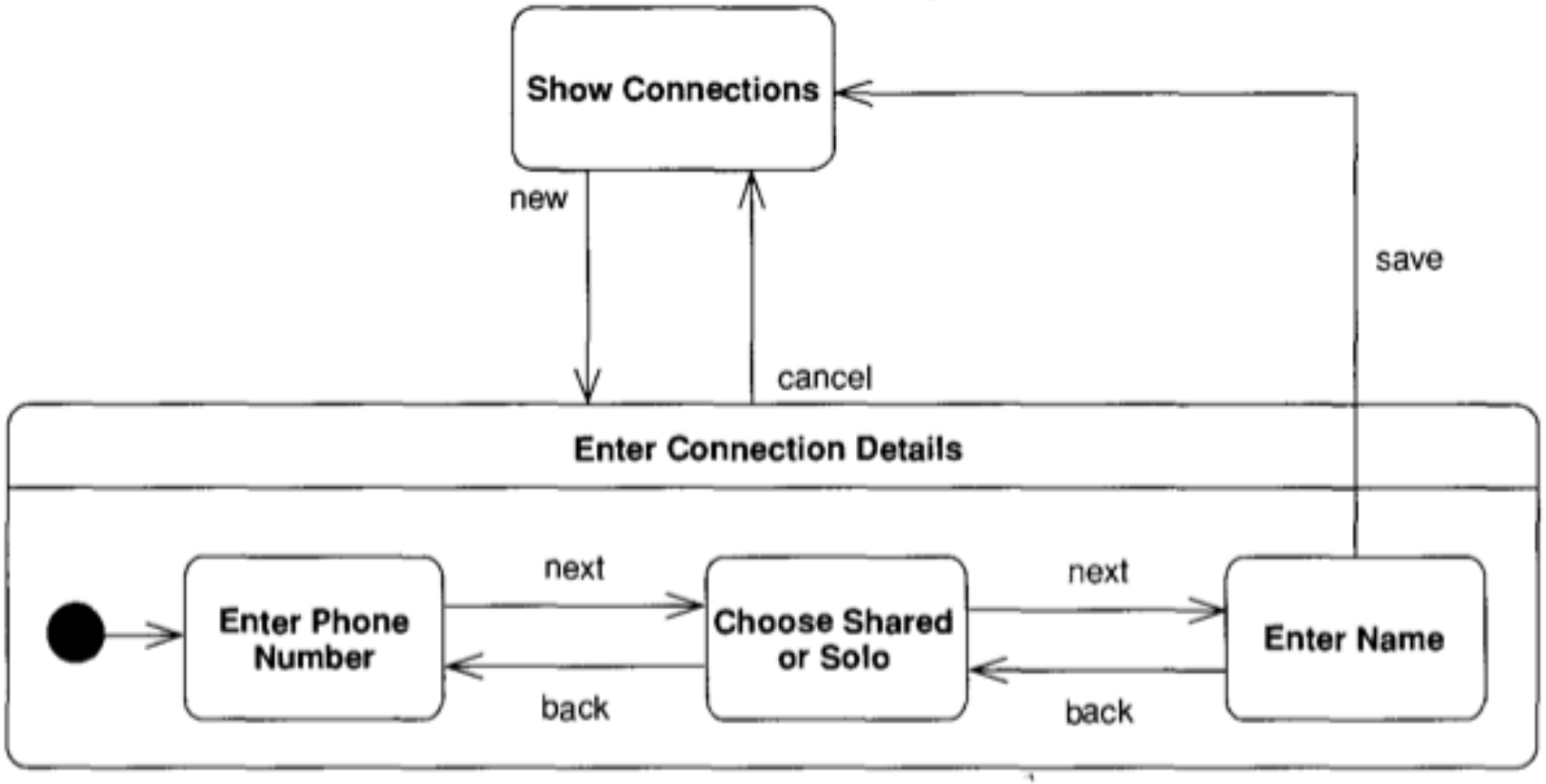
Activity States



We can have states in which the object is doing some ongoing work.

Superstates

If several states share common transitions and internal activities. In these cases, you can make them substates and move the shared behavior into a superstate



Implementing State Diagrams

A state diagram can be implemented in three main ways:

- State tables
- Nested switch
- **The State pattern**