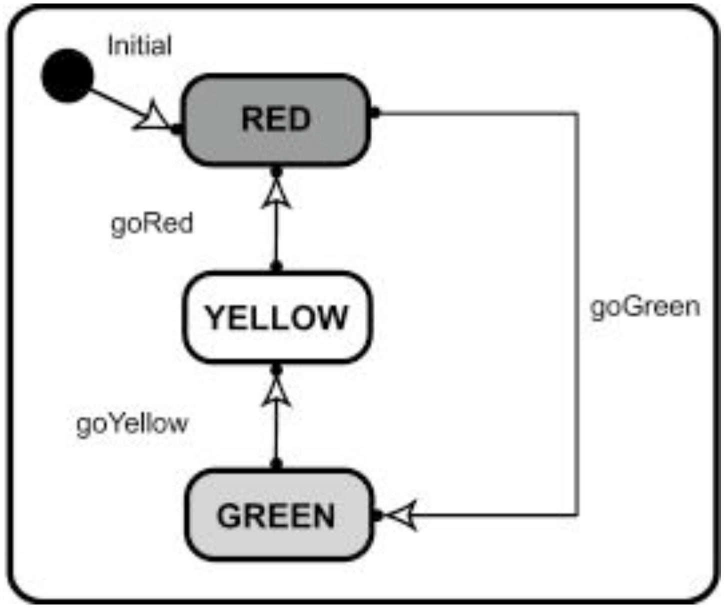
State Charts

Mehran Najafi

Reactive Systems

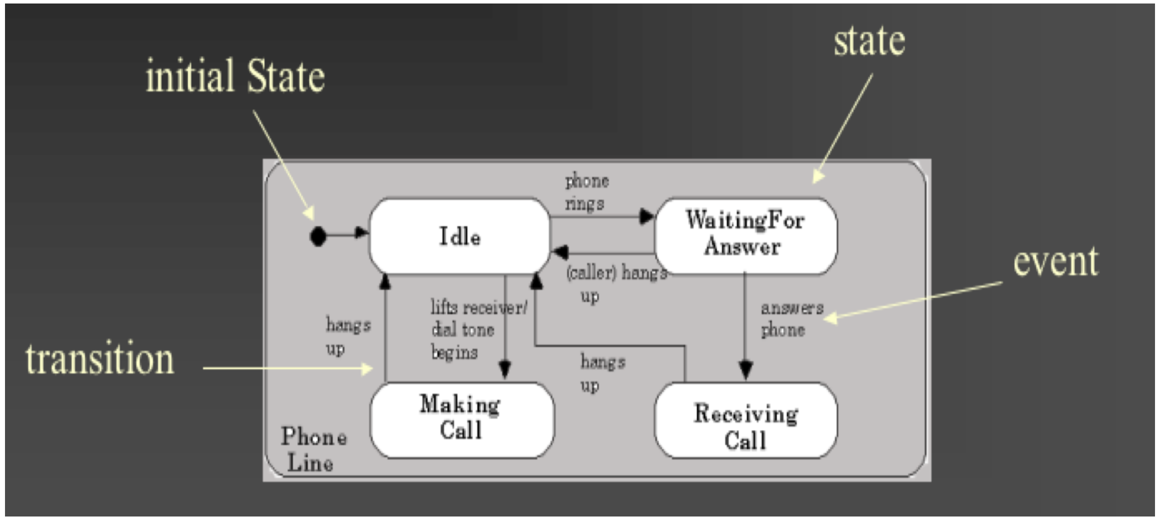
- A reactive, event-driven, object is one whose behavior is best characterized by its response to events dispatched from outside its context.



Traffic light statechart

Statechart Diagrams

- Graph whose nodes are states and whose directed arcs are transitions labeled by event names.
- Describe the dynamic behavior of a system.
- Consider a system as a finite state machine.

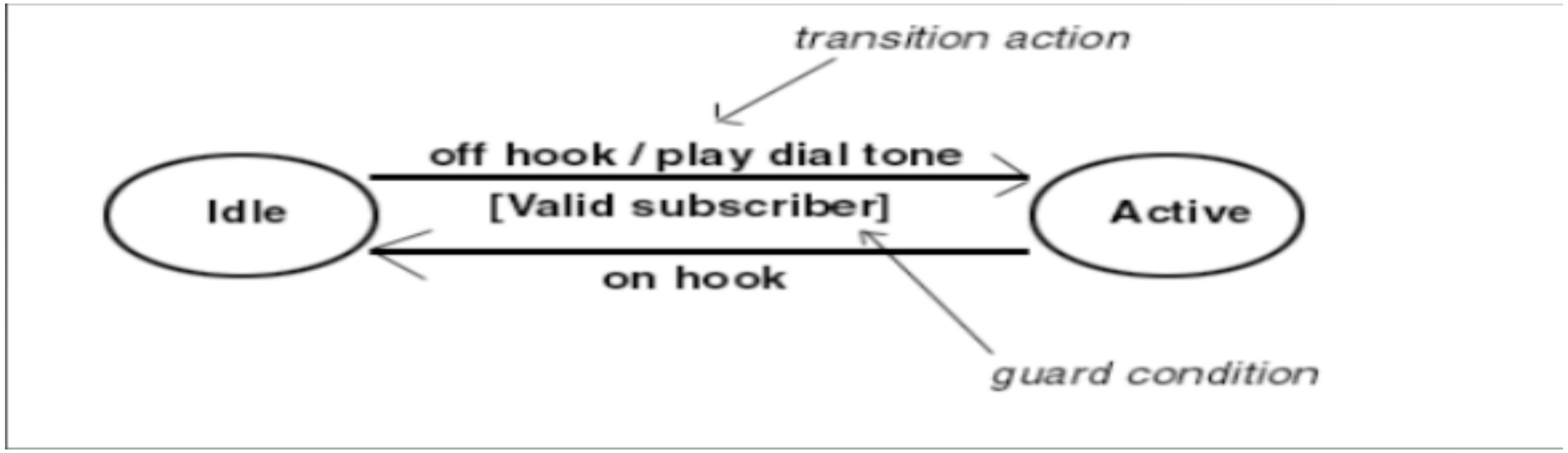


Statechart Components

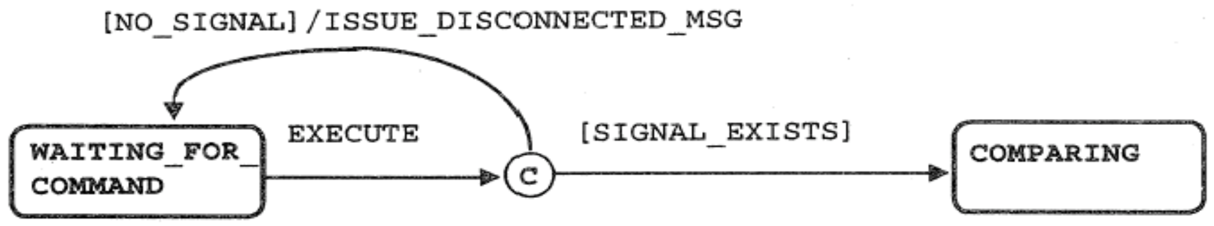
Statechart Components

- **State – Is the condition of an object at a specific time.**
- **Event – Is an occurrence that triggers the state.**
- **Transition – Involves going from one state to the other when an event occurs.**

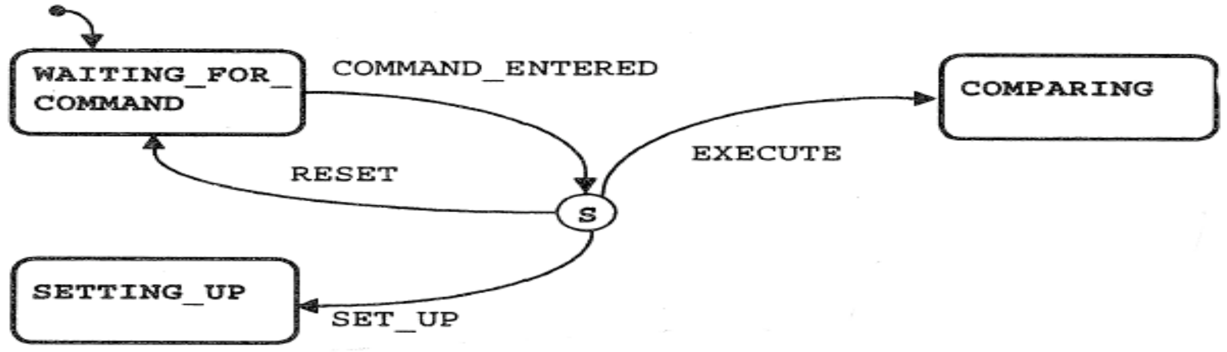
Event [Guard] / Action



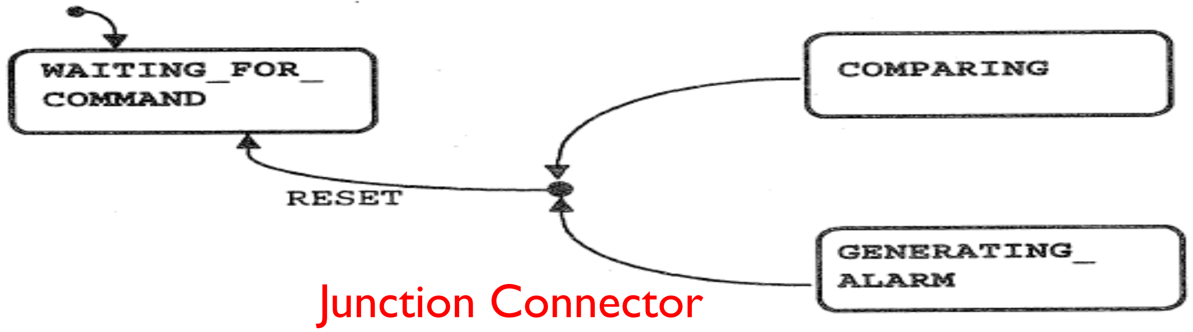
Connectors



Condition Connector



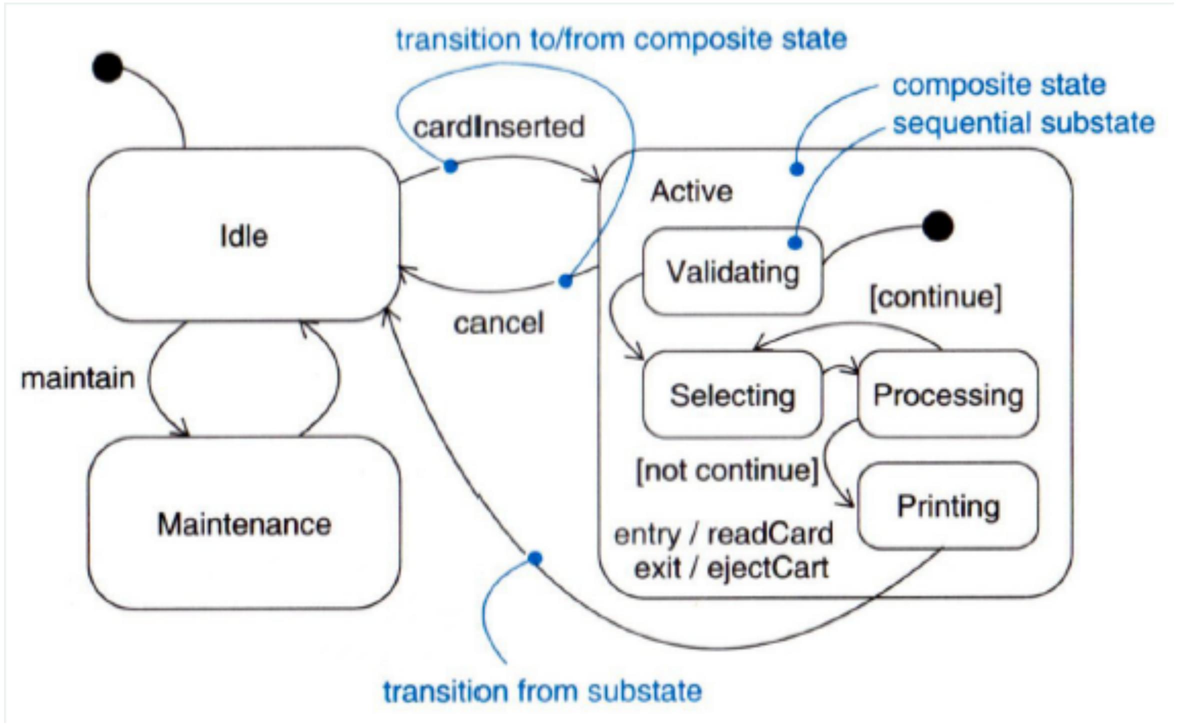
Switch Connector



Junction Connector

Composite State

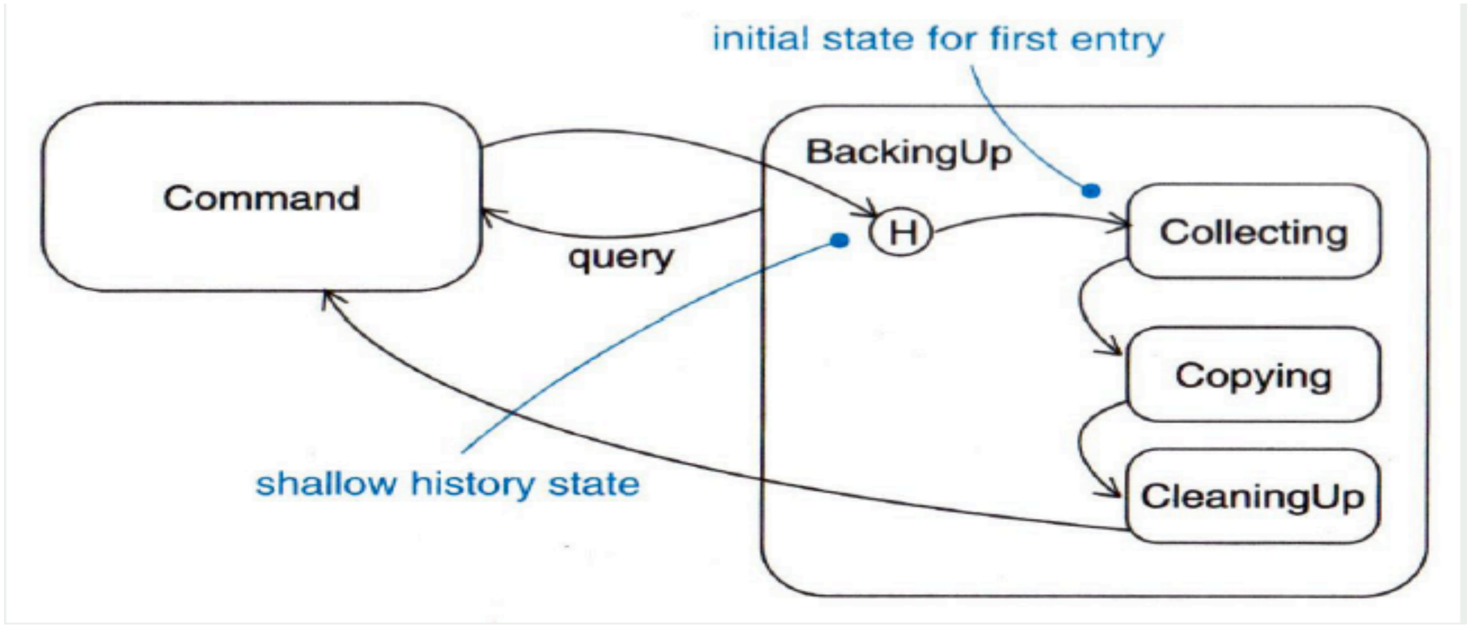
- Is a state that contains other states.
- Simplifies the modeling of complex behaviors.



ATM Statechart

History states

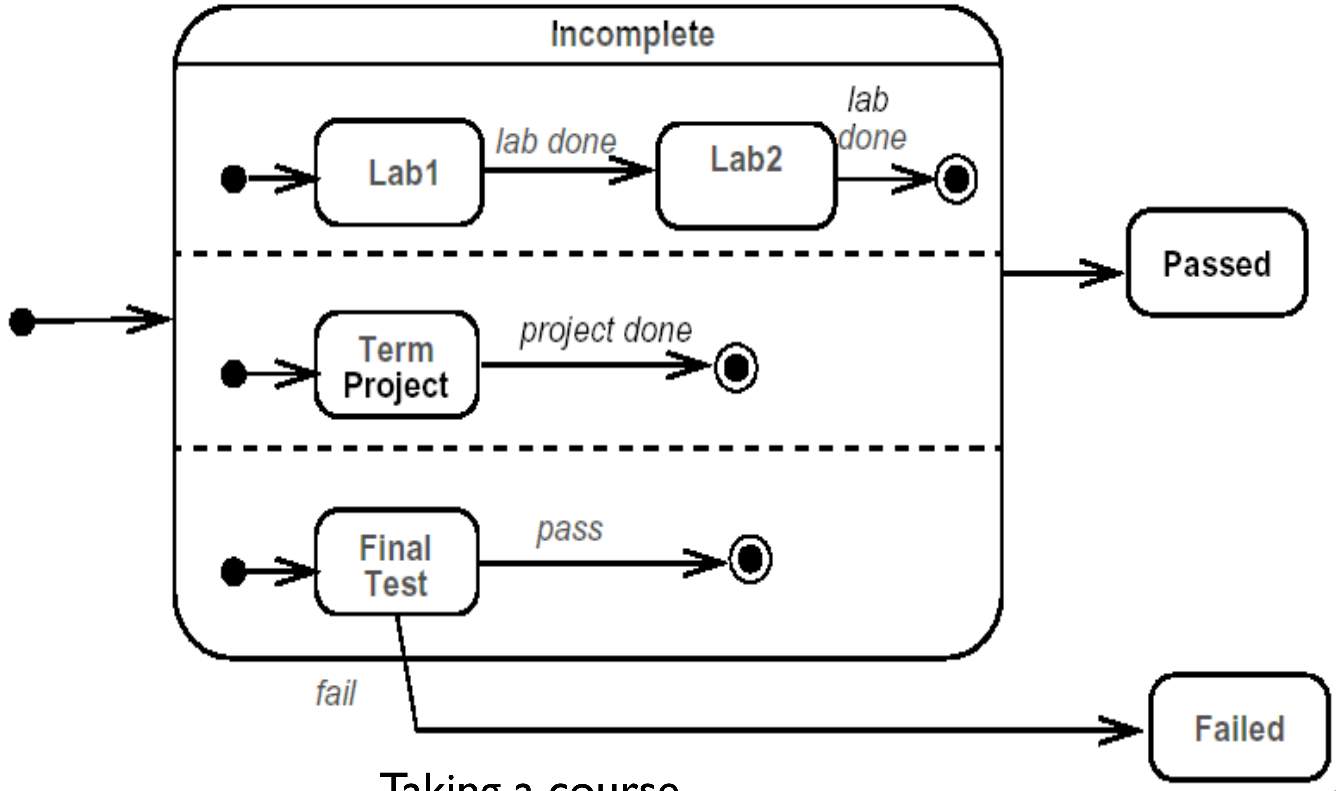
- Remember the last substate that the superstate was active in it prior to the transition from the composite state.



Backup process

Concurrent substates

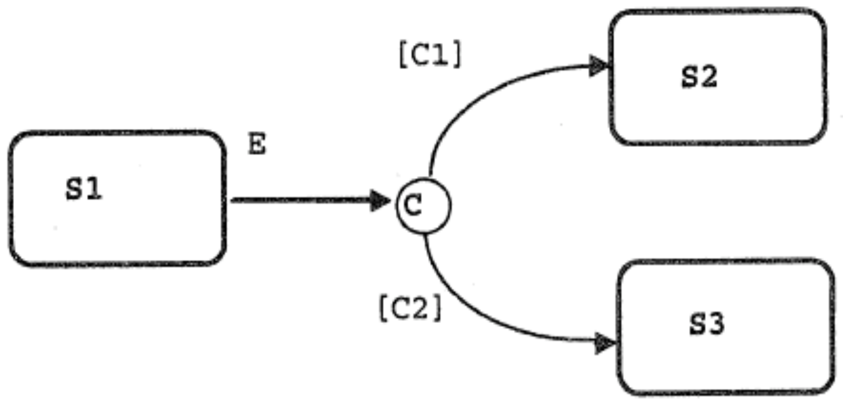
- Substates execute in parallel



Taking a course

Nondeterministic Situations

- If E occurs and both C1 and C2 are true, the system does not know which transition to take.
- The implementing tools make an arbitrary decision.



How to produce statechart diagrams

1. Identify entities that have complex behavior
2. Determine initial and final states of the entity
3. Identify the events that affect the entity
4. Identify entry and exit actions on states
5. Expand states using substates
6. All actions have to be implemented as operations on classes

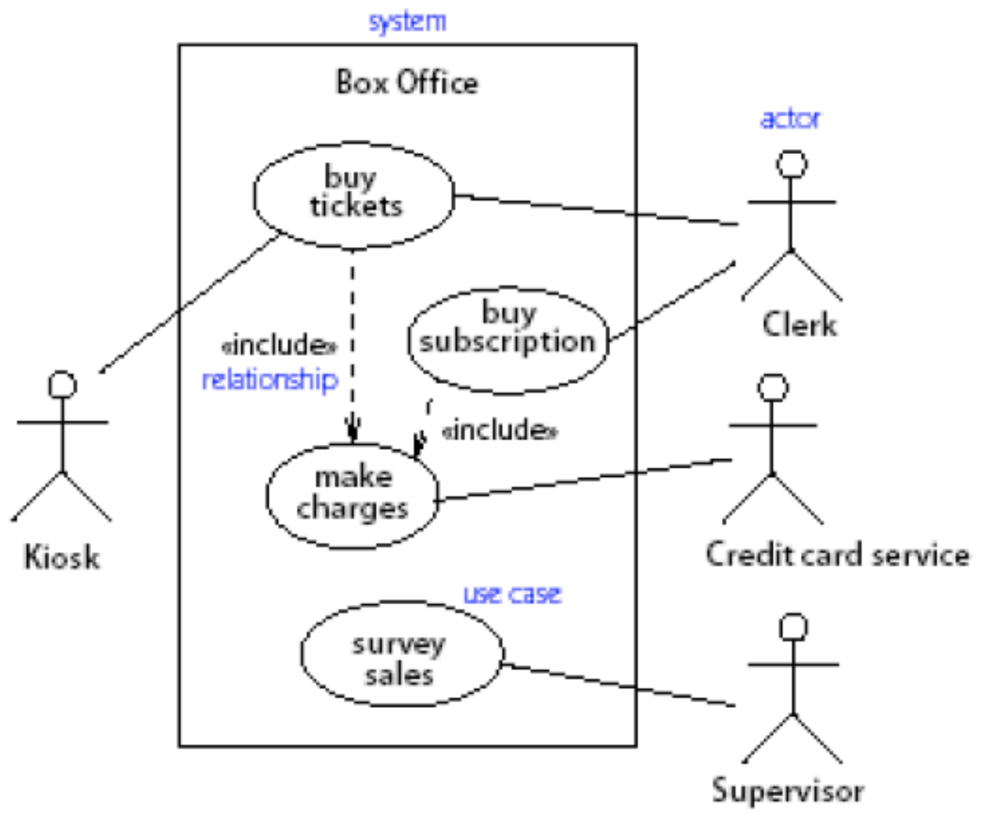


Figure 3-2. *Use case diagram*

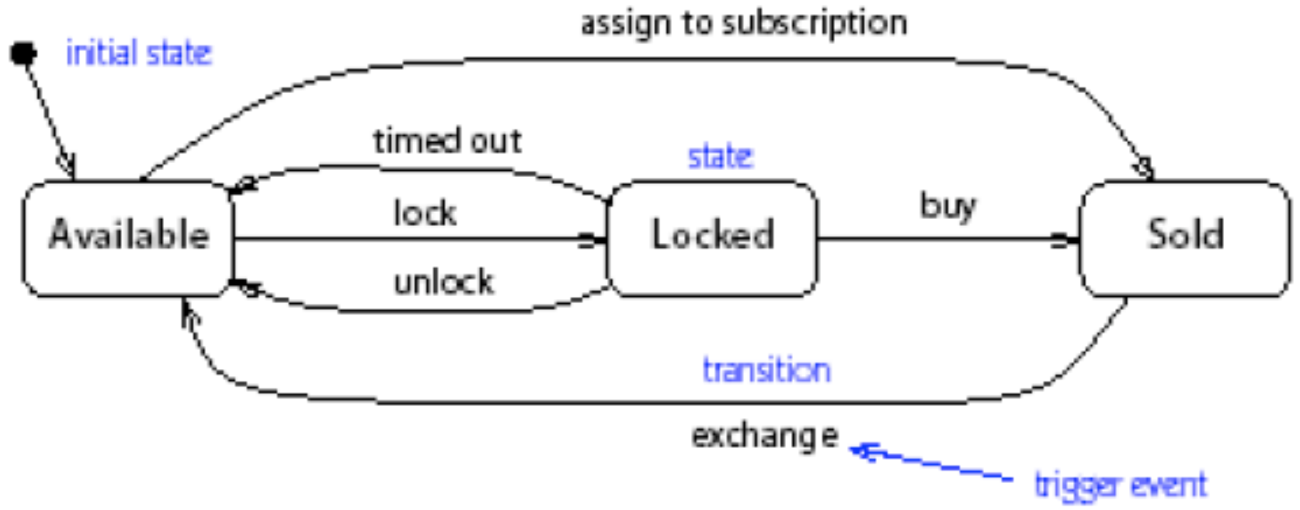


Figure 3-5. *Statechart diagram*

NOTE: UML-2 Notation is not Mandatory

COMPONENT DIAGRAM in UML 2.0

Veronica Carrega

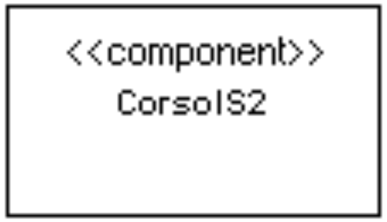
COMPONENT in UML 2.0

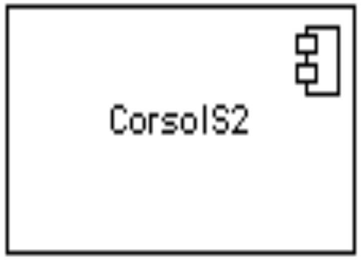
- Modular unit with well-defined interfaces that is replaceable within its environment
- **Autonomous** unit within a system
  - Has one or more provided and required interfaces
  - Its internals are hidden and inaccessible
  - A component is encapsulated
  - Its dependencies are designed such that it can be treated as independently as possible

COMPONENT NOTATION

- A component is shown as a rectangle with
  - A keyword <<component>>
  - Optionally, in the right hand corner a component icon can be displayed
    - A component icon is a rectangle with two smaller rectangles jutting out from the left-hand side
    - This symbol is a visual stereotype
  - The component name
- Components can be labelled with a stereotype

  there are a number of standard stereotypes ex: <<entity>>, <<subsystem>>





Component ELEMENTS

- A component can have
  - Interfaces

    An interface represents a declaration of a set of operations and obligations
  - Usage dependencies

    A usage dependency is relationship which one element requires another element for its full implementation
  - Ports

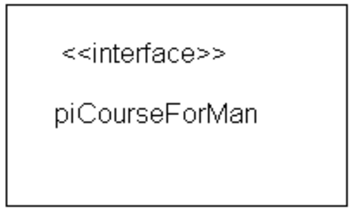
    Port represents an interaction point between a component and its environment
  - Connectors
    - Connect two components
    - Connect the external contract of a component to the internal structure

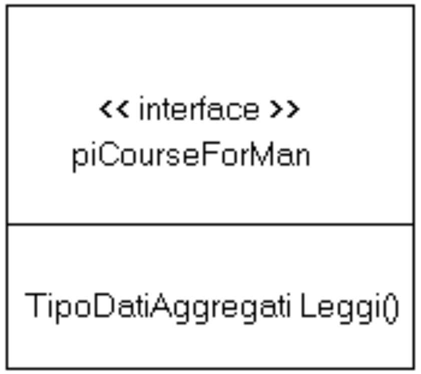
INTERFACE

- A component defines its behaviour in terms of provided and required interfaces
- An interface
  - Is the definition of a collection of one or more operations
  - Provides only the operations but not the implementation
  - Implementation is normally provided by a class/ component
  - In complex systems, the physical implementation is provided by a group of classes rather than a single class

INTERFACE

- May be shown using a rectangle symbol with a keyword <<interface>> preceding the name
- For displaying the full signature, the interface rectangle can be expanded to show details

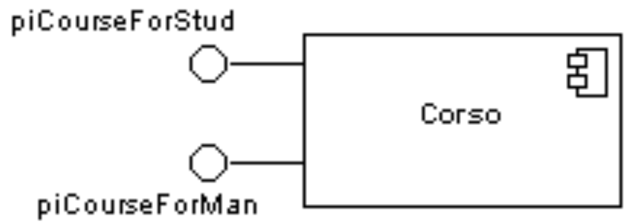


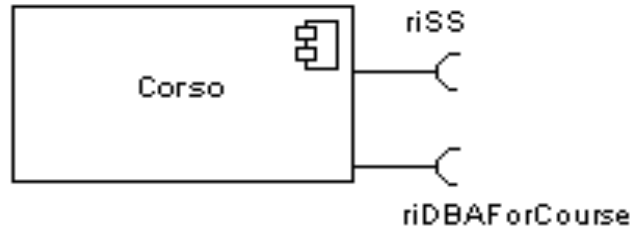


- Can be
  - Provided
  - Required

INTERFACE

- A provided interface
  - Characterize services that the component offers to its environment
  - Is modeled using a ball, labelled with the name, attached by a solid line to the component

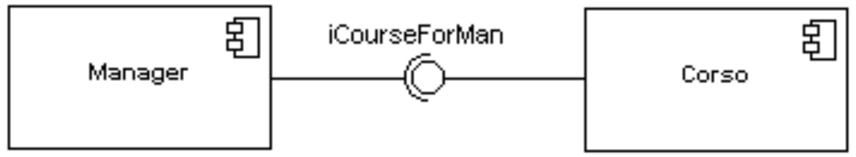




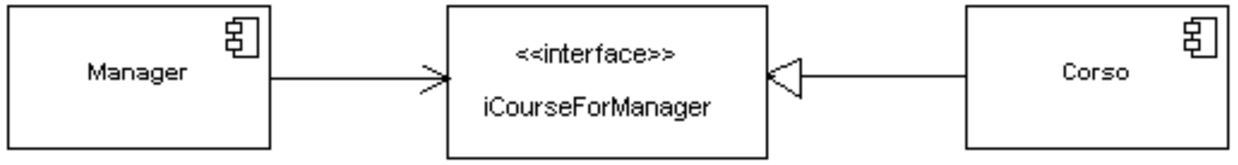
- **A required interface**
  - **Characterize services that the component expects from its environment**
  - **Is modeled using a socket, labelled with the name, attached by a solid line to the component**
  - **In UML 1.x were modeled using a dashed arrow**

INTERFACE

- Where two components/classes provide and require the same interface, these two notations may be combined



- The ball-and-socket notation hint at that interface in question serves to mediate interactions between the two components
- If an interface is shown using the rectangle symbol, we can use an alternative notation, using dependency arrows


Manager
<<interface>>
iCourseForManager
Corso