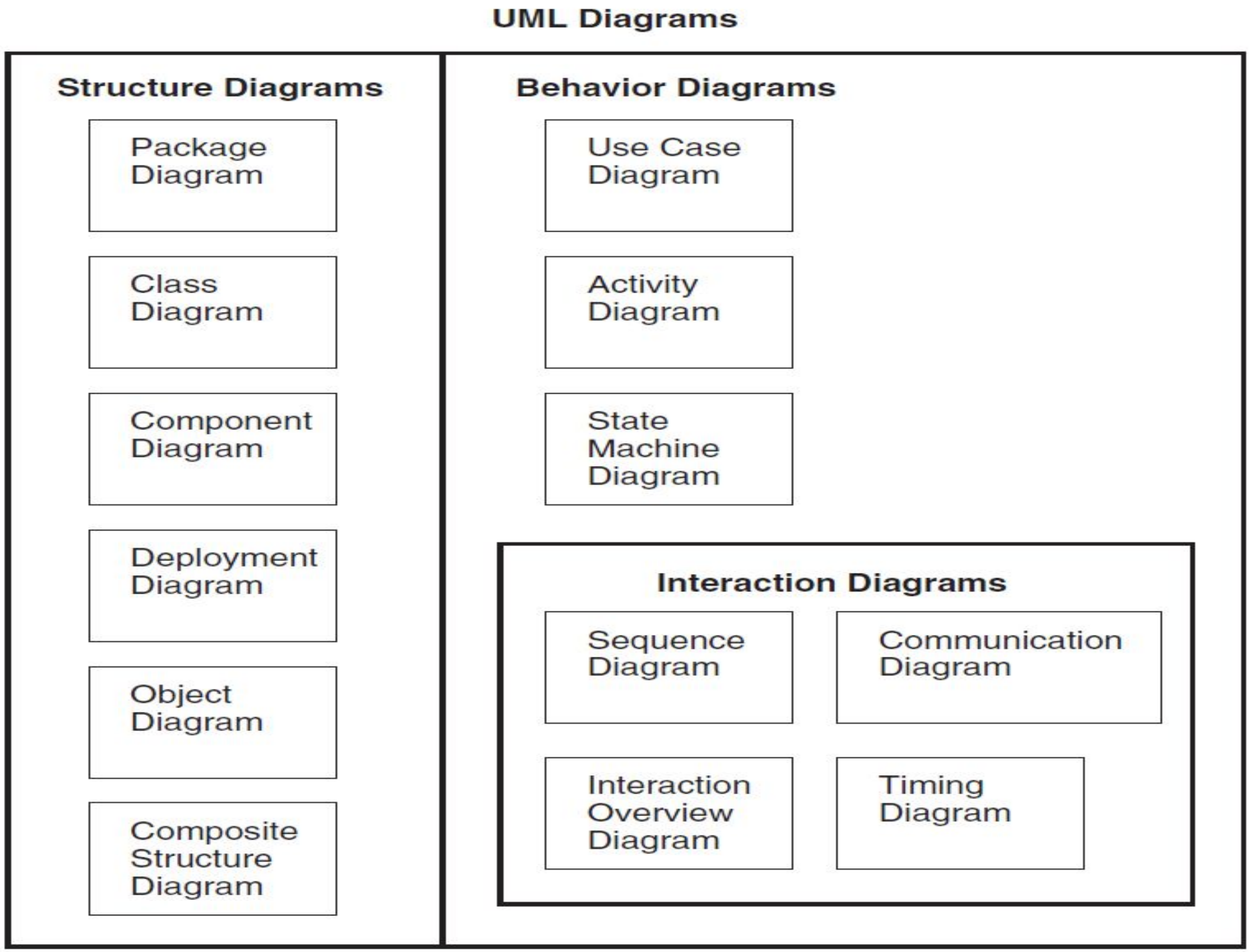
UML Modeling: State Diagram

Instructor: Mehroze Khan

UML Diagrams



State Diagram

- The UML includes state diagram notation to illustrate the **events and states of things—transactions, use cases, people**, and so forth.
- A UML state diagram illustrates the interesting events and states of an object, and the **behavior of an object in reaction to an event**.
- A state diagram shows the lifecycle of an object:
  - **What events it experiences**
  - **Its transitions**
  - **The states it is in between these events**.
- It is also known as **Statechart Diagram** or **State Machine Diagram**.

Events, States and Transitions

- An **event** is a significant or noteworthy occurrence.
- **For example:**

  A telephone receiver is taken off the hook.

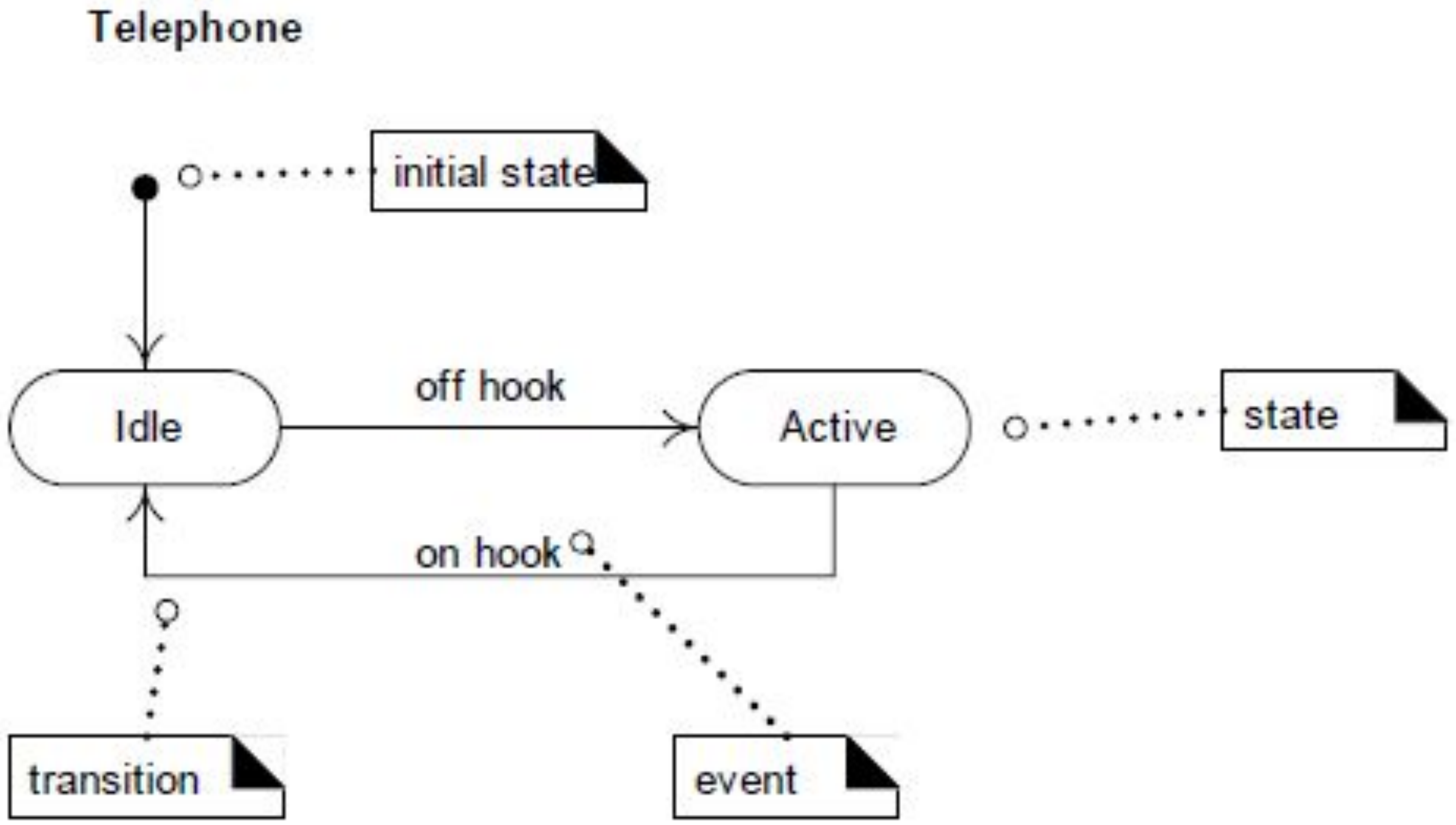
- A **state** is the condition of an object at a moment in time—the time between events.
- **For example:**

A telephone is in the state of being "idle" after the receiver is placed on the hook and until it is taken off the hook.

- A **transition** is a relationship between two states that indicates that when an event occurs, the object moves from the prior state to the subsequent state.
- **For example:**

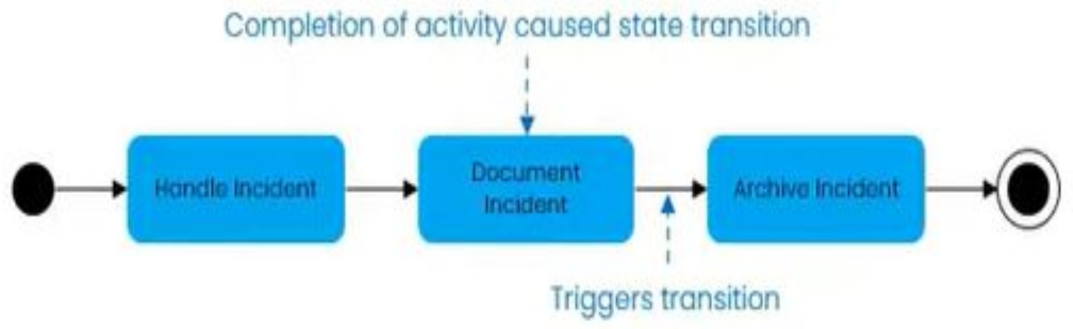
When the event "off hook" occurs, transition the telephone from the "idle" to "active" state.

State Diagram for a Telephone



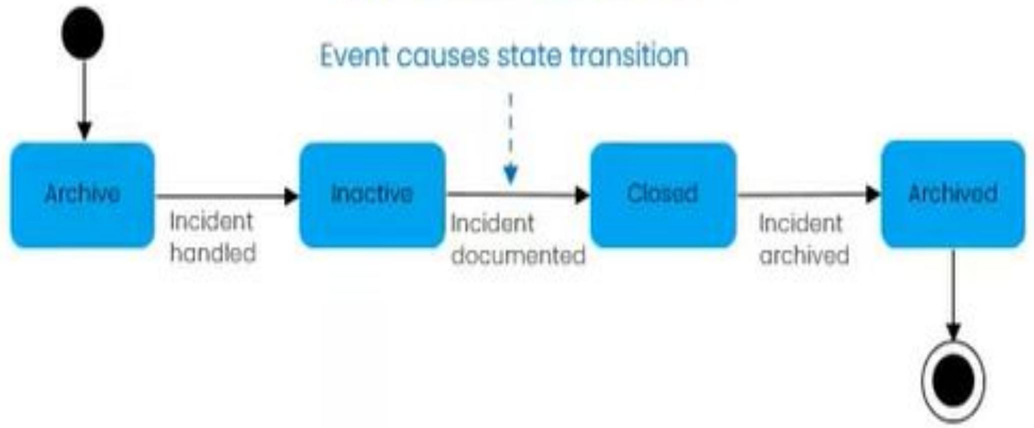
Activity vs State Diagram

Activity Diagrams



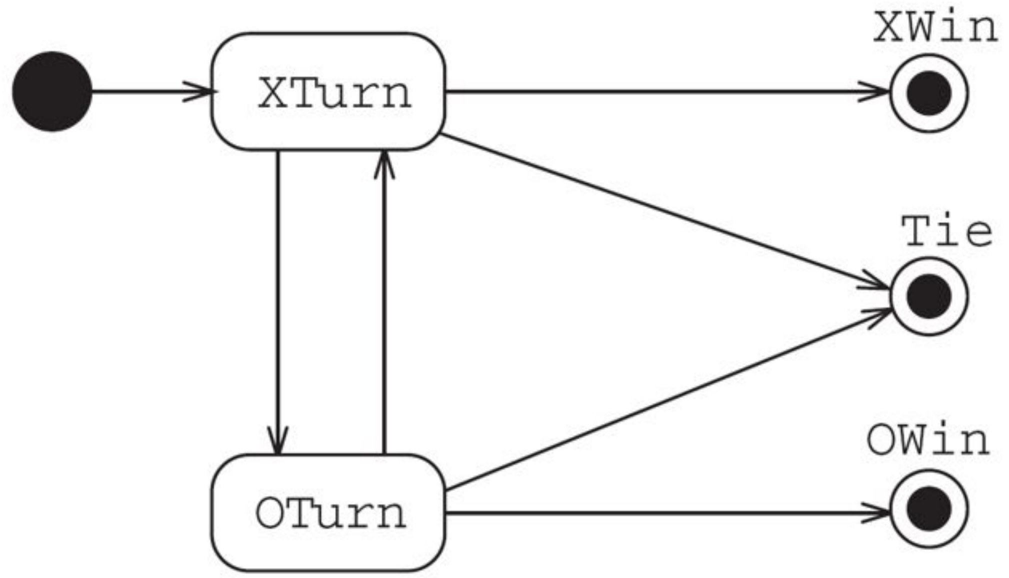
- Capture high-level activities aspects
- Possible to represent concurrency and coordination between activities

State Machines



- Additional notations capture how activities are coordinated
- Shows the possible states of the object and the transitions that cause a change in state

Tic-Tac-Toe Game



Event

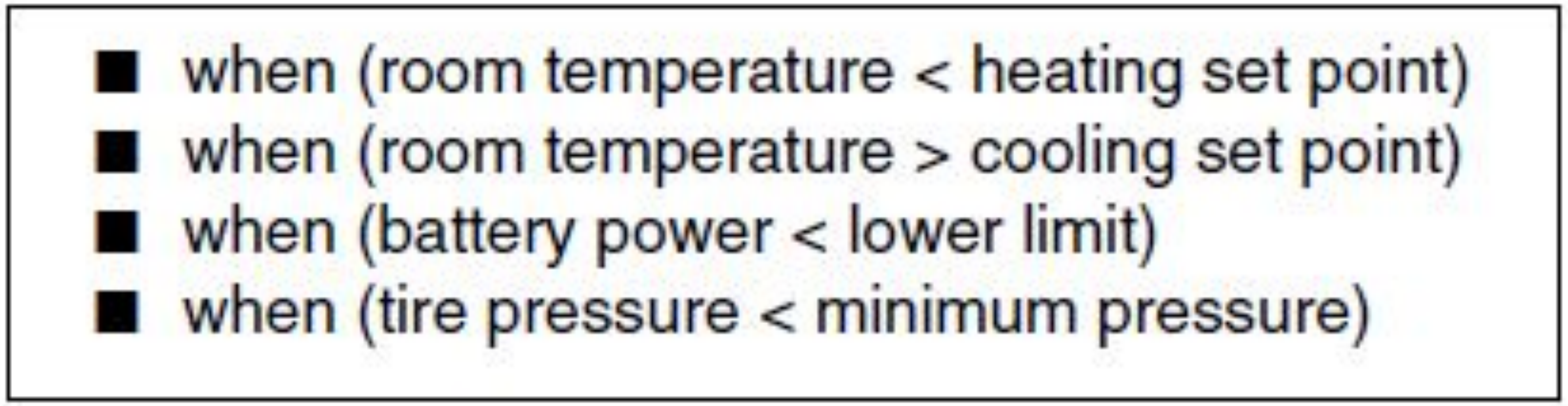
- An ***event*** is an occurrence of a stimulus that can trigger a state transition.
- Such as ***user presses left button*** or ***flight 123 departs from Chicago***.
- Events often correspond to **verbs in the past tense** (*power turned on*, *alarm set*) or to the **onset of some condition** (***paper tray becomes empty***, ***temperature becomes lower than freezing***).
- Events include **error conditions** as well as **normal occurrences**.
- **For example**, *motor jammed, transaction aborted,* and *timeout* are typical error events.
- There are several kinds of events. The most common are the **signal event, the change event, and the time event.**

Event

- One event may **logically precede** or follow another, or the two events may be **unrelated**.
- Flight 123 must depart Chicago before it can arrive in San Francisco; the two events are **causally related**.
- Flight 123 may depart before or after flight 456 departs Rome; the two events are **causally unrelated**.
- Two events that are causally unrelated are said to be ***concurrent****;* they have no effect on each other.

Change Event

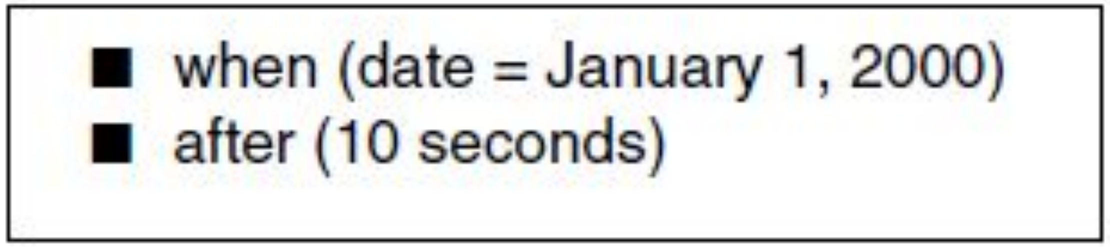
- A ***change event*** is an event that is caused by the satisfaction of a Boolean expression.
- The intent of a change event is that the expression is **continually tested**—whenever the expression changes from false to true, the event happens.
- The UML notation for a **change event** is the keyword ***when*** followed by a ***parenthesized Boolean expression***



- when (room temperature < heating set point)
- when (room temperature > cooling set point)
- when (battery power < lower limit)
- when (tire pressure < minimum pressure)

Time Event

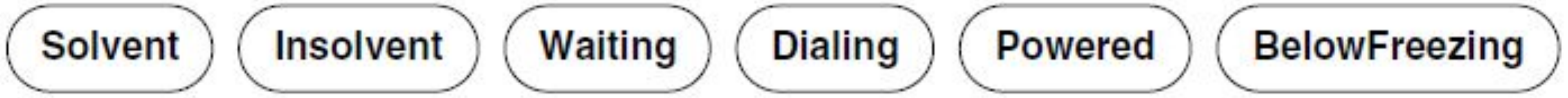
- A *time event* is an event caused by the occurrence of an **absolute time** or the **elapse of a time interval**.
- UML notation for an **absolute time** is the keyword ***when*** followed by a ***parenthesized expression involving time***.
- The notation for a **time interval** is the keyword ***after*** followed by a ***parenthesized expression that evaluates to a time duration.***



- when (date = January 1, 2000)
- after (10 seconds)

State

- A ***state*** is an abstraction of the values and links of an object. Sets of values and links are grouped together into a state according to the gross behavior of objects.
- **For example**, A light bulb has two possible states: On and Off. When you press a switch, the light bulb changes state from Off to On or from On to Off.
- States often correspond to **verbs** with a suffix of "ing" (*Waiting*, *Dialing*) or the duration of some **condition** (*Powered*, *BelowFreezing*).
- UML notation for a state—**a rounded box containing state name**



Transitions

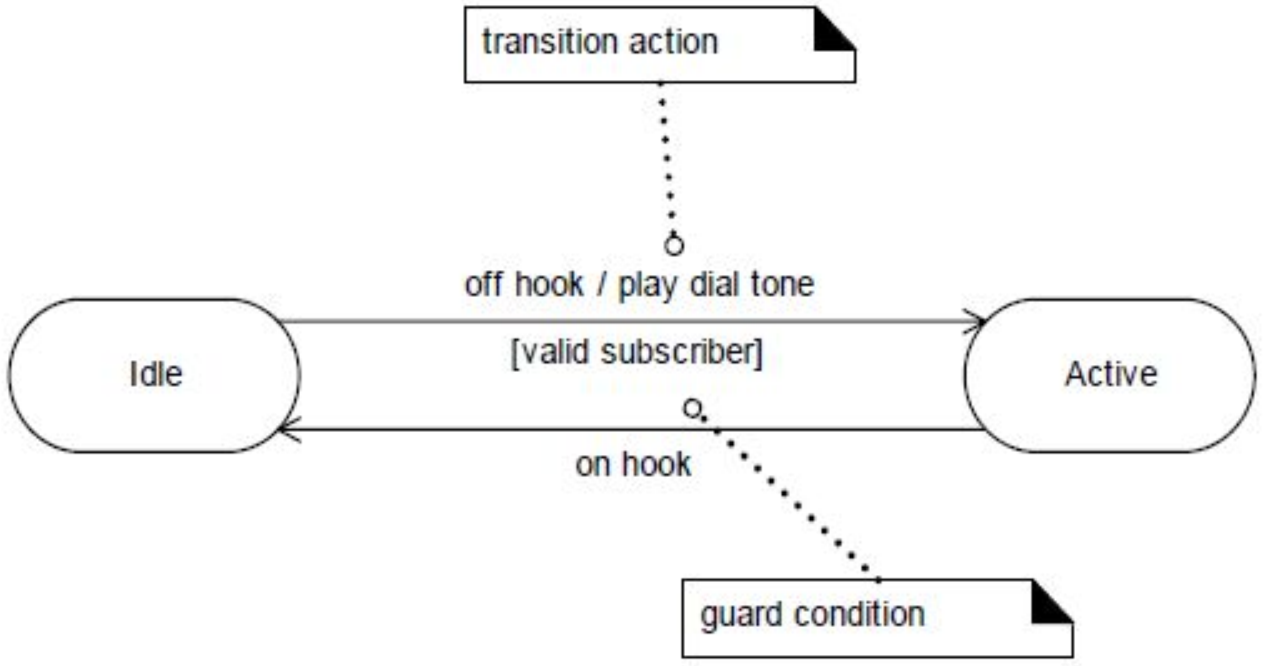
- A ***transition*** is an instantaneous change from one state to another.
- **For example**, when a called phone is answered, the phone line transitions from the *Ringing* state to the *Connected* state.
- The transition is said to ***fire*** upon the change from the source state to the target state.
- The origin and target of a transition usually are different states but may be the same.
- A transition fires when its event occurs (unless an **optional guard condition** causes the event to be ignored).

Transitions

- A ***guard condition*** is a Boolean expression that must be true in order for a transition to occur.
- **For example**, a traffic light at an intersection may change only if a road has cars waiting.
- A **guarded transition** fires when its event occurs, but only if the guard condition is true.
- A guard condition is checked **only once**, at the time the event occurs, and the transition fires if the condition is true. If the condition becomes true later, the transition does not then fire.
- **Example: Door Lock**
- **Event**: You try to open the door (handle is turned).
- **Guard Condition**: The door is unlocked.
- **Transition**: If the door is unlocked (condition), the door opens (next state). If the door is locked, the transition does not occur, and the door stays closed.

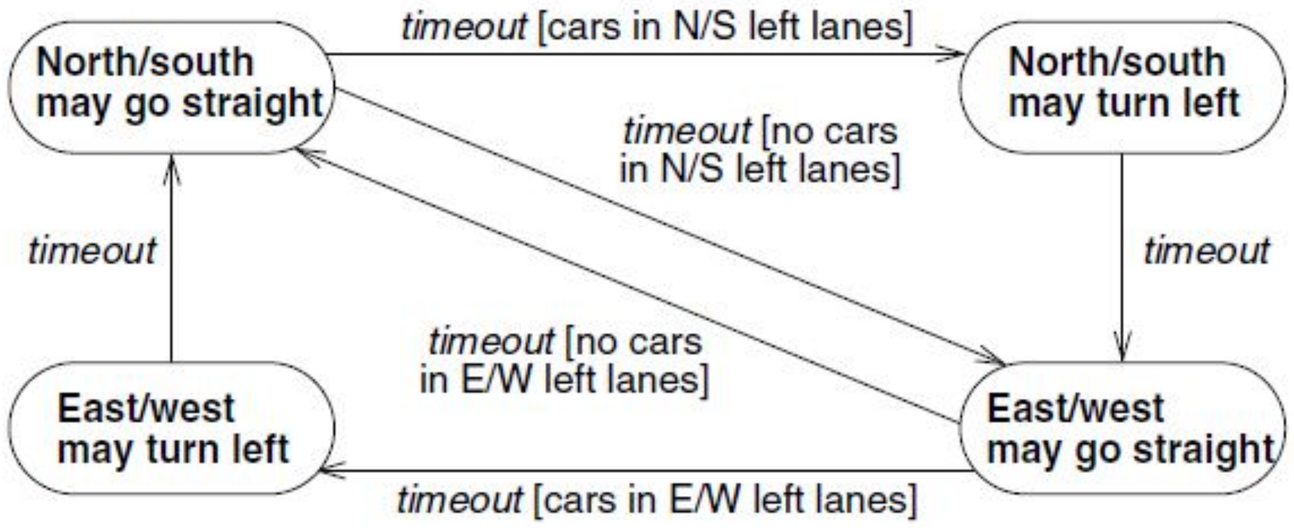
Transition Actions and Guards

- A transition can cause an action to fire.
- A transition may also have a conditional guard—or boolean test. The transition only occurs if the test passes.



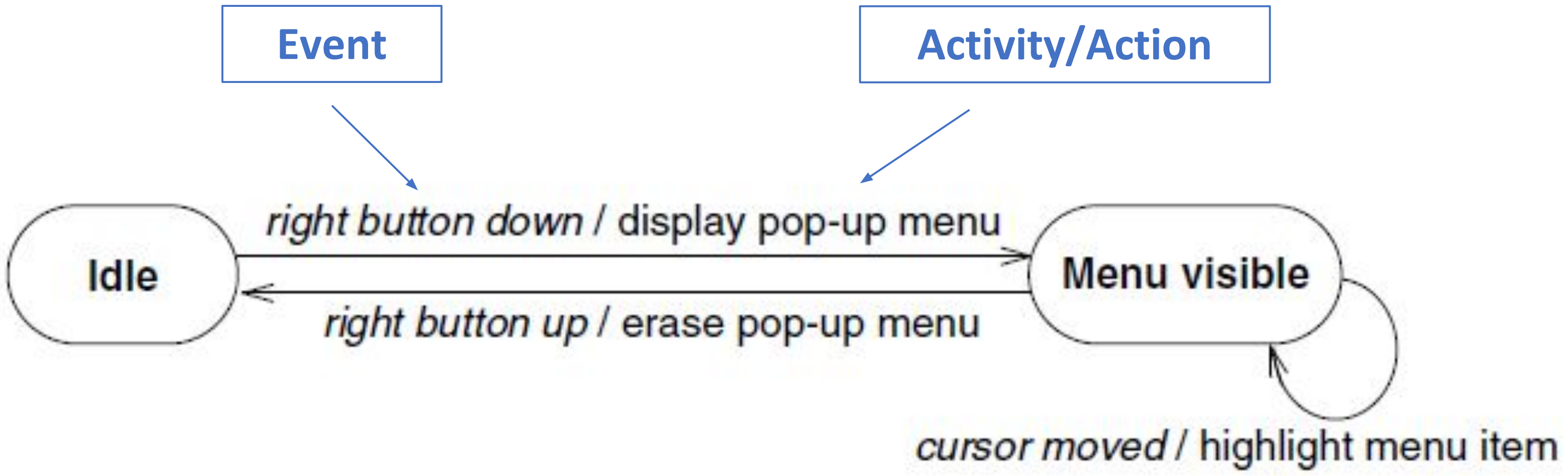
Guarded Transitions for Traffic Lights at an intersection

- One pair of electric eyes checks the north-south left turn lanes; another pair checks the east-west turn lanes.
- If no car is in the north-south and/or east-west turn lanes, then the traffic light control logic is smart enough to skip the left turn portion of the cycle.


North/south may go straight
timeout [cars in N/S left lanes]
North/south may turn left
timeout [no cars in N/S left lanes]
timeout
timeout
timeout [no cars in E/W left lanes]
East/west may turn left
East/west may go straight
timeout [cars in E/W left lanes]

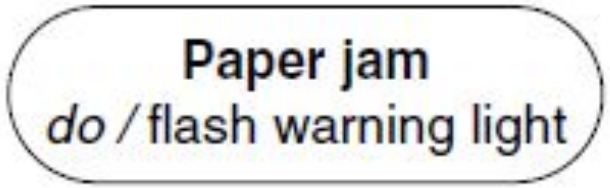
Activity Effects

- An ***effect*** is a reference to a behavior that is executed in response to an event. An ***activity*** is the actual behavior that can be invoked by any number of effects.
- The notation for an activity is a slash ("/") and the name (or description) of the activity, following the event that causes it.
- State diagram for a **pop-up menu** on a workstation. When the right button is pressed, the menu is displayed; when the right button is released, the menu is erased. While the menu is visible, the highlighted menu item is updated whenever the cursor moves.


Event
Activity/Action
right button down / display pop-up menu
Idle
Menu visible
right button up / erase pop-up menu
cursor moved / highlight menu item

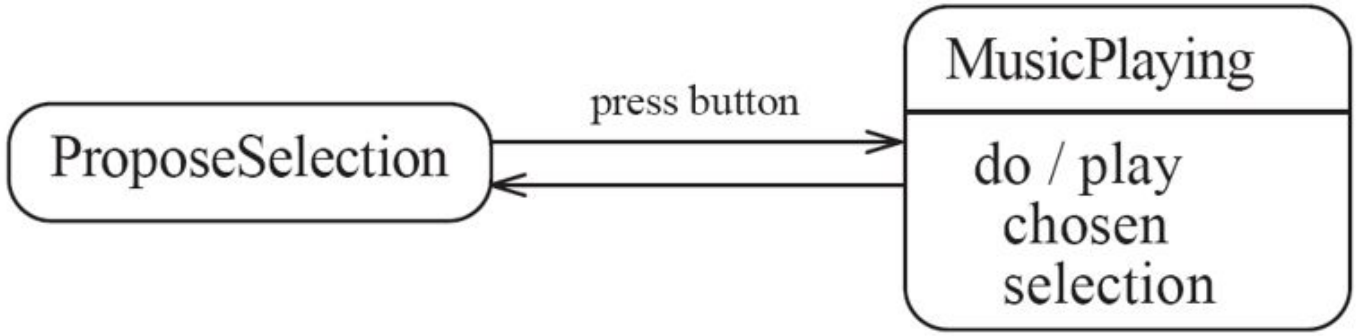
Do-Activities

- A ***do-activity*** is an activity that continues for an **extended time**.
- The keyword *do* is reserved for indicating an ongoing activity
- By definition, a do-activity **can only occur within a state** and cannot be attached to a transition.
- For example, the warning light may flash during the *Paper jam* state for a copy machine

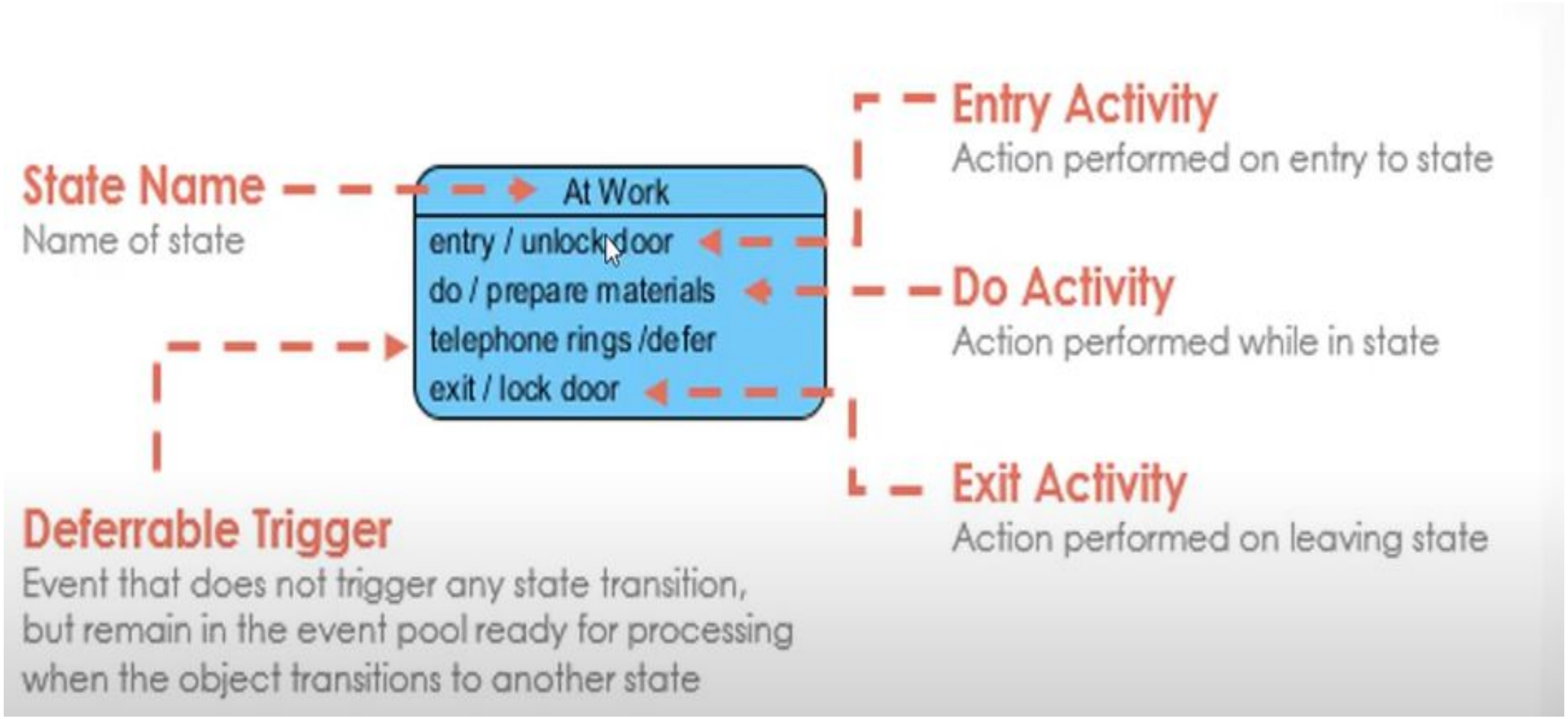


Activities in State Diagram

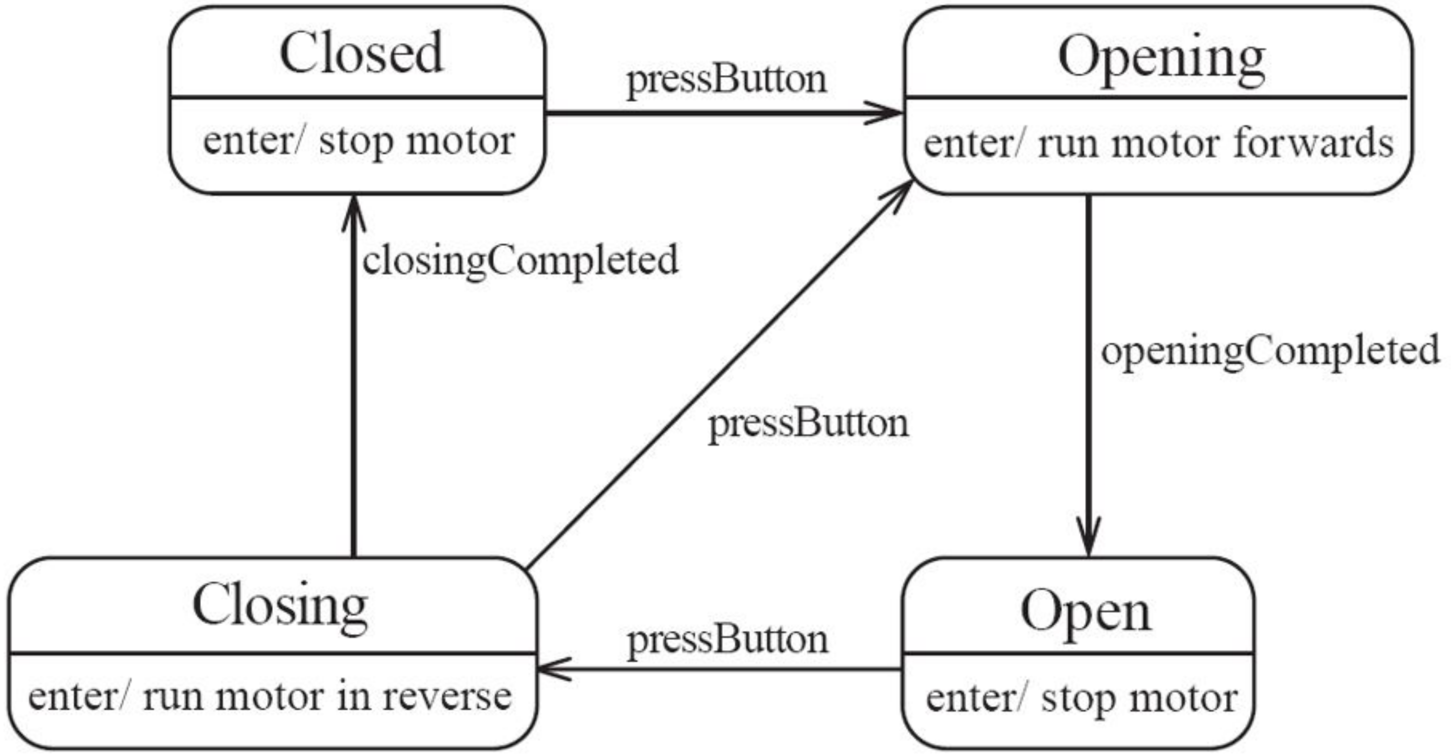
- An *activity* is something that takes place while the system is in a state.
  - It takes a period of time.
  - The system may take a transition out of the state in response to completion of the activity,
  - Some other outgoing transition may result in:
    - The interruption of the activity, and
    - An early exit from the state.



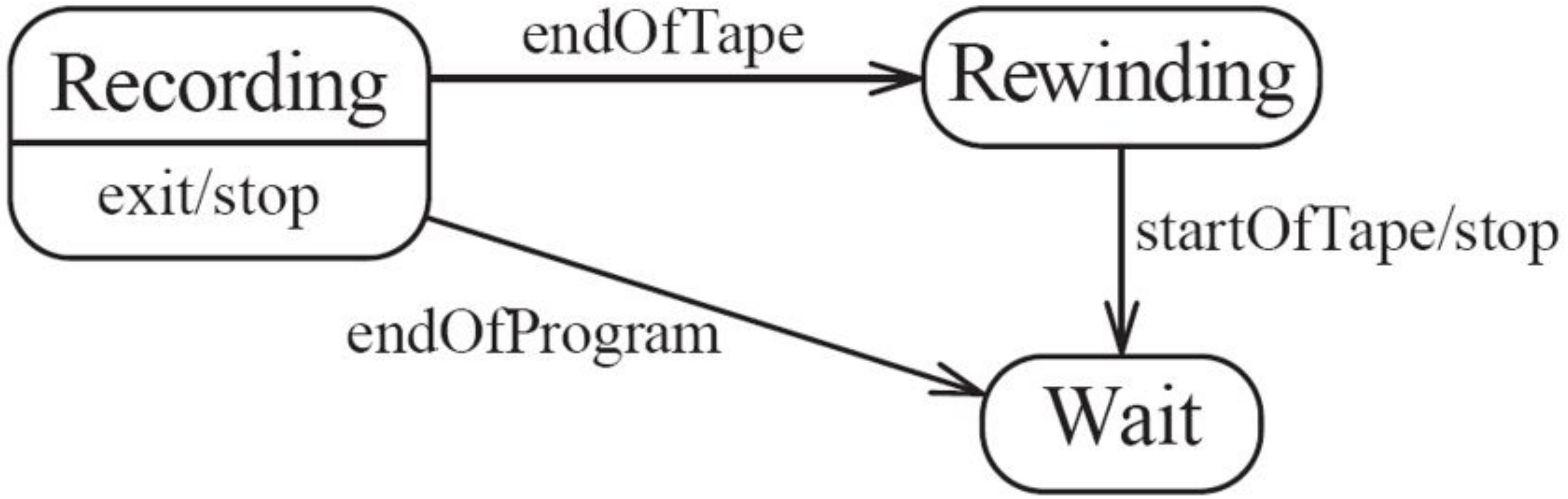
Activities


State Name
Name of state
At Work
entry / unlock door
do / prepare materials
telephone rings /defer
exit / lock door
Entry Activity
Action performed on entry to state
Do Activity
Action performed while in state
Exit Activity
Action performed on leaving state
Deferrable Trigger
Event that does not trigger any state transition, but remain in the event pool ready for processing when the object transitions to another state

Enter Activity



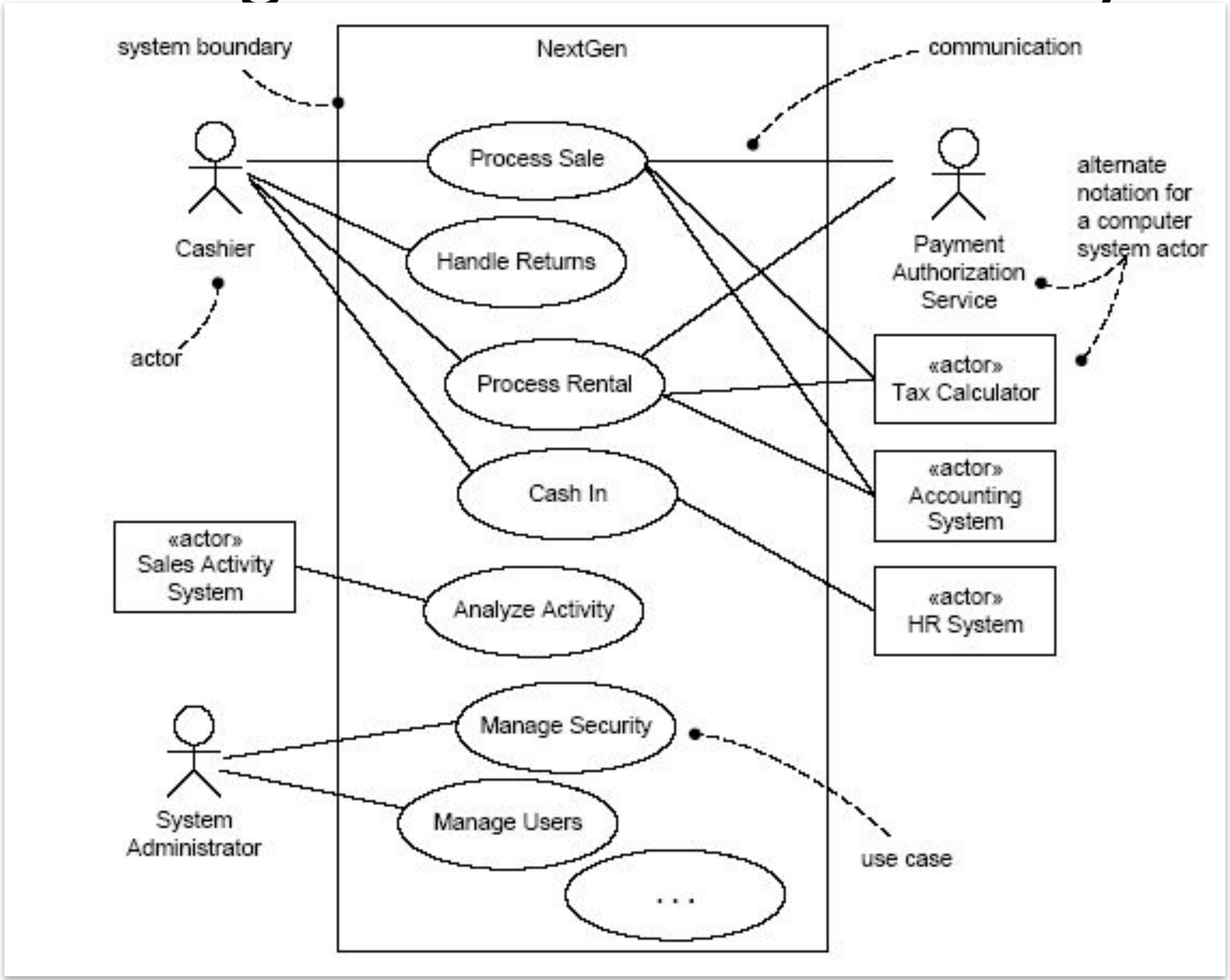
Exit Activity



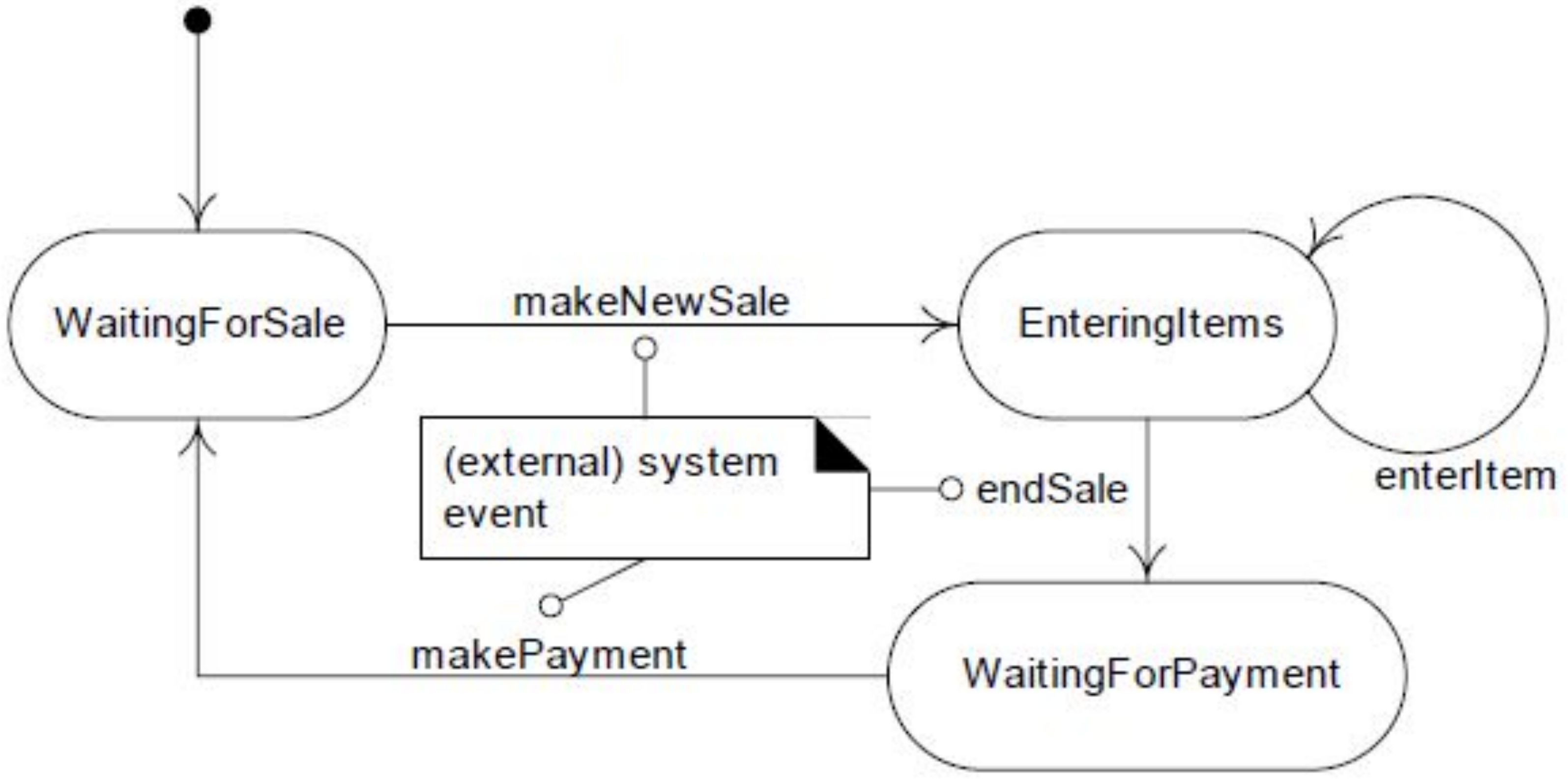
Subject of State Diagram

- A statechart diagram may be applied to a variety of UML elements, including:
- Classes (conceptual or software)
- Use cases
- Since an entire "system" may be represented by a class, it too may have its own statechart diagram.

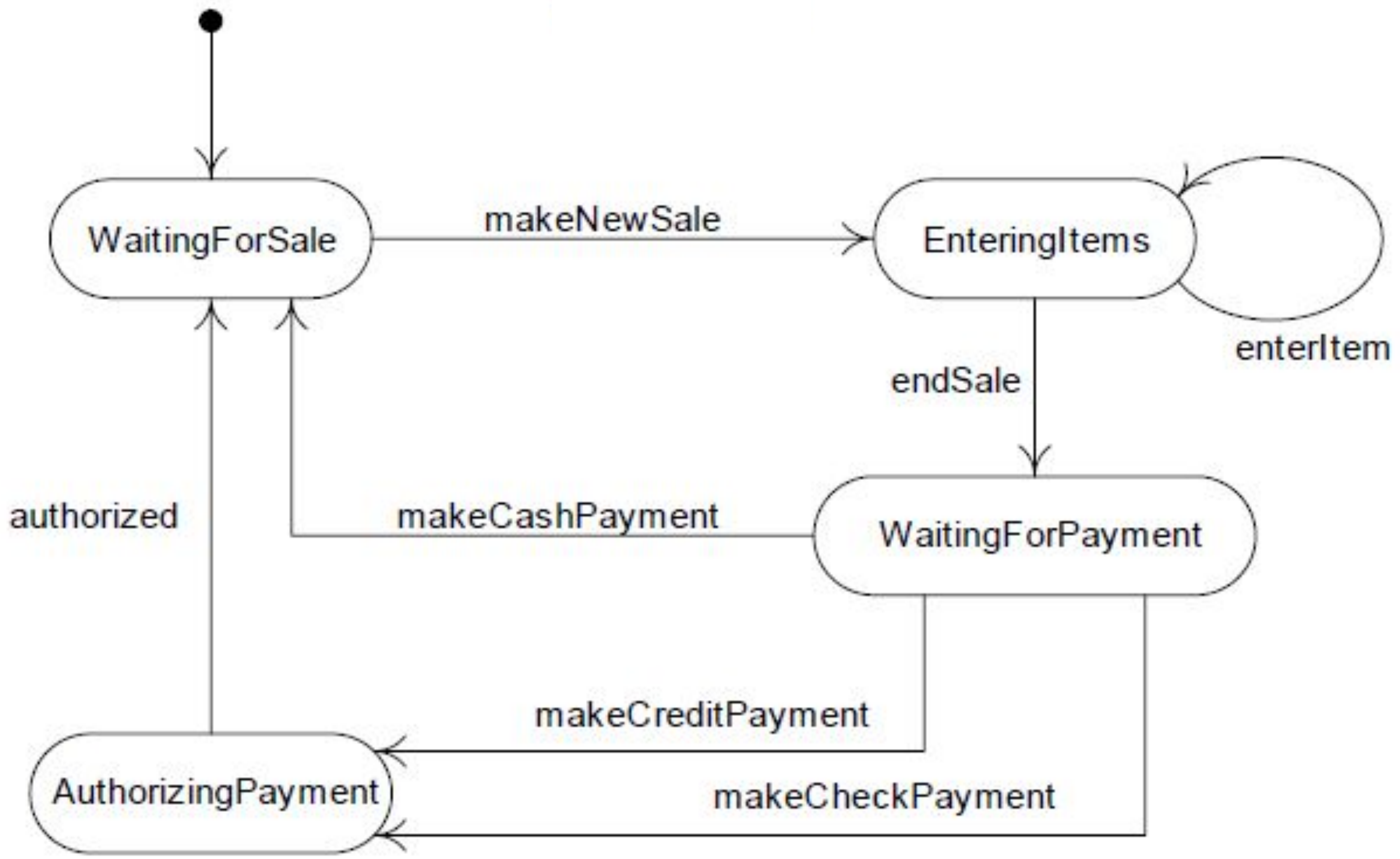
Use Case Diagram of NexGen POS System



State Diagram for Process Sale



State Diagram for Process Sale

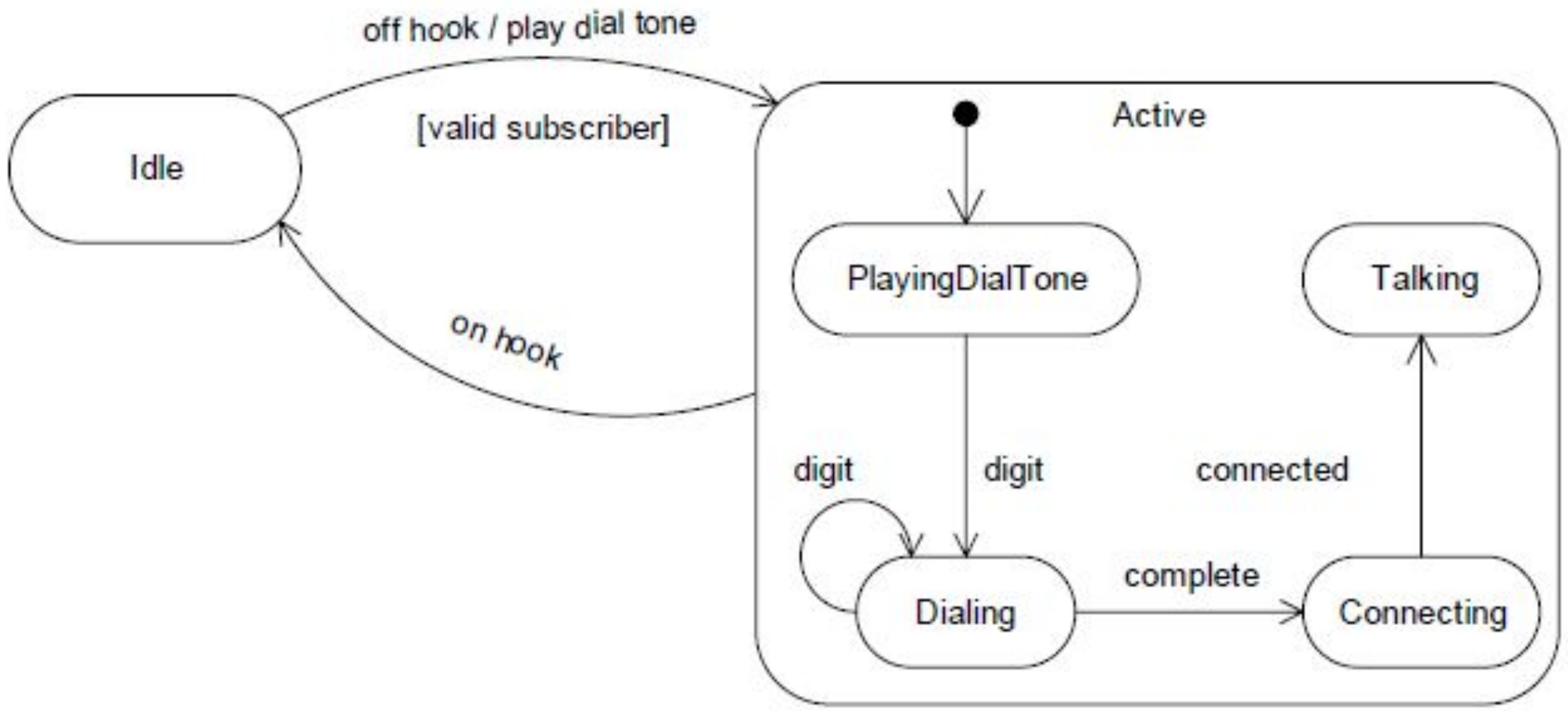


Event Types

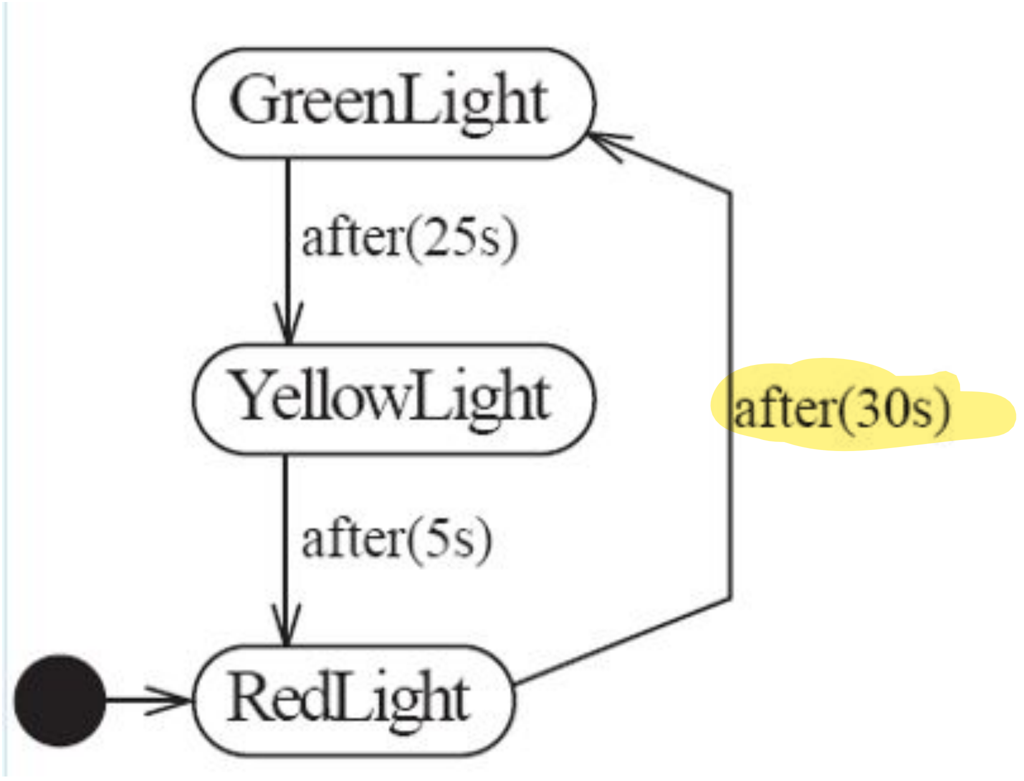
- It is useful to categorize events as follows:
- **External event**—Also known as a system event, is caused by something (for example, an actor) outside our system boundary. SSDs illustrate external events. Noteworthy external events precipitate the invocation of system operations to respond to them.
  - When a cashier presses the "enter item" button on a POS terminal, an external event has occurred.
- **Internal event**—Caused by something inside our system boundary. In terms of software, an internal event arises when a method is invoked via a message or signal that was sent from another internal object. Messages in interaction diagrams suggest internal events.
  - When *a Sale* receives a *makeLineItem* message, an internal event has occurred.
- **Temporal event**—Caused by the occurrence of a specific date and time or passage of time. In terms of software, a temporal event is driven by a real-time or simulated-time clock.
  - Suppose that after an *endSale* operation occurs, a *makePayment* operation must occur within five minutes, otherwise the current sale is automatically purged.

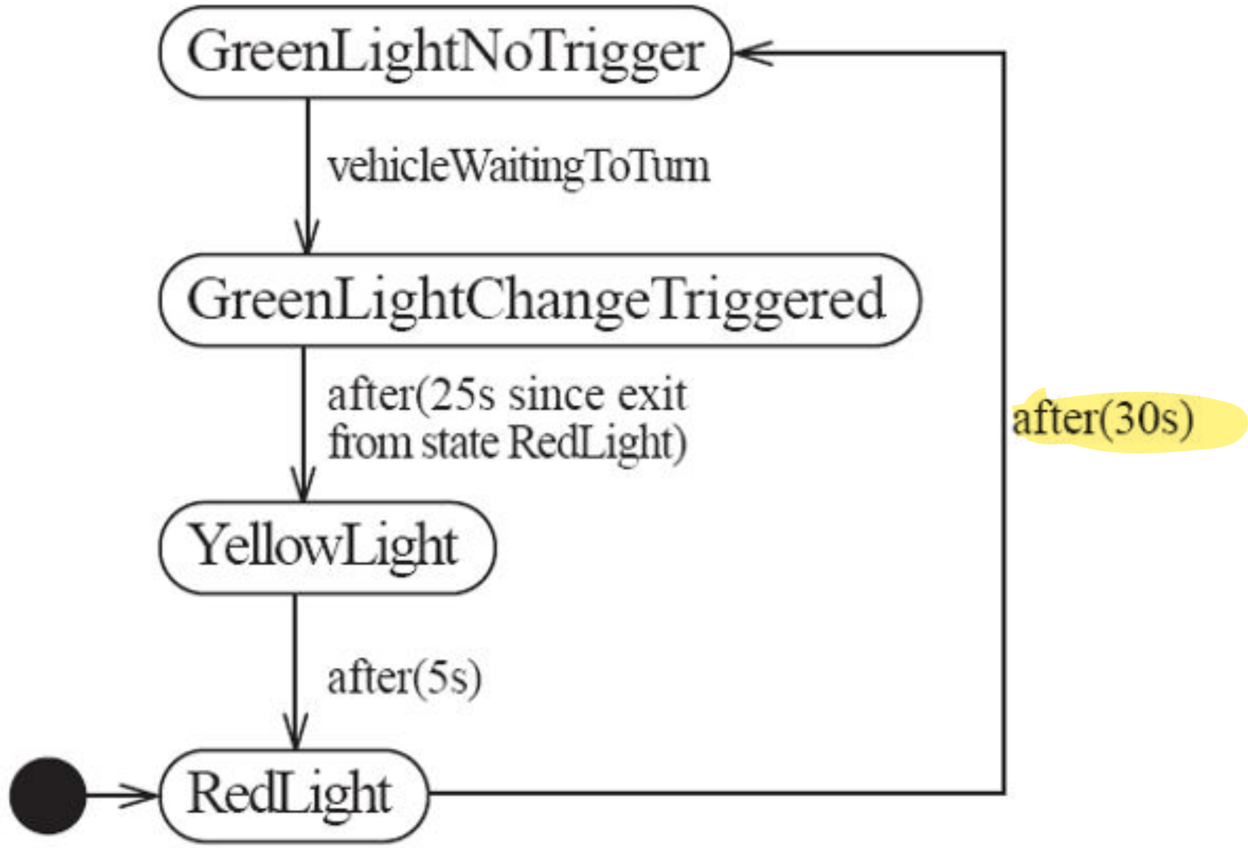
Nested States

- A state allows nesting to contain substates; a substate inherits the transitions of its superstate (the enclosing state)



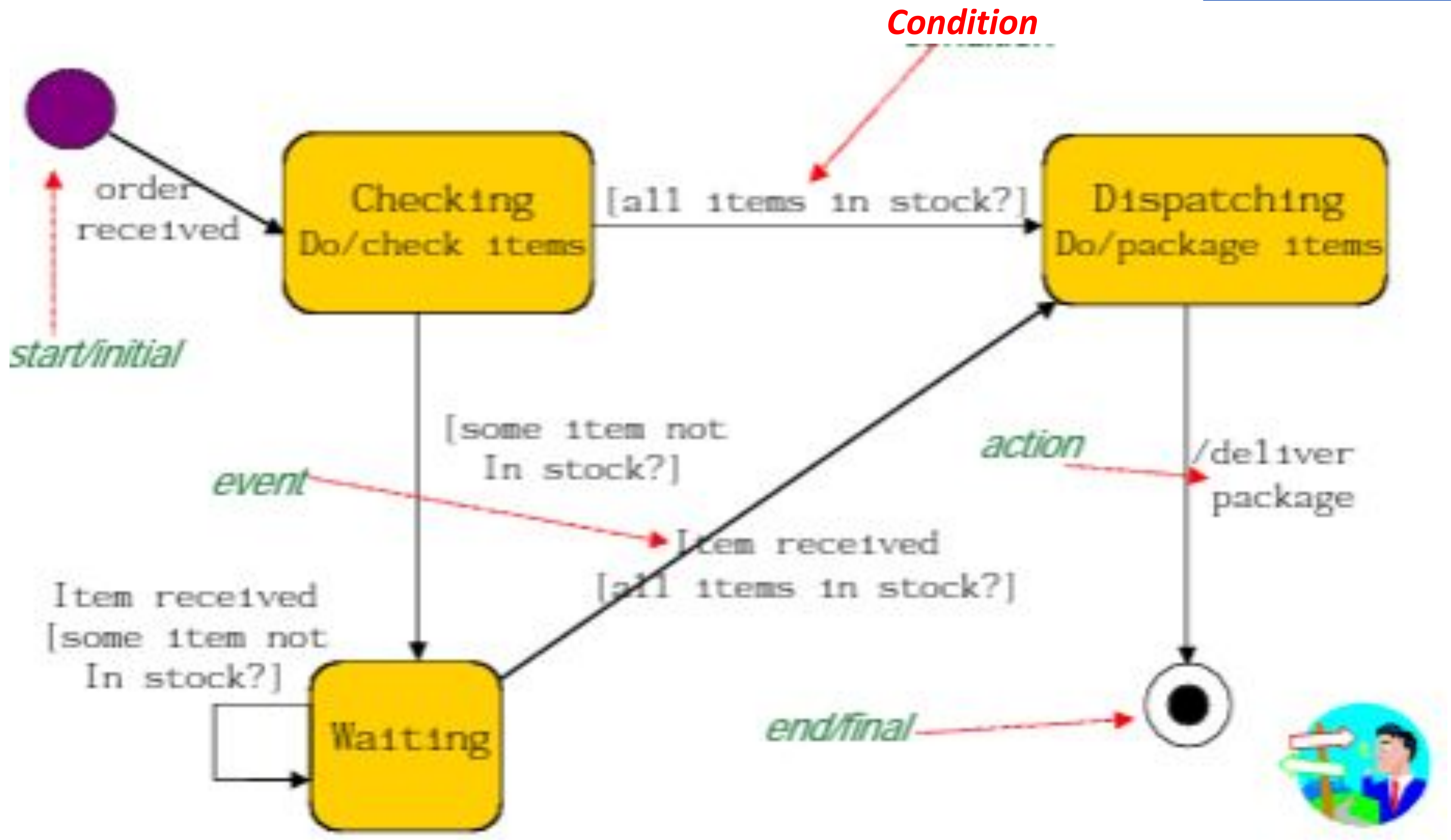
Example of Transitions with Time-Out and Conditions





State Diagram for Purchase Order

Event[Condition]/Activity or Action



References

- Larman, Craig. Applying UML and patterns: an introduction to object oriented analysis and design and interative development. Pearson Education India, 2012.

- Object-Oriented Analysis and Design with Applications, Grady Booch et al., 3rd Edition, Pearson, 2007.

- Timothy C. Lethbridge, Robert Laganaiere, Object-Oriented Software Engineering (2nd Edition), McGraw Hill, 2005

- Object-Oriented Modeling and Design with UML, Michael R. Blaha and James R. Rumbaugh, 2nd Edition, Pearson, 2005.