

UML Behavioral Diagrams

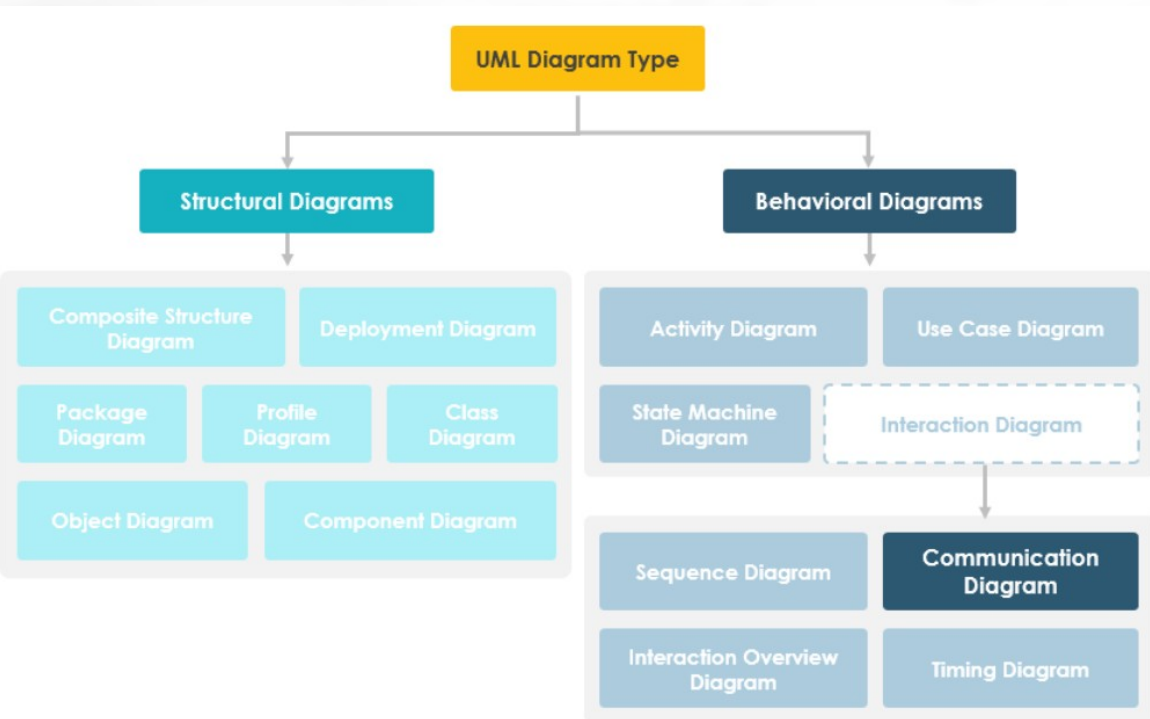
Behavioral models describe the internal behavior of a system.

Behavioral model types:

1. Representations of the details of a business process identified by use-cases
 - Interaction diagrams (Sequence & Communication)
Shows how objects collaborate to provide the functionality defined in the use cases.
2. Representations of changes in the data
 - Behavioral state machines

Focus (for now) is on the dynamic view of the system, not on how it is implemented

Communication Diagram



- A kind of interaction diagram, shows how objects interact.
- An extension of object diagram that shows the objects along with the messages that travel from one to another.
- Shows the messages the objects send each other.

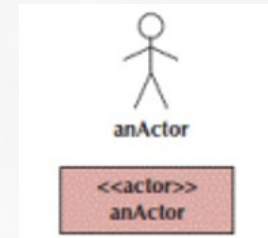
Purpose of Communication Diagram

- Model message passing between objects or roles that deliver the functionalities of use cases and operations
- Model mechanisms within the architectural design of the system
- Model alternative scenarios within use cases or operations that involve the collaboration of different objects and interactions
- Support the identification of objects (hence classes), and their attributes (parameters of message) and operations (messages) that participate in use cases

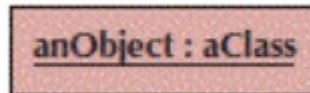
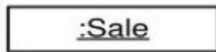
Communication Diagram Elements

1. Actor

- external to the system
- Participates in the collaboration by sending and/or receiving messages



2. Object



- Instances of a class
- Supplier objects are the objects that supply the method that is being called, and therefore receive the message.
- Client objects call methods on supplier objects, and therefore send messages.

3. Links



- The connecting lines drawn between objects in a communication diagram are links.
- Each link represents a relationship between objects and symbolizes the ability of objects to send messages to each other.
- If an object sends messages to itself, the link carrying these messages is represented as a loop icon.

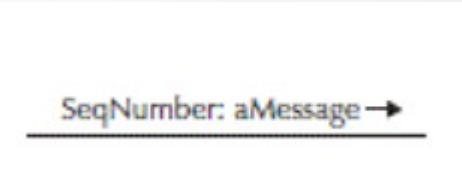
Communication Diagram Elements

4. Messages

Shown as arrows pointing from the Client object to the Supplier object.

Represent a client invoking an operation on a supplier object. They can be modeled along with the objects in the following manner:

- Message icons have one or more messages associated with them.
- Messages are composed of message text prefixed by a sequence number.
- This sequence number indicates the time-ordering of the message.
- The first message in a communication diagram is always numbered 1, the second is 2, and so on.
- You can indicate that a message is nested under a parent message by adding a decimal point and incremental digits to the parent's sequence number.

A diagram showing a horizontal arrow pointing to the right. Above the arrow is the text "SeqNumber: aMessage".

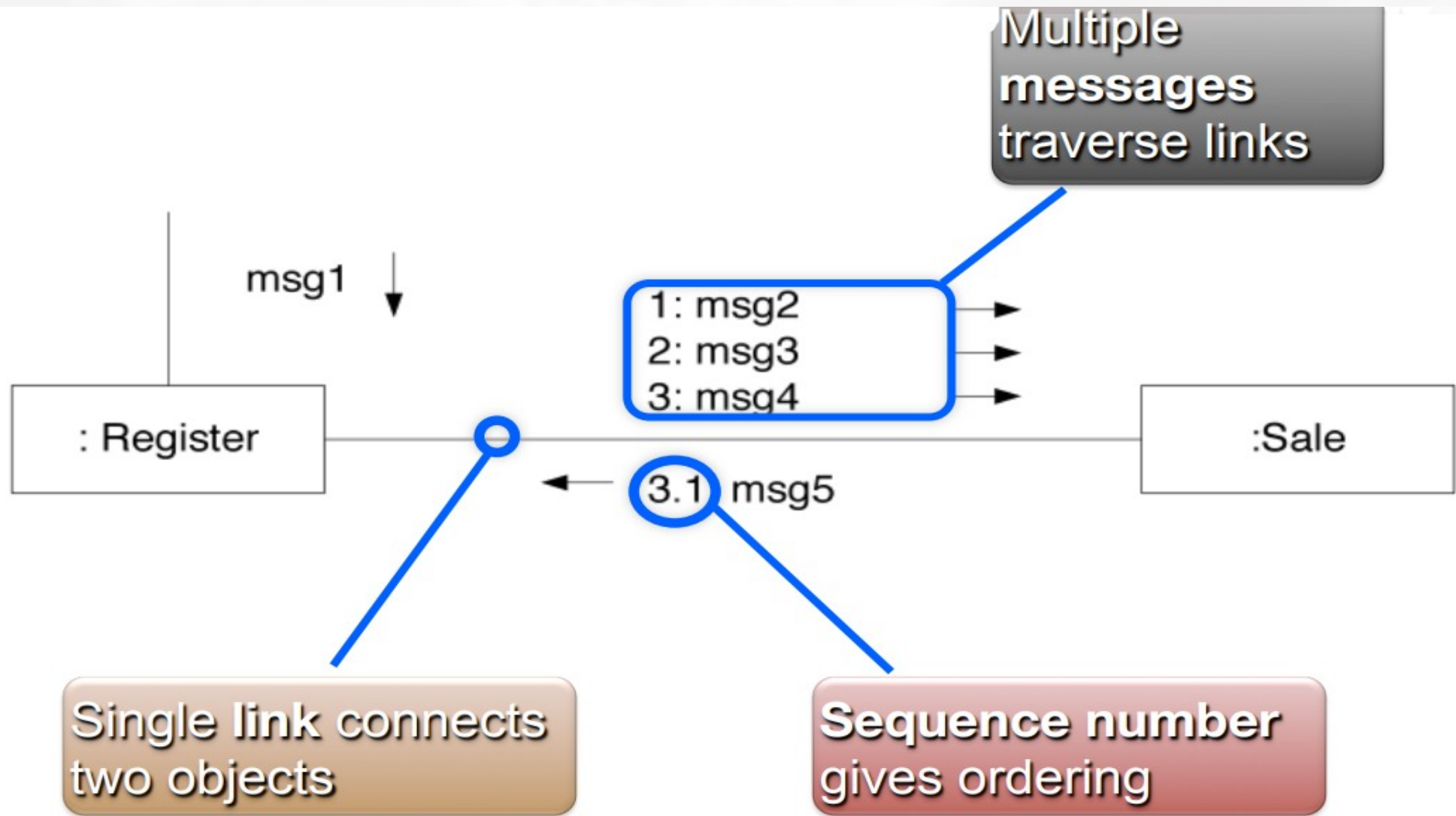
SeqNumber: aMessage →

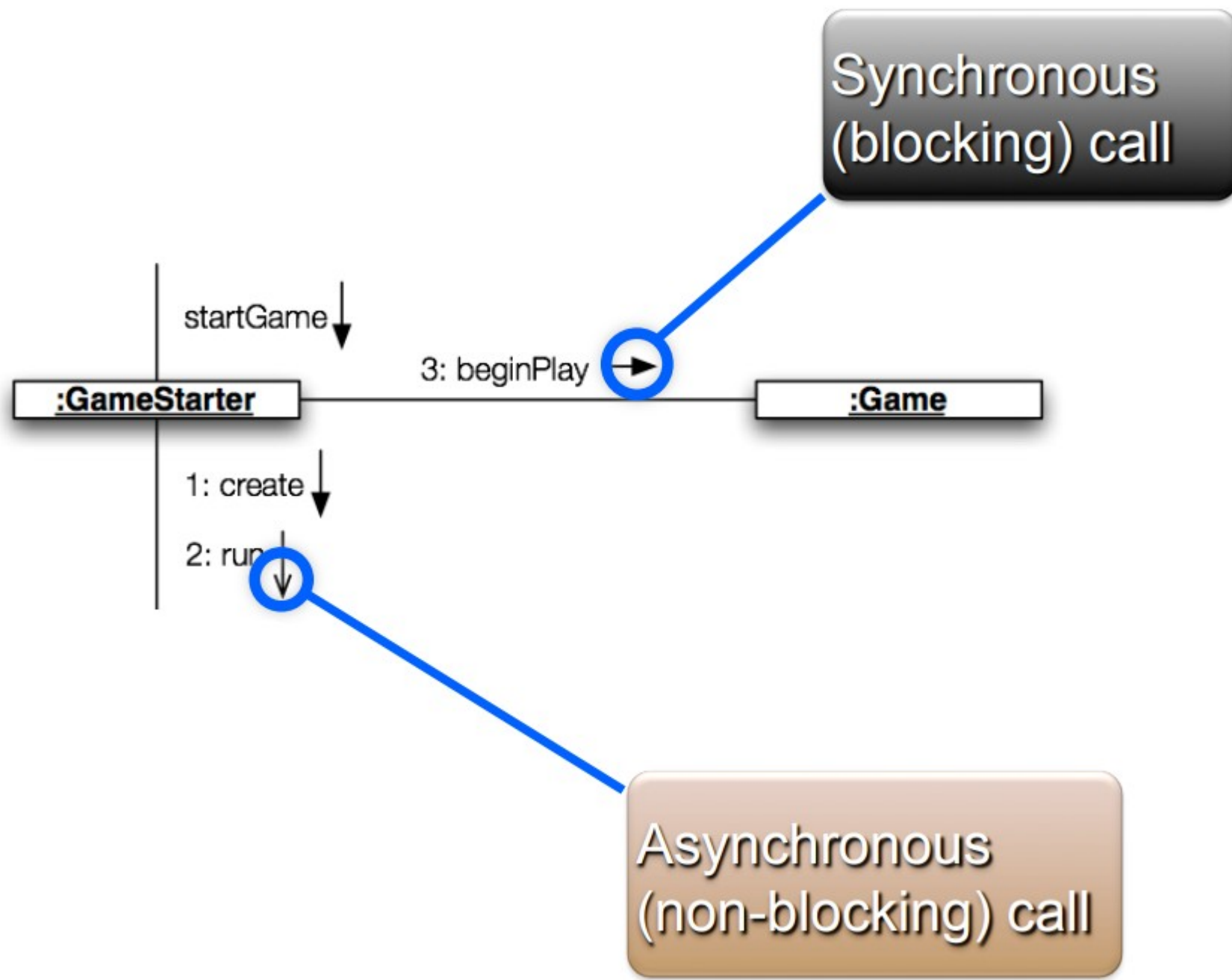
Guard Condition

- Represent a test that must be met for the message to be sent

A diagram showing a horizontal arrow pointing to the right. Above the arrow is the text "SeqNumber: [aGuardCondition]: aMessage".

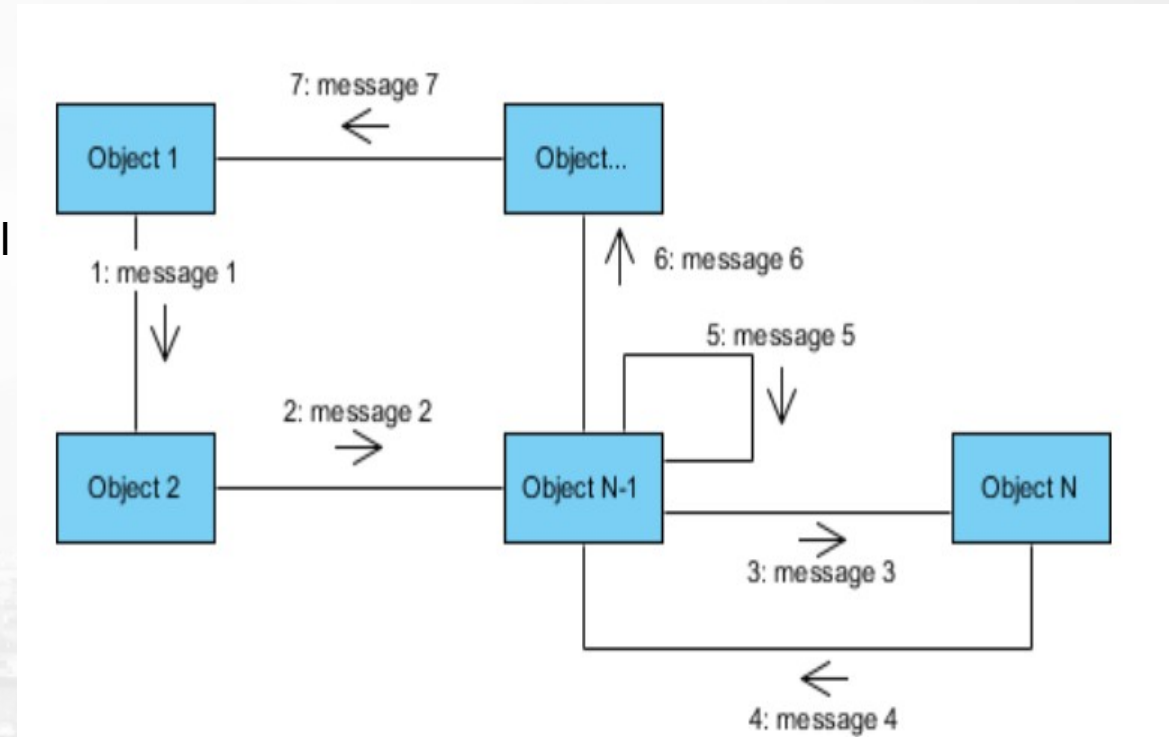
SeqNumber: [aGuardCondition]: aMessage →



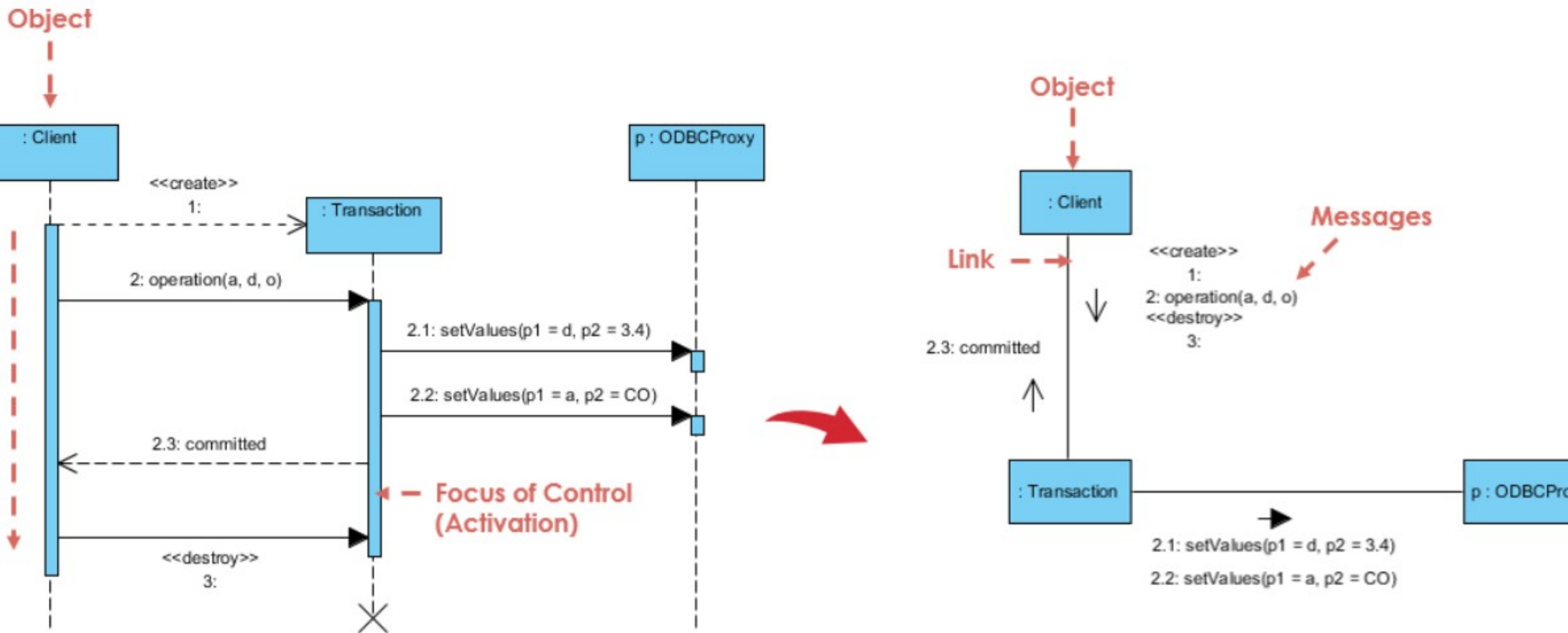


Communication Diagram at a Glance

- The objects are Object1, Object2, ..., ObjectN.
- Messages passed between objects are represented by labeled arrows that start with the sending object (actor) and end with the receiving object.
- The sample messages passed between objects are labeled 1: message1, 2: message2, 3: message3, etc., where the numerical prefix to the message name indicates its order in the sequence.
- Object1 first sends Object2 the message message1, Object2 in turn sends ObjectN-1 the message message2, and so on.
- Messages that objects send to themselves are indicated as loops (e.g., message message5).
- Each message in a communication diagram has a sequence number.
- Messages sent during the same call have the same decimal prefix, but suffixes of 1, 2, etc. according to when they occur.



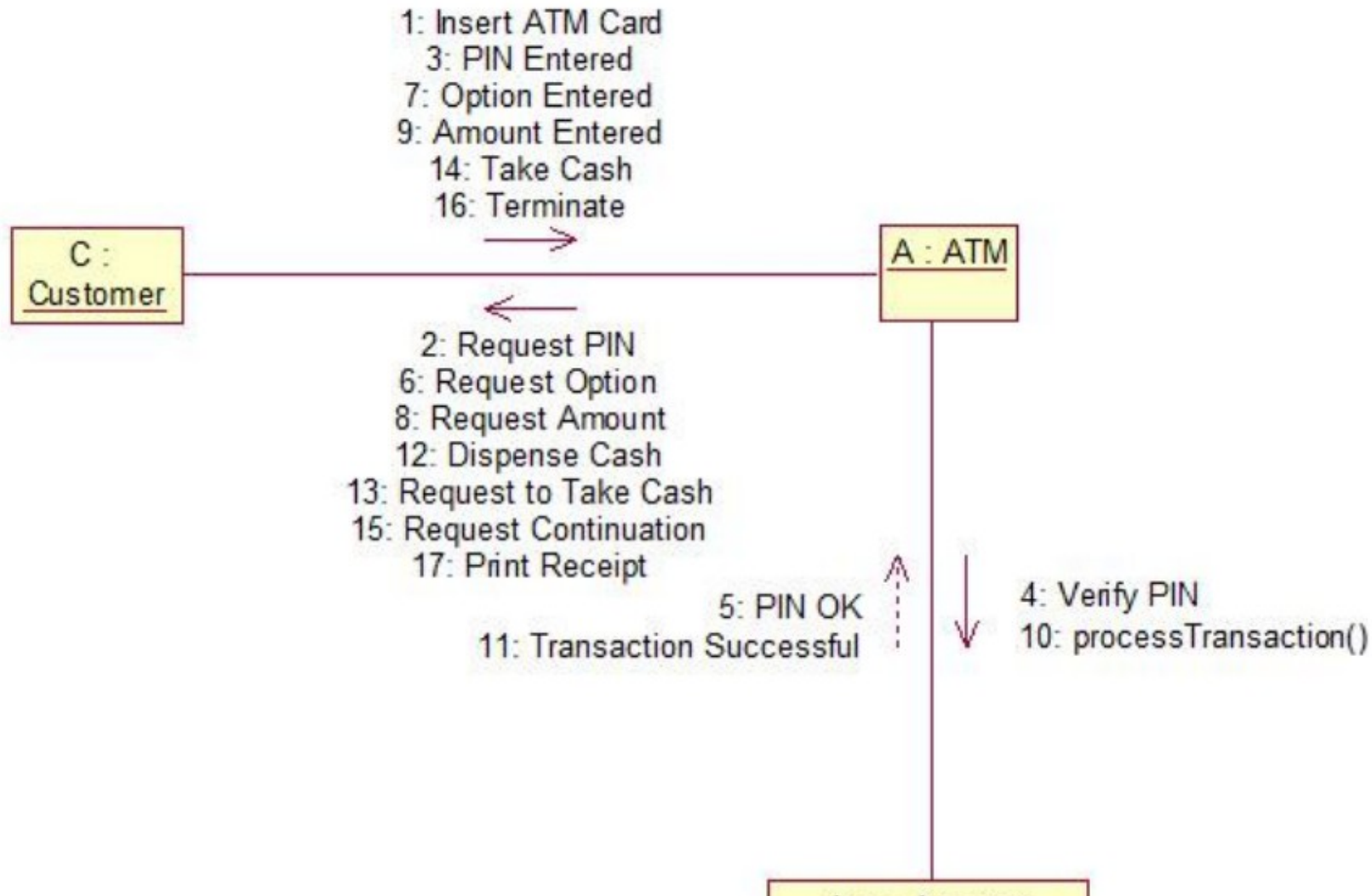
From Sequence Diagram to Communication Diagram



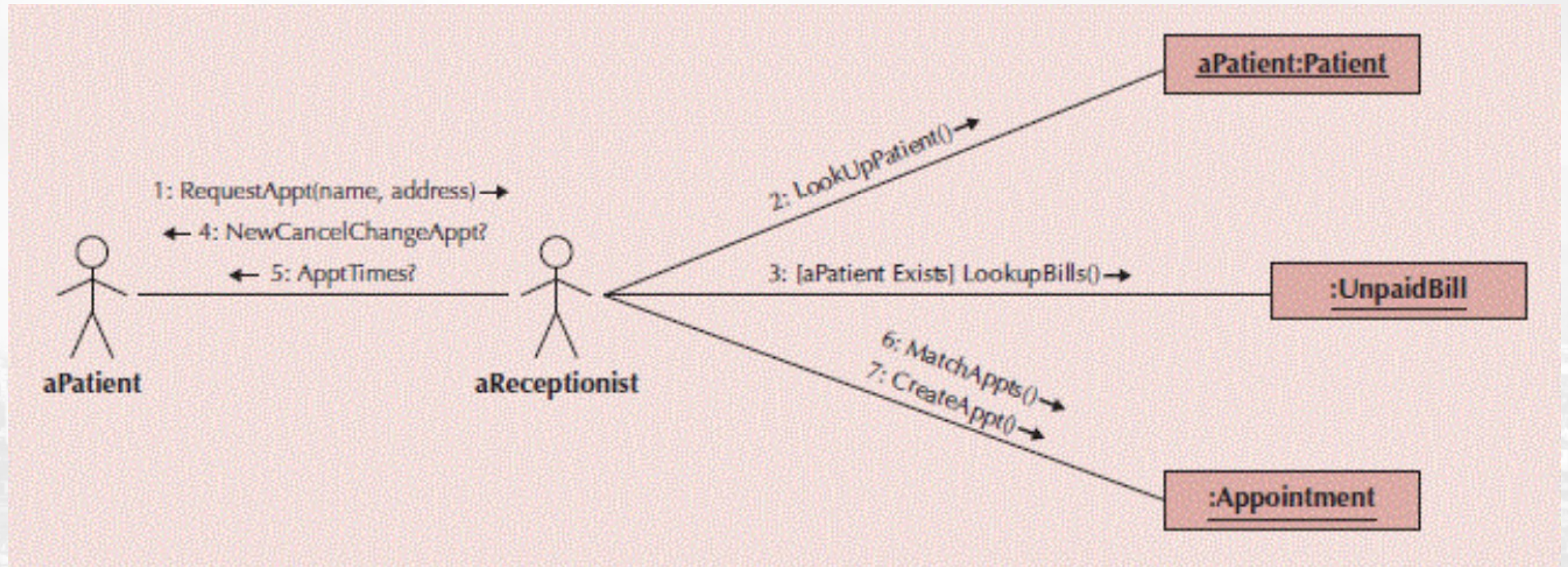
Example: Scenario (Course Registration)



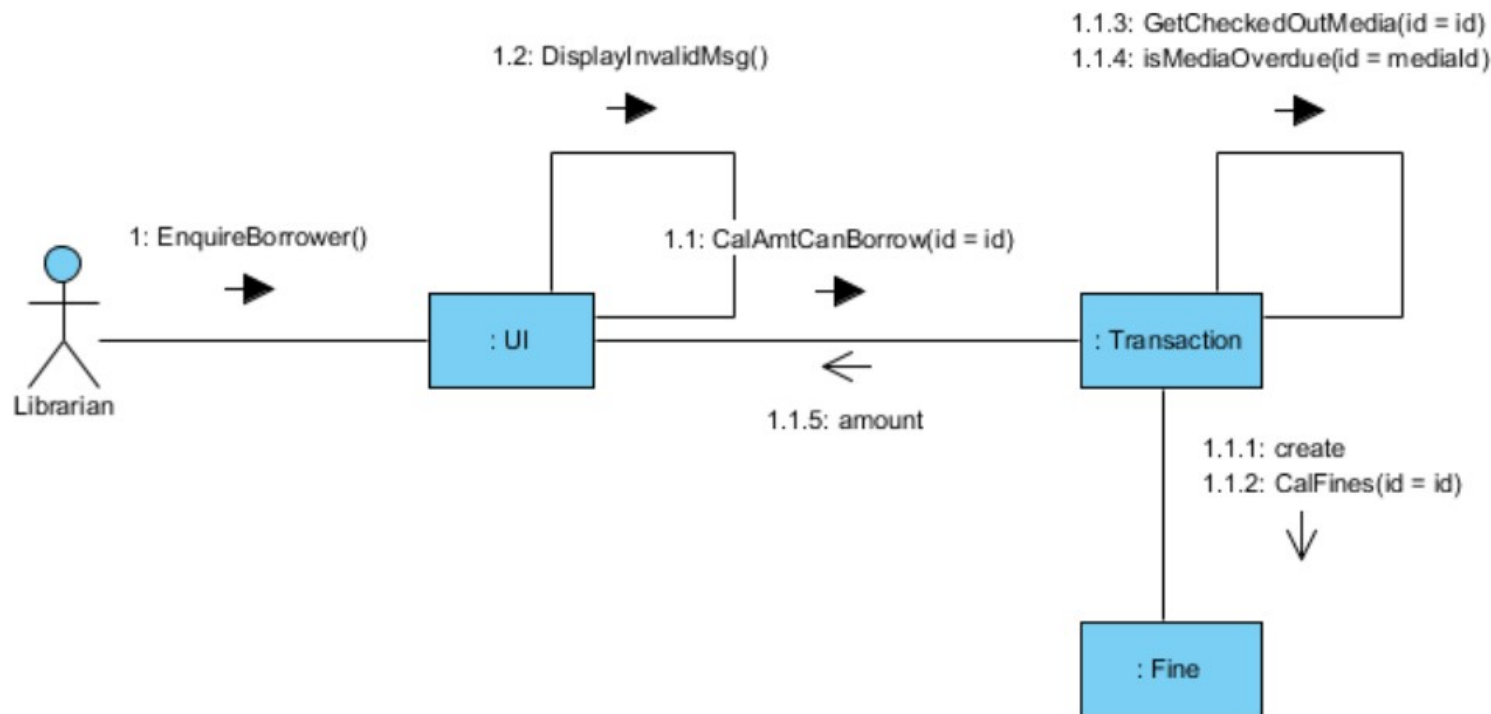
Example: Scenario (Withdraw Cash from ATM)



Example: Scenario (Book Doctor's Appointment)



Example: Scenario(Library Book Overdue)



1. EnquireBorrower

1.1 CalAmtCanBorrow

1.1.1 create

1.1.2 CalFines

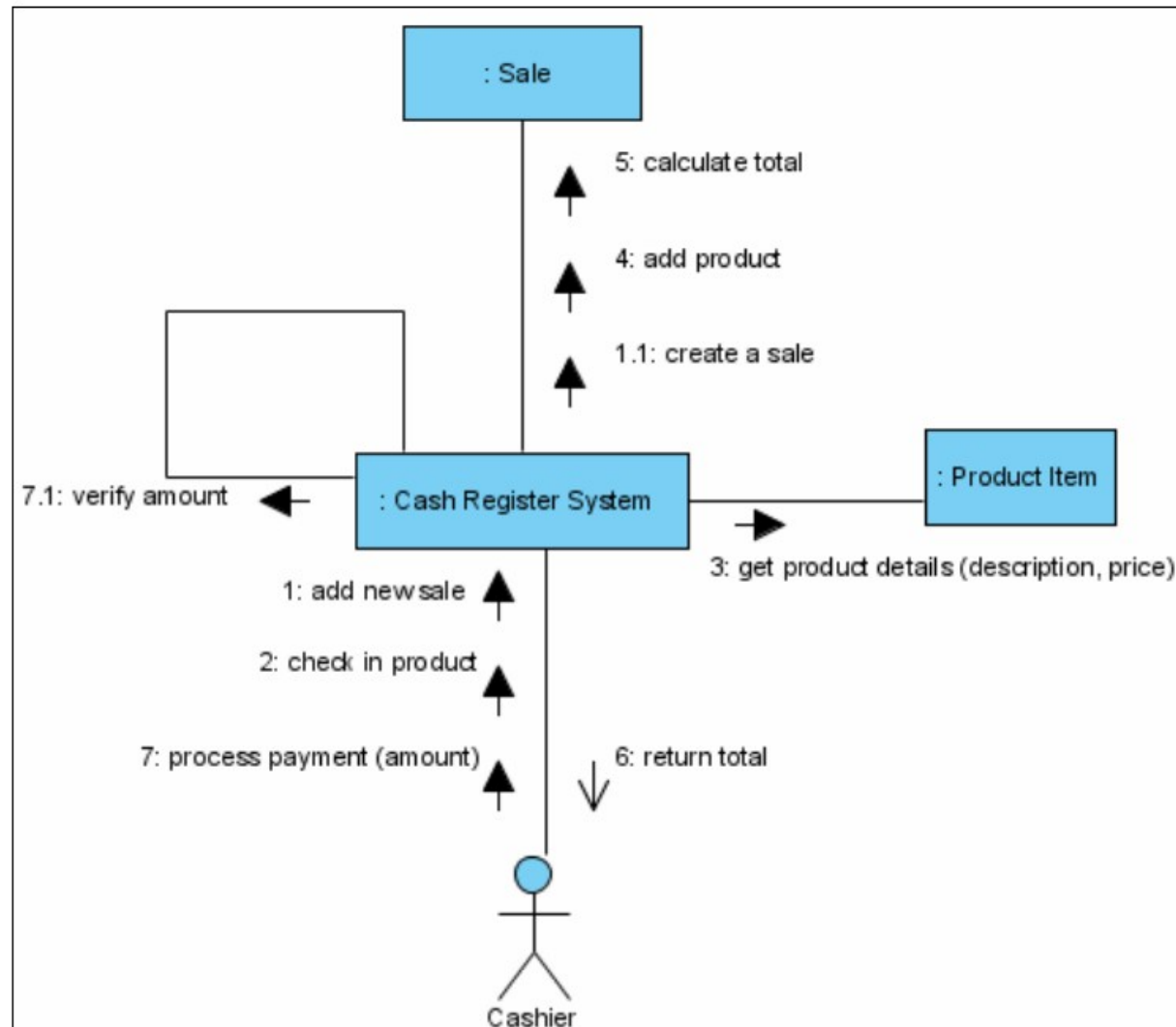
1.1.3 GetCheckedOutMedia

1.1.4 IsMediaOverdue

1.1.5 amount

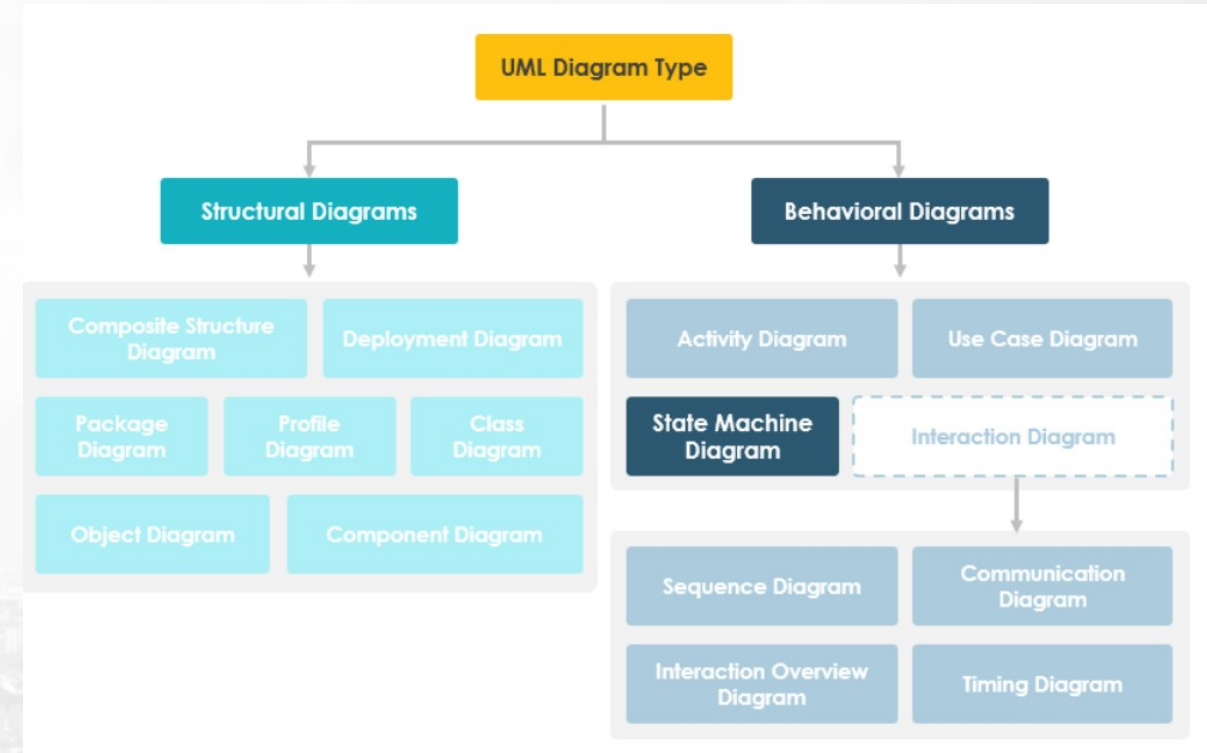
1.2 DisplayInvalidMsg

Example: Scenario(Checkout from a shop)



State Machine Diagram

- A state diagram describes the behavior of a system, some part of a system, or an individual object.
- At any given point in time, the system or object is in a certain state.
 - Being in a state means that it will behave in a specific way in response to any events that occur.
- Some events will cause the system to change state.
 - In the new state, the system will behave in a different way to events.



State diagrams are good for modeling the lifetime of an object or actor, also for modeling business processes which involve many states.

State Machine Diagram Elements

1. A State

Represents the state of object at an instant of time

Situation during the life of an object in which it satisfies some condition, performs some activity, or waits for some event.

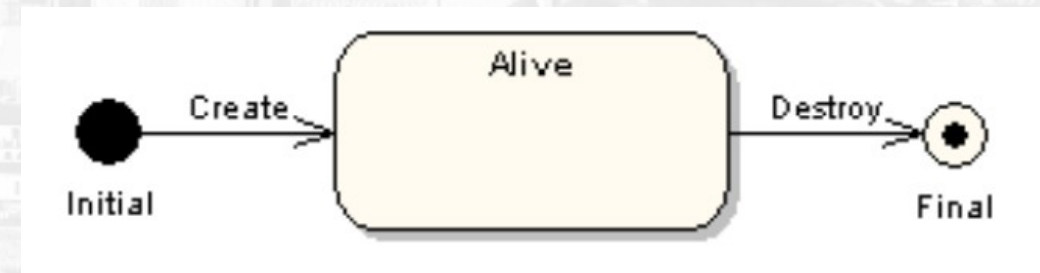
A state may include:

- Name
- Entry/exit actions
- Internal transitions
- Activities
- Substates



2. Initial state - indicates the initial starting state for the state machine or a substate.

3. Final state - indicates the state machine's execution has completed.



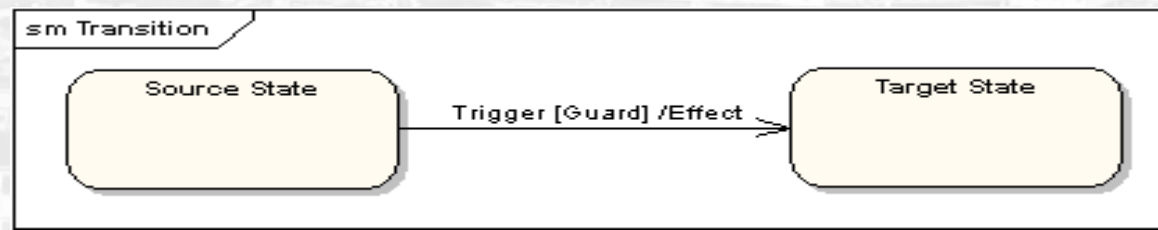
State Machine Diagram Elements

4. Transitions

- Relationship between two states indicating that an object in the first state will perform certain actions and enter the second state when a specified event occurs, and specified conditions are satisfied.
- On such a change of state, the transition is said to fire.
- Until the transition fires, the object is said to be in the *source state*; after it fires, it is said to be in the *target state*.

5. Event /Trigger

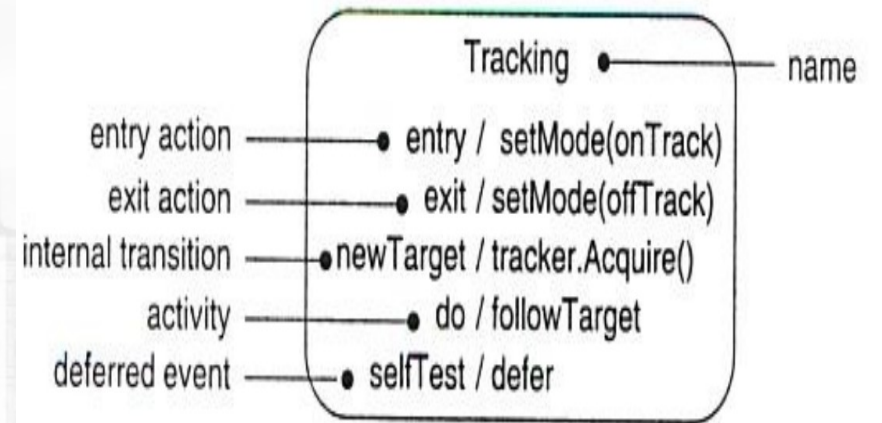
- "Trigger" is the cause of the transition, which could be a signal, an event, a change in some condition, or the passage of time.
- "Guard" is a condition which must be true in order for the trigger to cause the transition.
- "Effect" is an action which will be invoked directly on the object that owns the state machine as a result of the transition.



State Machine Diagram

A state has several parts.

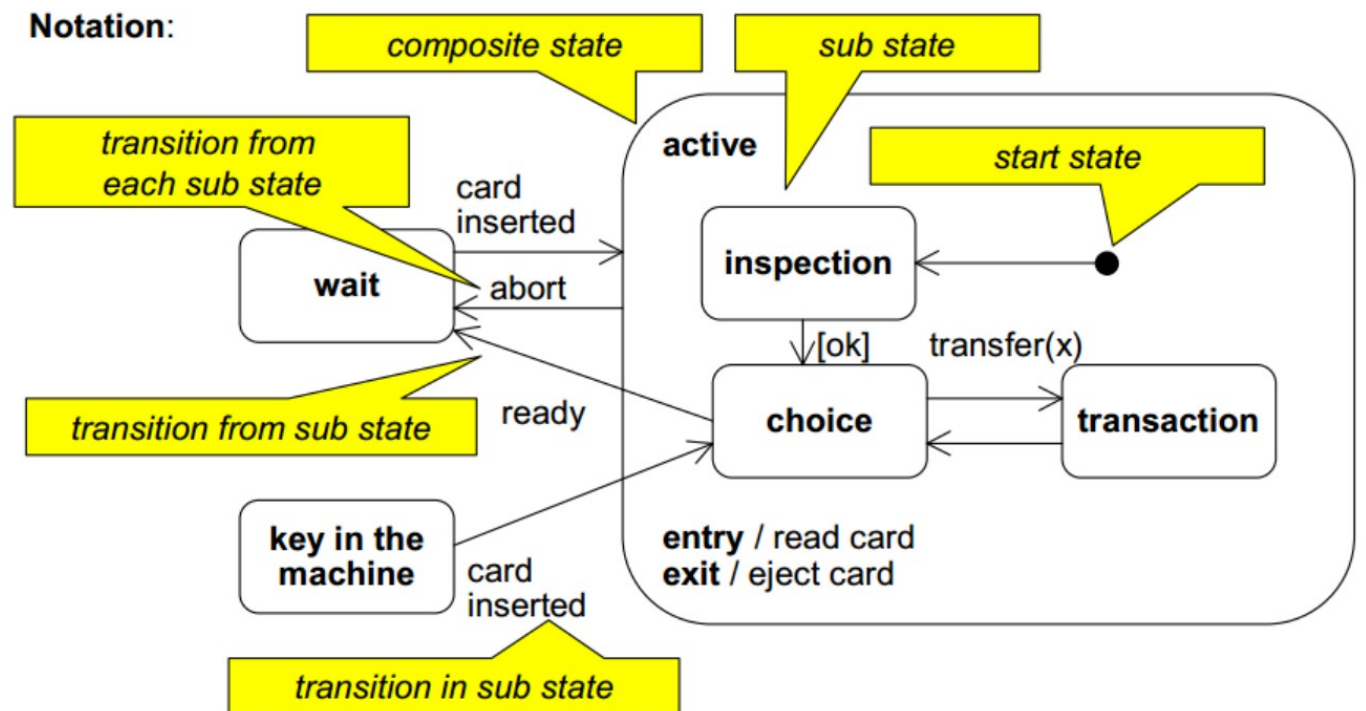
1. **Name** -A textual string that distinguishes the state from other states; a state may be anonymous, meaning that it has no name
2. **Entry action** - Upon each entry to a state, a specified action is automatically executed.
 - Syntax (to be placed inside the state symbol): entry / action
3. **Exit action** - Just prior to leaving a state, a specified action is automatically executed.
 - Syntax (to be placed inside the state symbol): exit / action
4. **Internal Transitions** - The handling of an event without leaving the current state.
 - Used to avoid a states entry and exit actions.
 - Syntax (to be placed inside the state symbol): event / action
5. **Activities** - Ongoing work that an object performs while in a particular state. The work automatically terminates when the state is exited.
 - Syntax (to be placed inside the state symbol): do / activity
6. **Deferred Event** - An event whose occurrence is responded to at a later time.
 - Syntax (to be placed inside the state symbol): event / defer



State Machine Diagram

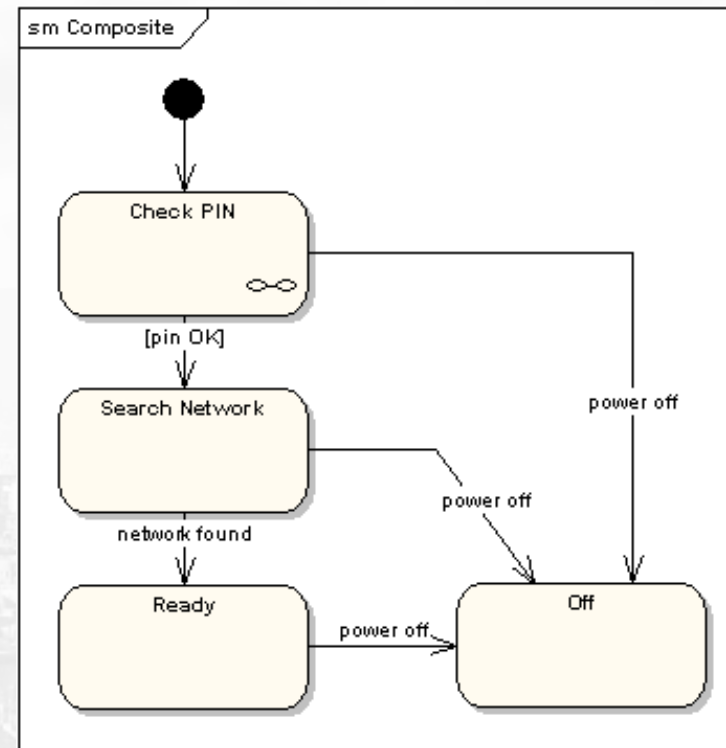
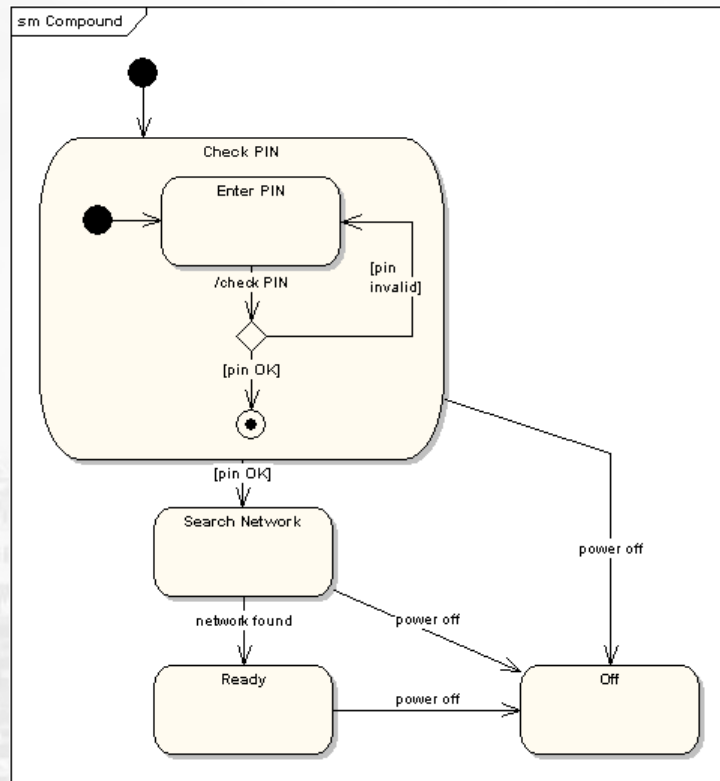
Advanced States

1. **Simple state** - A state that contains no substates.
2. **Composite state** - A state that contains substates.
3. **Substate** - A state that is nested inside another state.
 - Substates allow state diagrams to show different levels of abstraction.
 - Substates may be sequential or concurrent.
 - Substates may be nested to any level.



State Machine Diagram

- **Compound States** - A state machine diagram may include sub-machine diagrams, as in the example below.

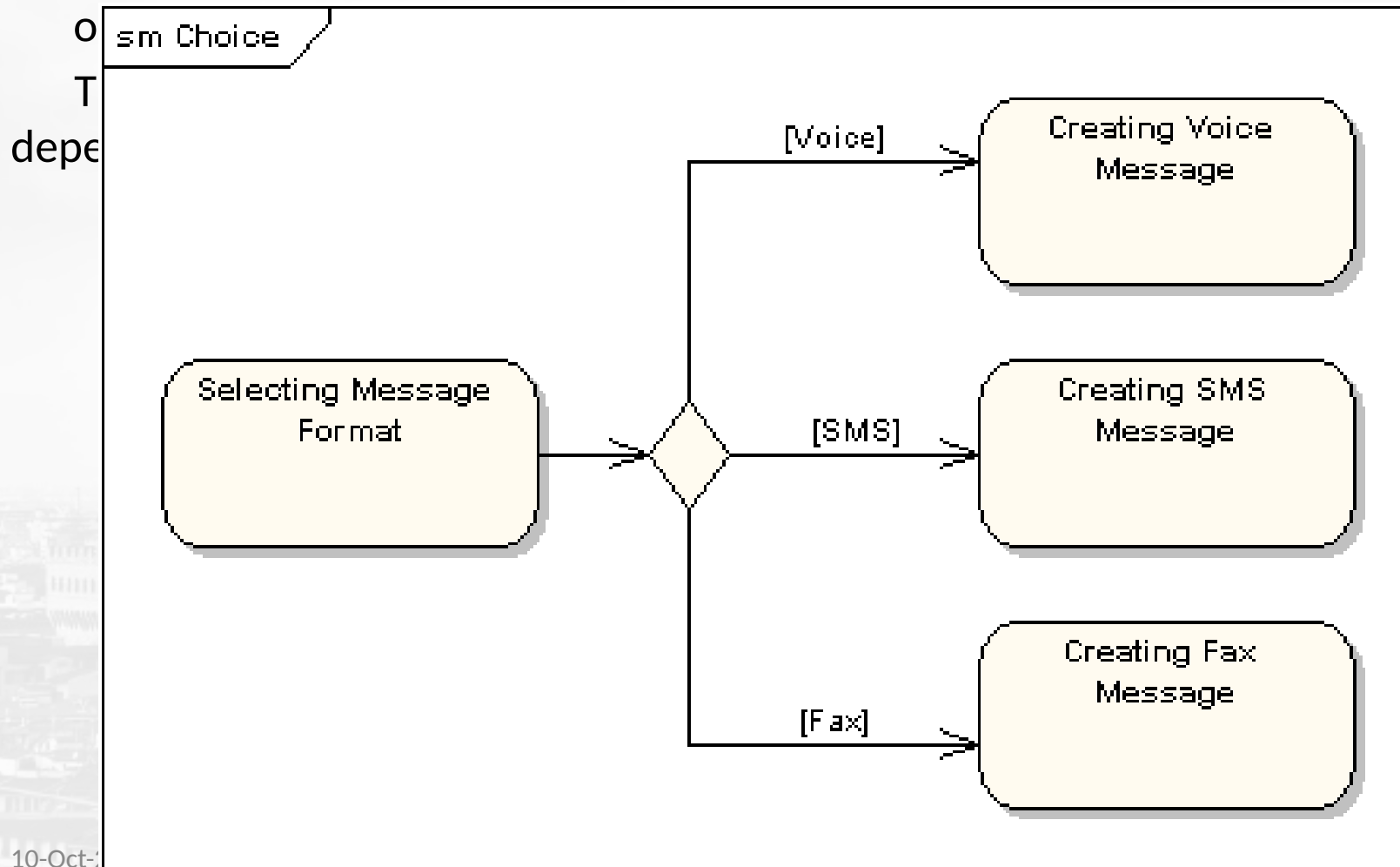


Alternative way to show the same information

- The ∞ symbol indicates that details of the Check PIN sub-machine are shown in a separate diagram.

State Machine Diagram

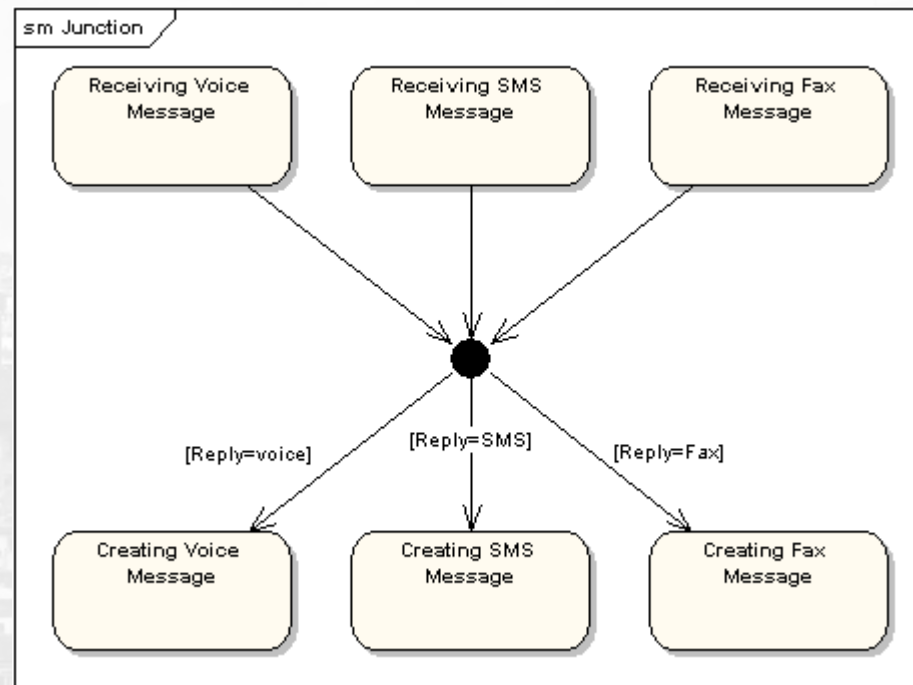
- **Choice Pseudo-State** - A choice pseudo-state is shown as a diamond with one transition arriving and two



the choice pseudo-state, is
this state.

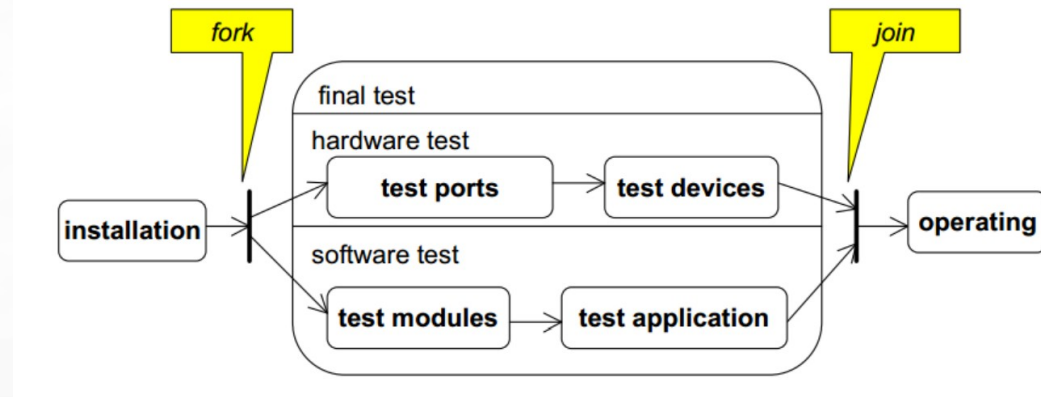
State Machine Diagram

- **Junction Pseudo-State** - Junction pseudo-states are used to chain together multiple transitions. A single junction can have one or more incoming, and one or more outgoing, transitions; *a guard* can be applied to each transition. A junction which splits an incoming transition into multiple outgoing transitions realizes a static conditional branch

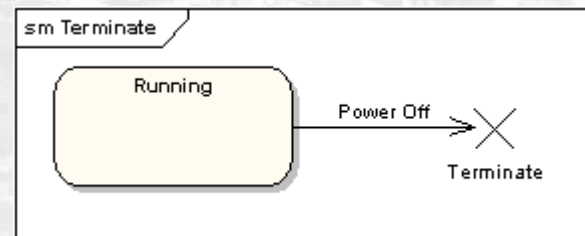


State Machine Diagram Elements

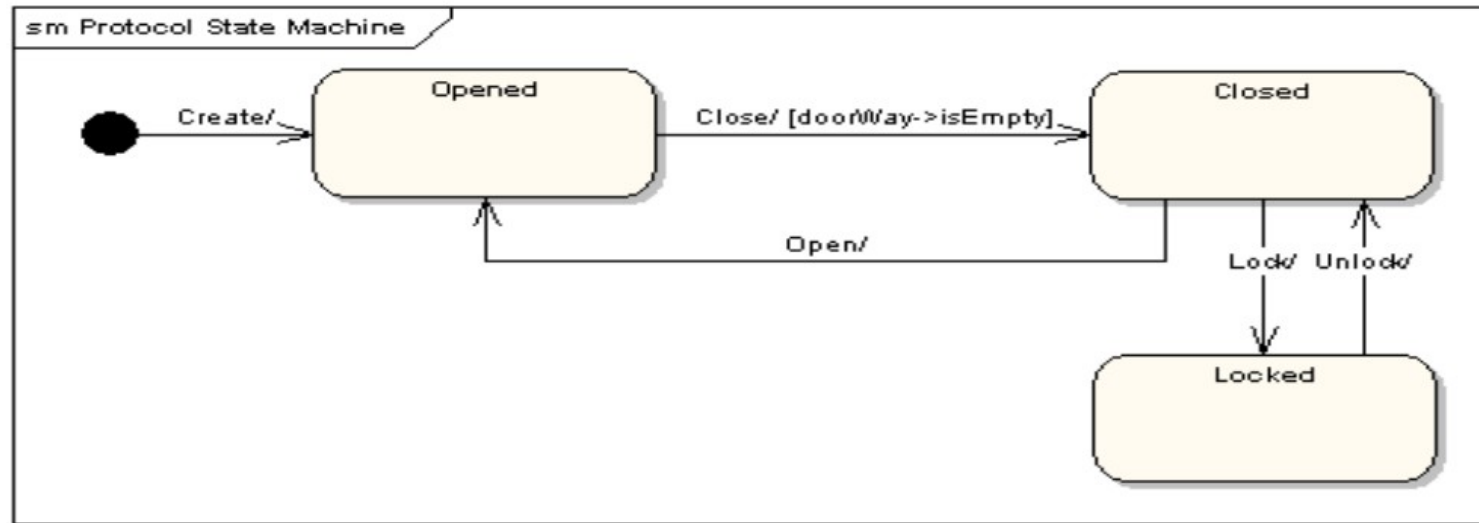
- **Concurrent Regions** - A state may be divided into regions containing substates that exist and execute concurrently.



- **Terminate Pseudo-State** - Entering a terminate pseudo-state indicates that the lifeline of the state machine has ended. A terminate pseudo-state is notated as a cross.



Example to Understand



The state machine diagram shows the states that a door goes through during its lifetime.

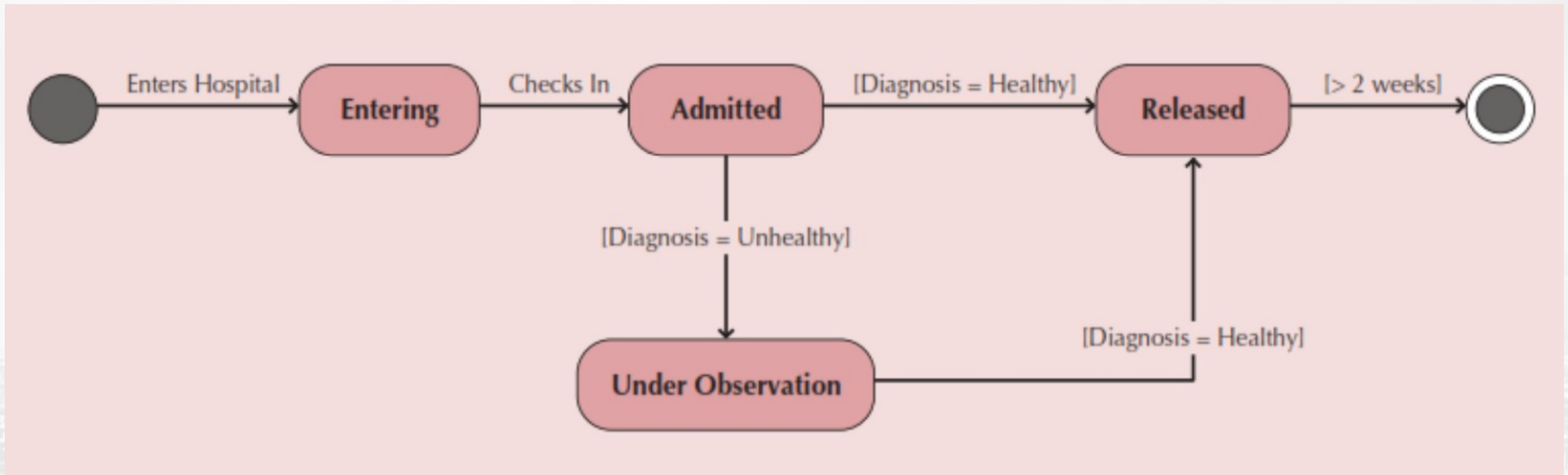
- The door can be in one of three states: "Opened", "Closed" or "Locked".
- Notice that not all events are valid in all states;

Example: if a door is opened, you cannot lock it until you close it.

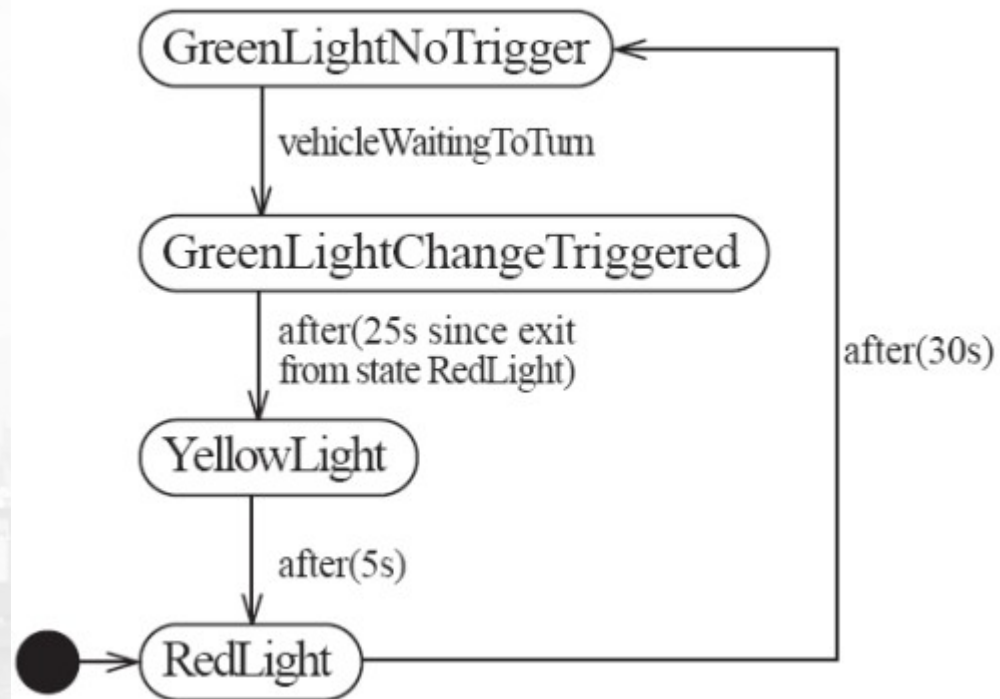
- Also notice that a state transition can have a guard condition attached

Example: if the door is Opened, it can only respond to the Close event if the condition `doorWay->isEmpty` is fulfilled.

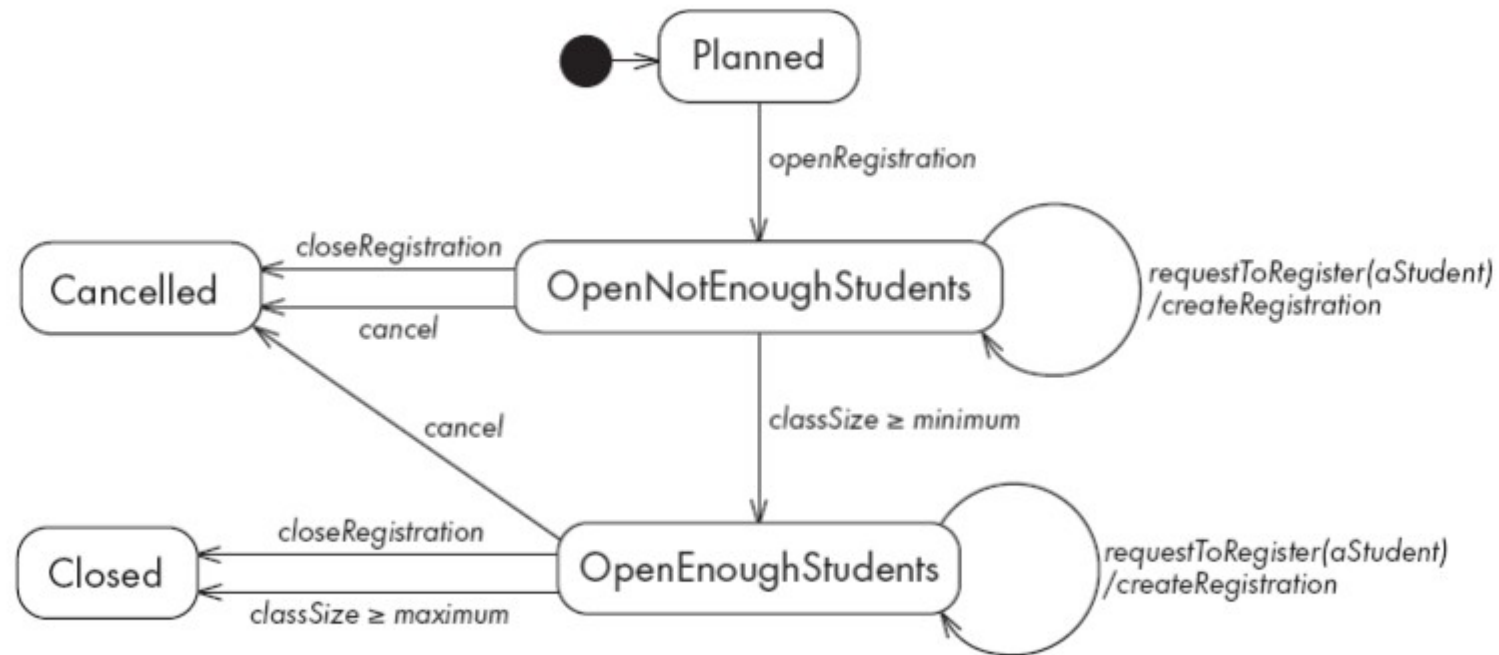
Example: Stages of a Patient



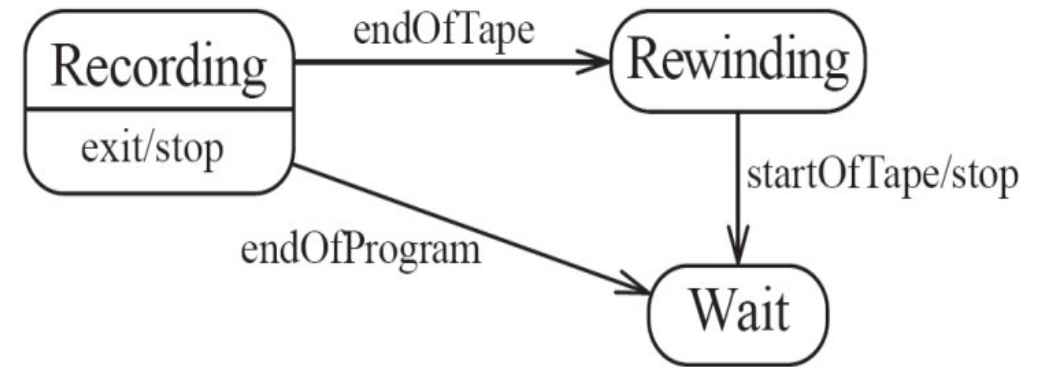
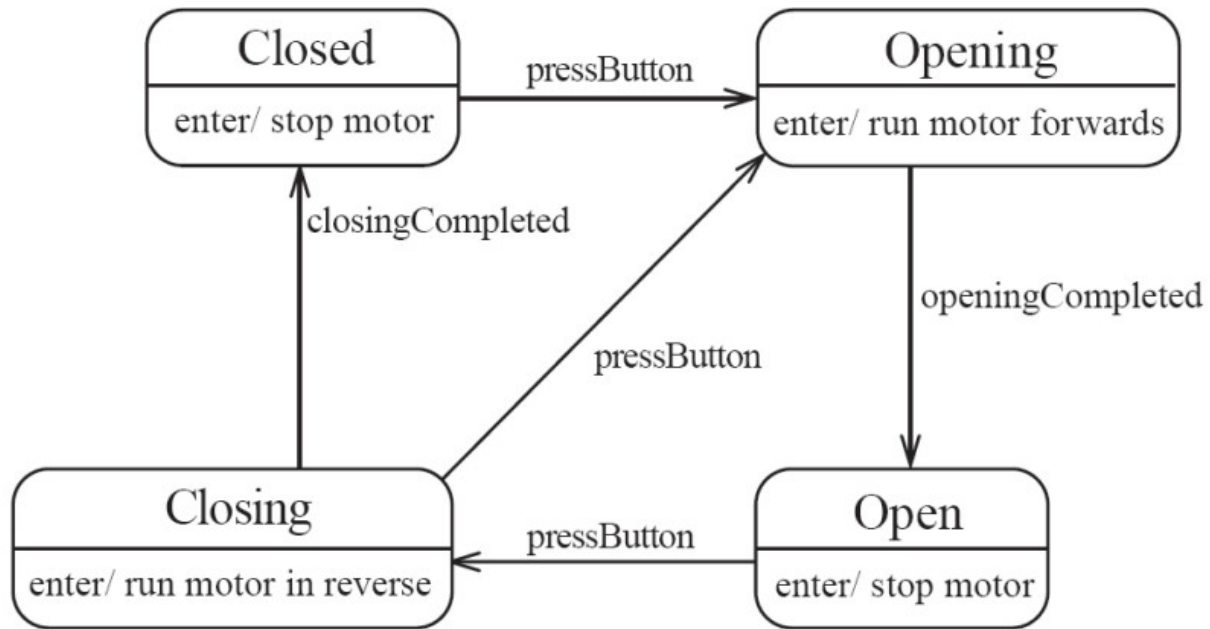
Example with conditions



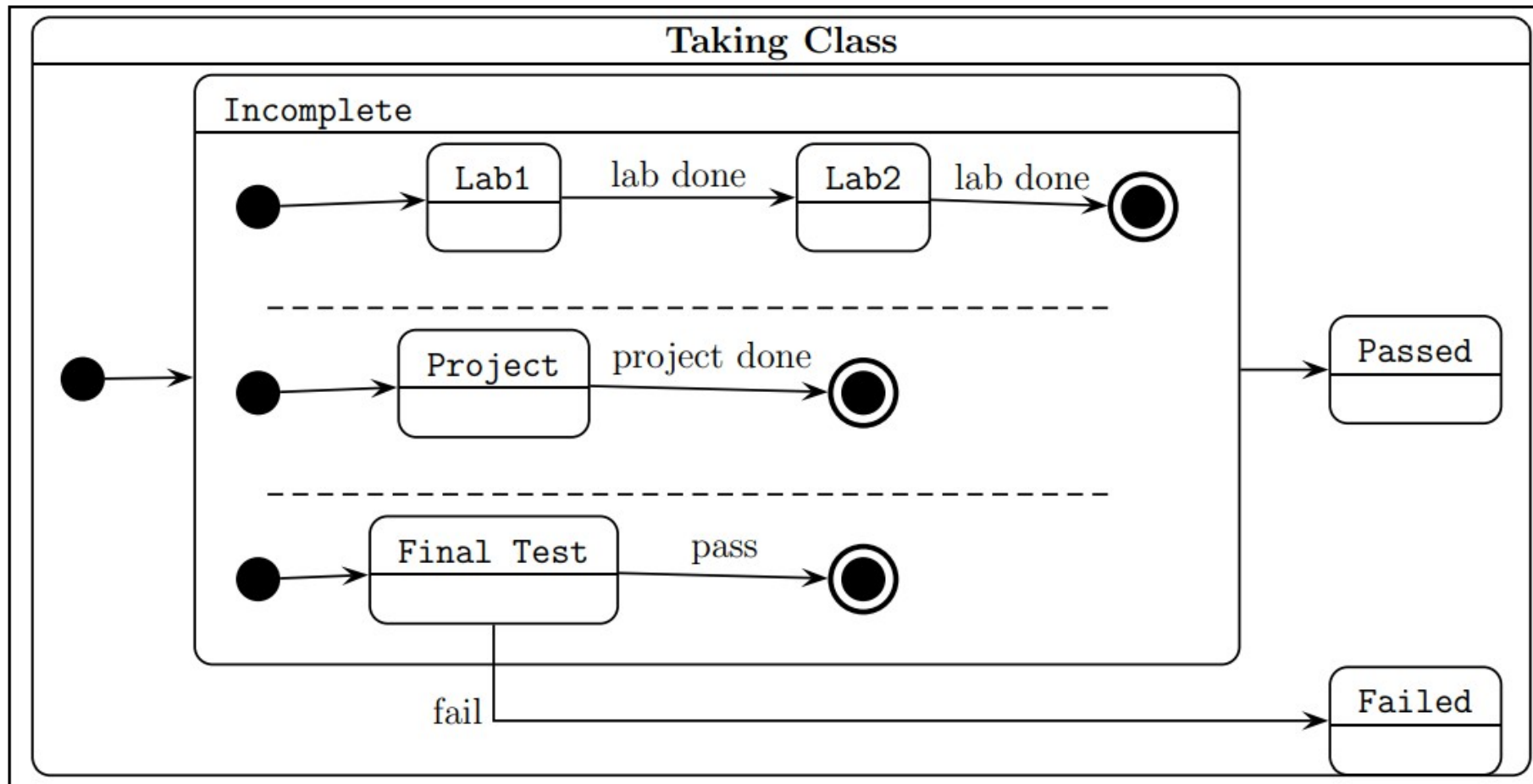
Example with conditional transitions

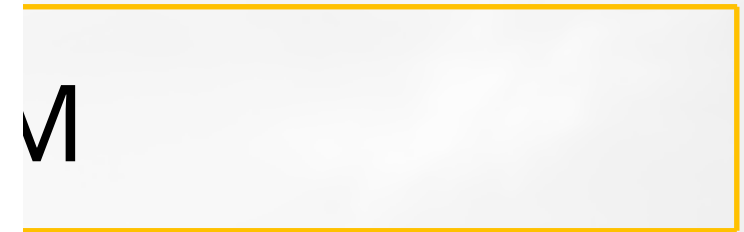
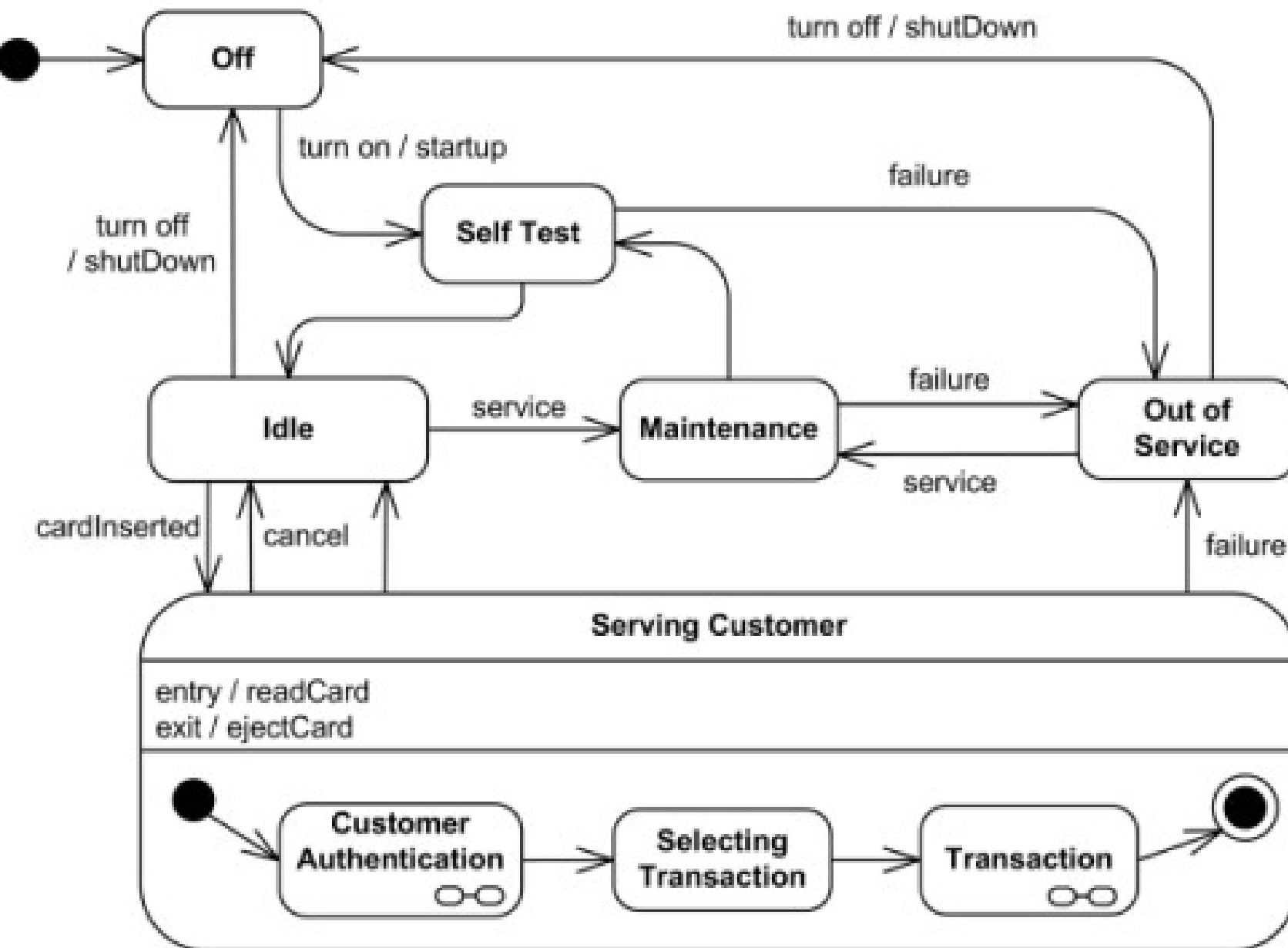


Example with entry/exit actions

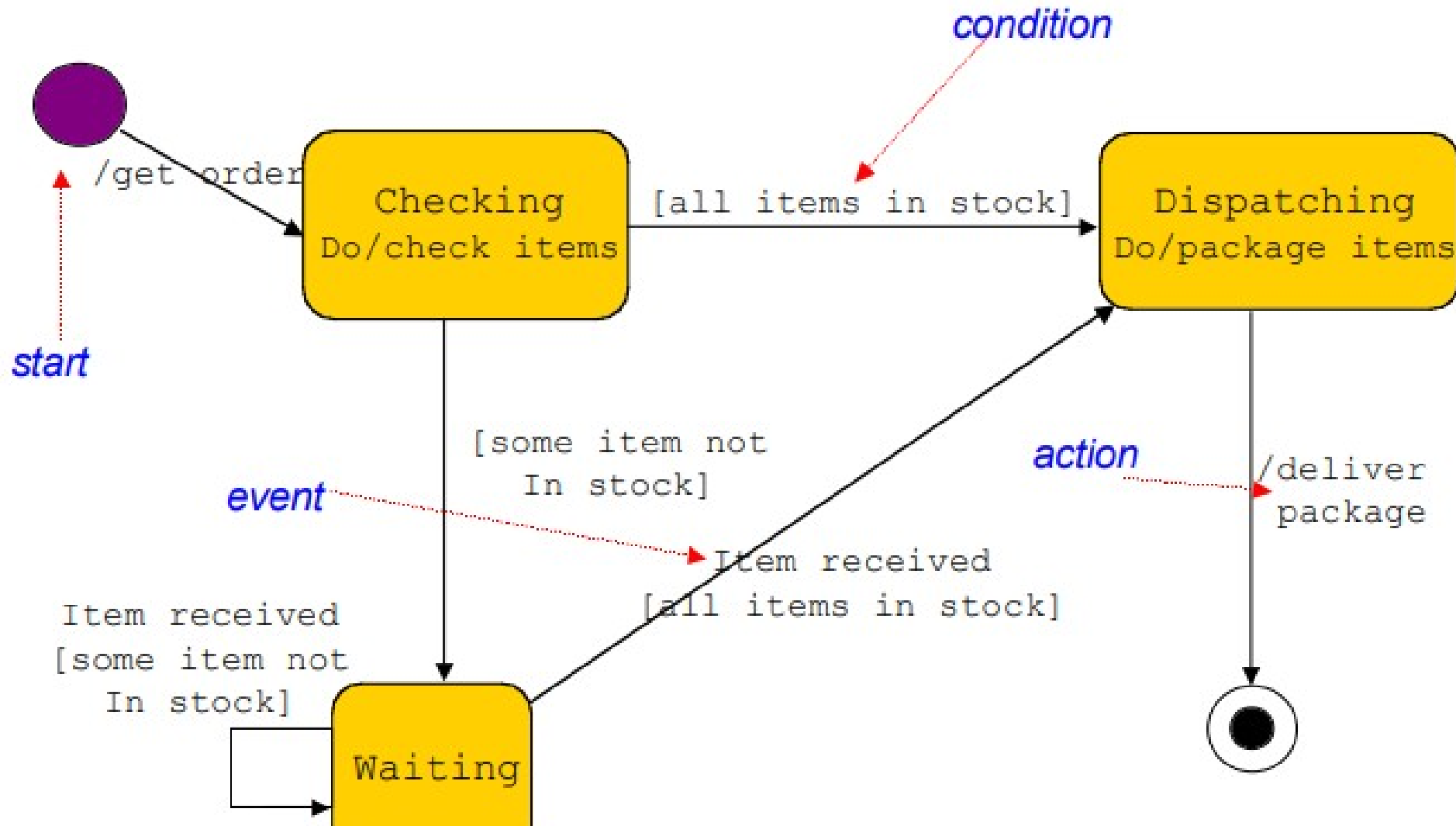


Example: Concurrent Regions





Example: Purchase Order



Example: Drive Vehicle

