Vietnam National University of HCMC
International University
School of Computer Science and Engineering

Object – Oriented Analysis and Design

State Diagram

Instructor: Le Thi Ngoc Hanh, Ph.D

ltnhanh@hcmiu.edu.vn

Content

- State diagram in UML
- Reading: [R3] – Chapter 5, Section 5.11

Image credit: UML-diagram; materials shared by dept.CSE

Why state diagrams?

Object-orientation = Structure + Behavior

- How do we catch the dynamic behavior and life cycle of an object?
  - Creation and deletion.
  - Attribute and association changes.
- How does the object interact with other objects?
  - Reacting to events and to messages received by the object.
  - Triggering actions and sending messages to other objects.
  - Handling of sequences of events accepted and actions triggered.

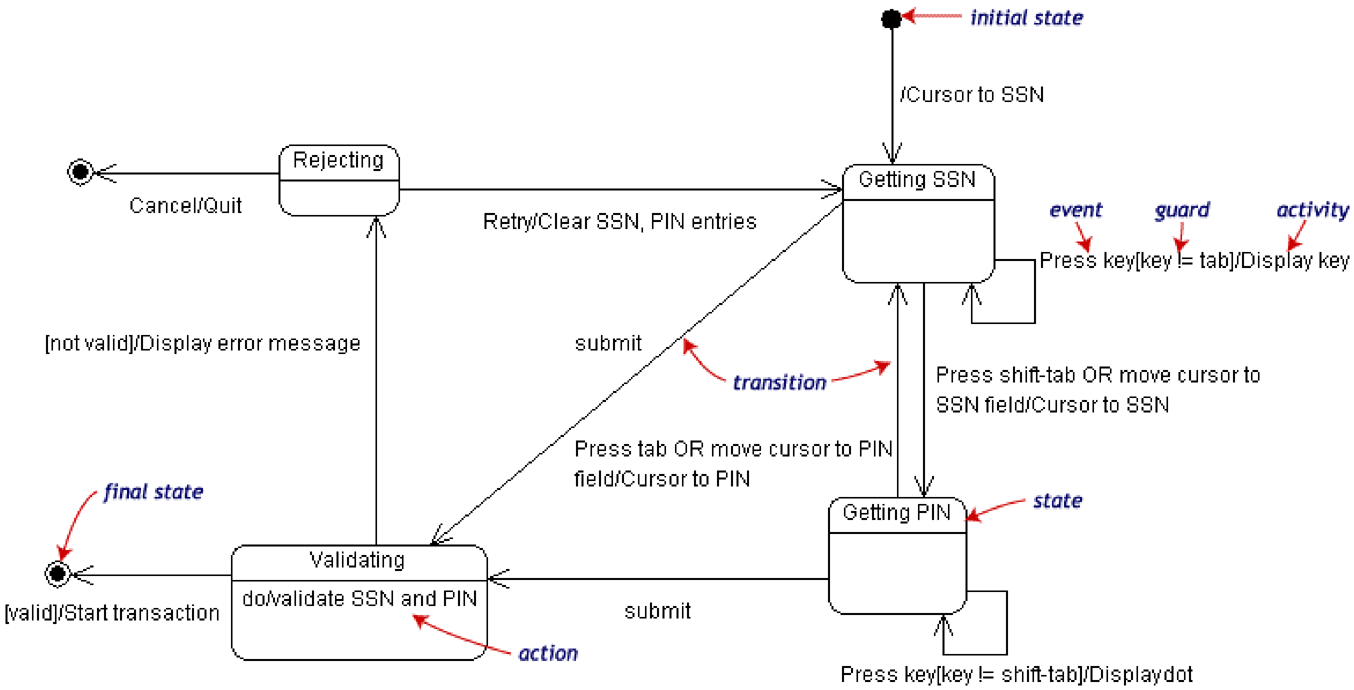
State Diagrams

State diagrams are a technique to describe the behavior, i.e., state changes of a single class according to events and messages which the class sends and receives.

Activity vs. State Diagrams

- Activity Diagrams are reducible to State Diagrams with some additional notations.
- Activity Diagrams: vertices represent the carrying out of an activity and the edges represent the transition on the completion of one collection of activities to the commencement of a new collection of activities.
- State Diagrams: vertices represent states of an object in a class and edges represent occurrences of events.

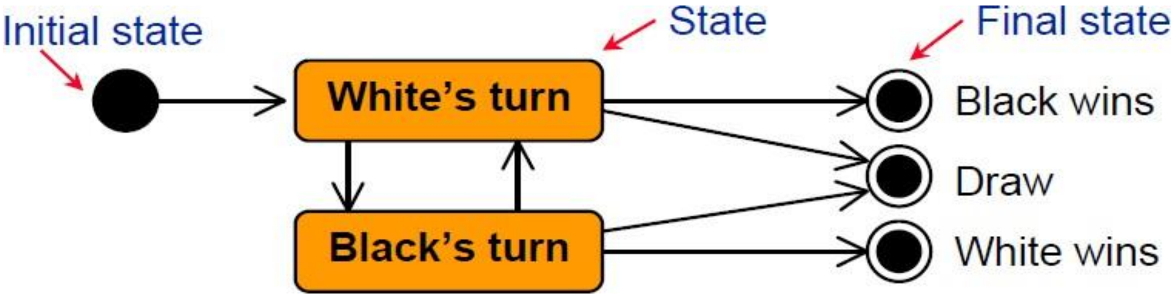
State diagram example


initial state
/Cursor to SSN
Rejecting
Getting SSN
Cancel/Quit
Retry/Clear SSN, PIN entries
event
guard
activity
Press key[key != tab]/Display key
[not valid]/Display error message
submit
transition
Press shift-tab OR move cursor to
SSN field/Cursor to SSN
Press tab OR move cursor to PIN
field/Cursor to PIN
final state
Getting PIN
state
Validating
do/validate SSN and PIN
[valid]/Start transaction
submit
action
Press key[key != shift-tab]/Displaydot

States

- A state:
  - abstracts from attribute values and associations of an object;
  - represents the internal condition/state of an object for a certain period of time;
  - corresponds to an interval of time between two events.
- The response to events may depend on the state of an object.
- Object creation comes together with an initial object

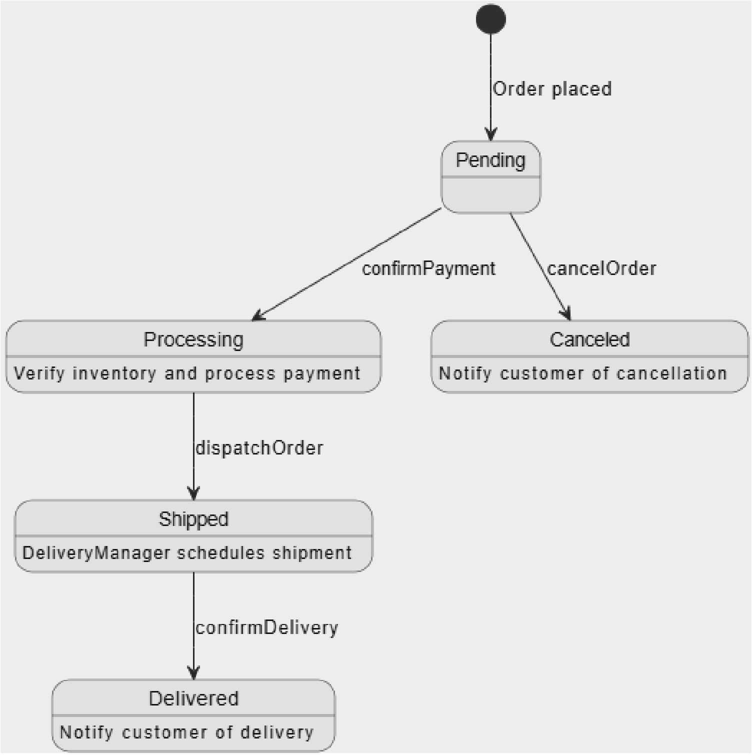
States


Initial state
State
Final state
White's turn
Black's turn
Black wins
Draw
White wins

Events

- Internal or External Events trigger some activity that changes the state of the system and of some of its parts.
- Events pass information, which is elaborated by Objects operations. Objects realize Events.
- Design involves examining events in a State Diagram and considering how those events will be supported by system objects.
- Events may be declared in a class diagram with arguments shown as attributes

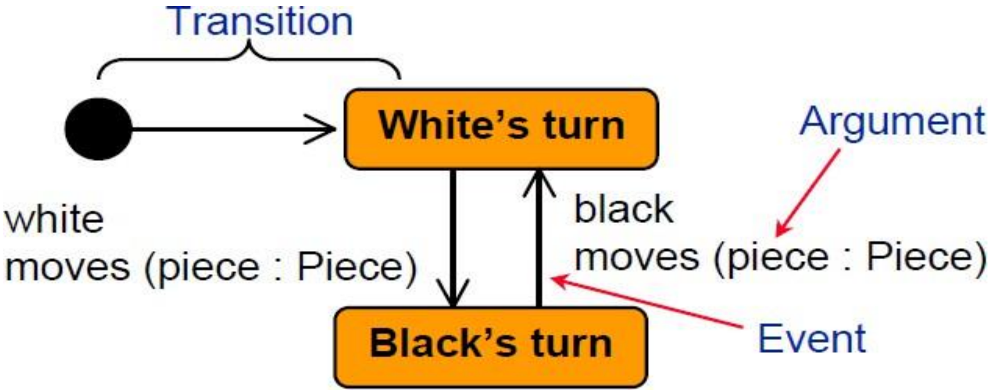
Events


Order placed
Pending
confirmPayment
cancelOrder
Processing
Verify inventory and process payment
Canceled
Notify customer of cancellation
dispatchOrder
Shipped
DeliveryManager schedules shipment
confirmDelivery
Delivered
Notify customer of delivery

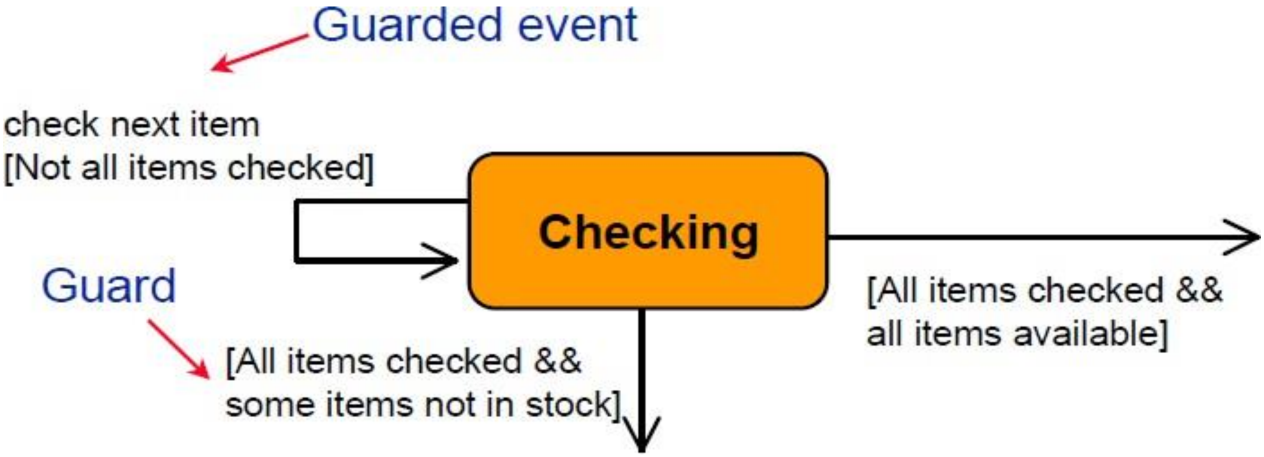
Transitions

- A transition represents a change of the internal condition/state of an object.
- A transition is usually triggered (“fired”) by an event. Transitions without event label (“lambda transitions”) fire immediately.
- Transitions fire instantly: from exactly one state to another state or to itself (self-transition).
- Multiple transitions occur either when different events result in a state terminating or when there are guard conditions on the transitions.
- A transition without an event and action is known as automatic transitions.

Transitions



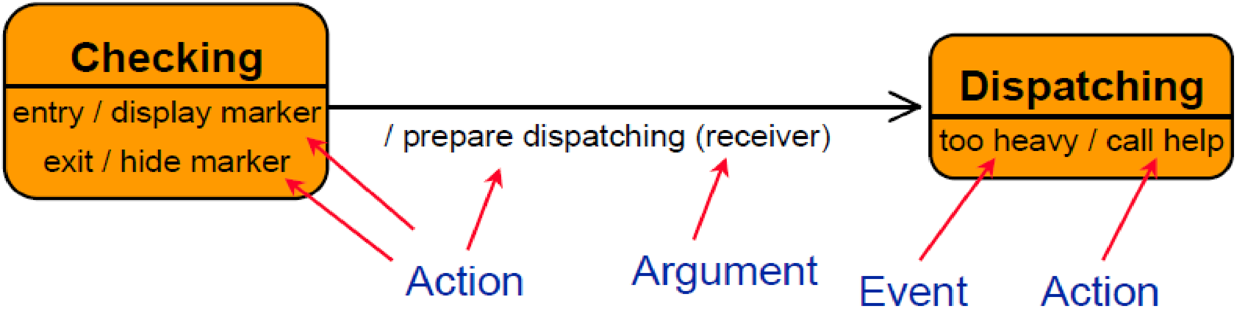
Guards


Guarded event
check next item
[Not all items checked]
Checking
Guard
[All items checked &&
all items available]
[All items checked &&
some items not in stock]

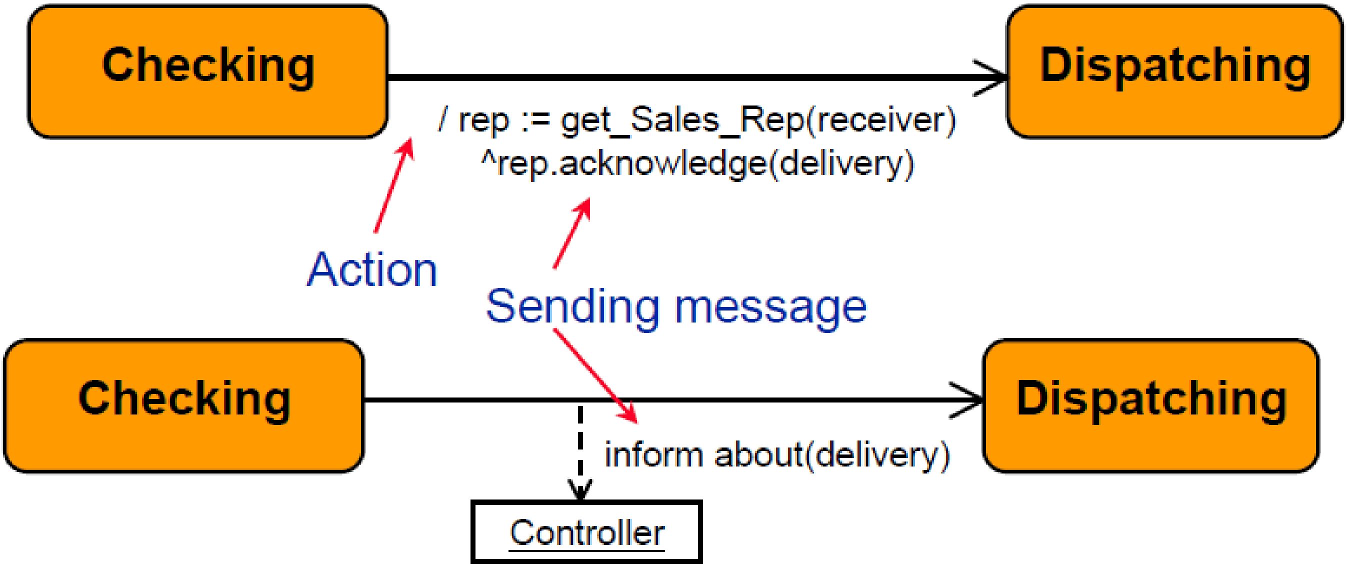
Actions

- An action is a short software process that executes immediately.
- A transition may trigger an action.
- Actions may be triggered on entry or exit of states (instead of labeling each incoming (entry) and outgoing (exit) transition with these actions).
- An event may trigger an action without leaving the state, i.e., without triggering exit and entry actions as a self-transition would do.
- An action may trigger events, usually in other objects.
- Actions may take arguments.

Actions

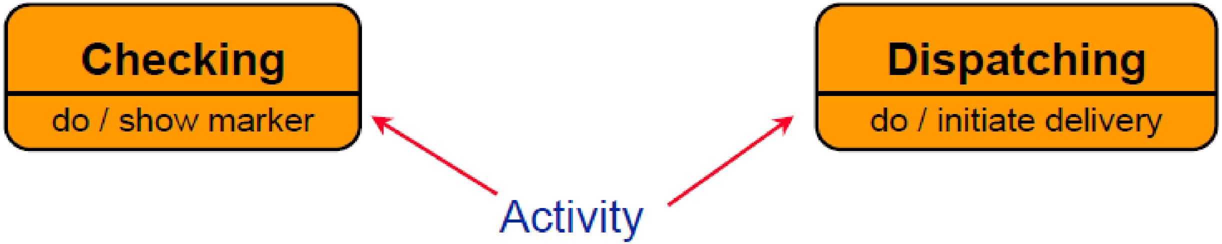

Checking
entry / display marker
exit / hide marker
/ prepare dispatching (receiver)
Dispatching
too heavy / call help
Action
Argument
Event
Action

Sending message

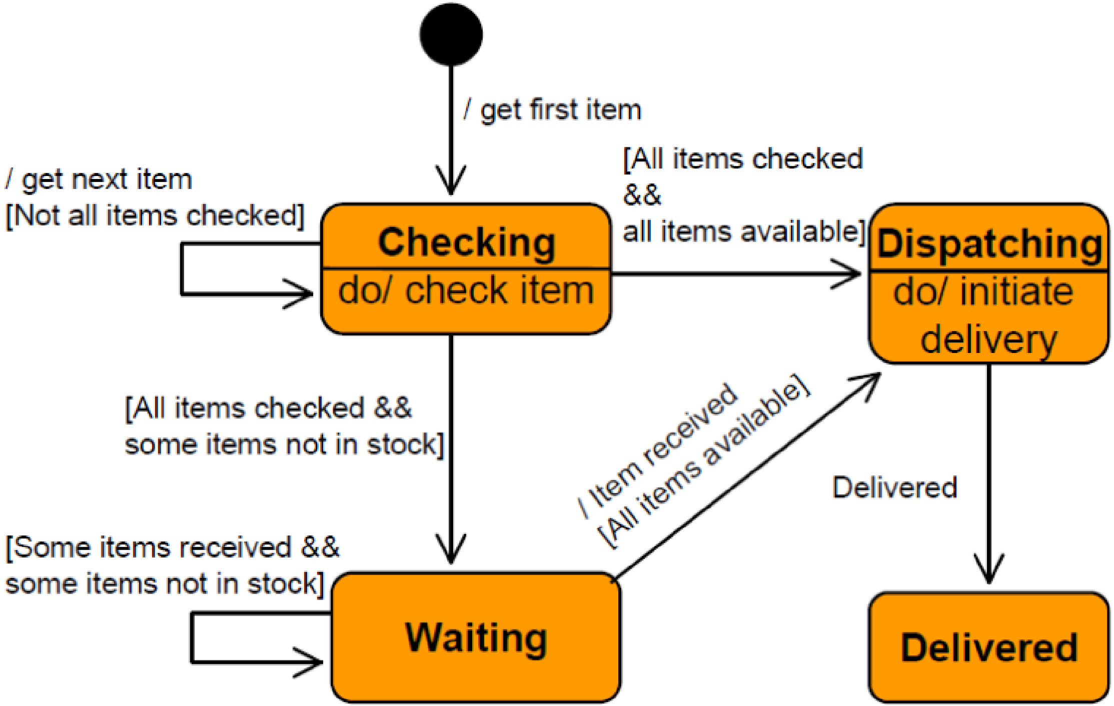


Activities

- Activities can take “longer”, i.e., they are processes which last as long as an object is in a certain state.
- Activities are interruptible, i.e., an event causing a state transition may abort an activity.
- Activities may be constructed from a start and a final action.

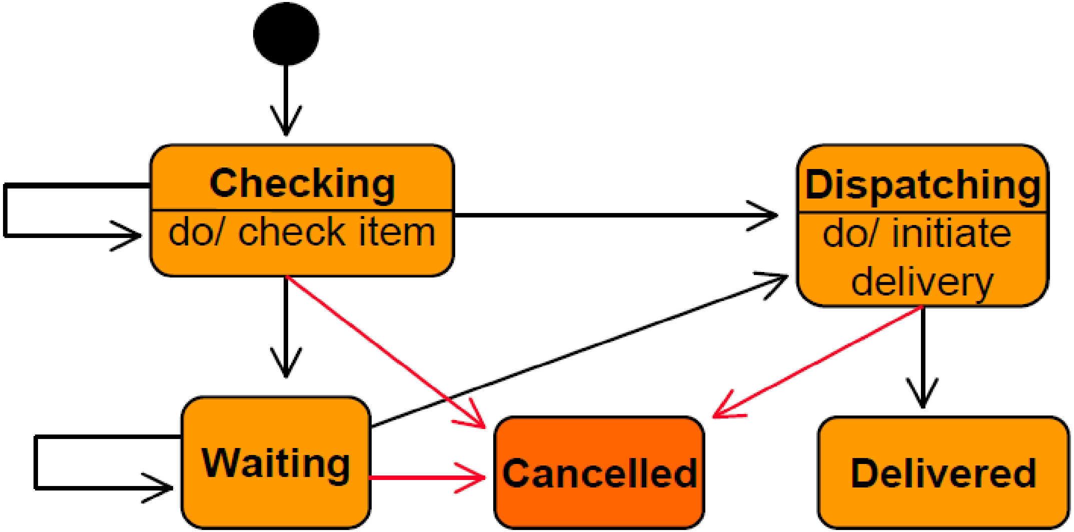


Example


/ get first item
/ get next item
[Not all items checked]
[All items checked
&&
all items available]
Checking
do/ check item
Dispatching
do/ initiate delivery
[All items checked &&
some items not in stock]
/ Item received
[All items available]
Delivered
[Some items received &&
some items not in stock]
Waiting
Delivered

Nesting

Example: A state Cancelled is added to which transitions from all existing states exist.

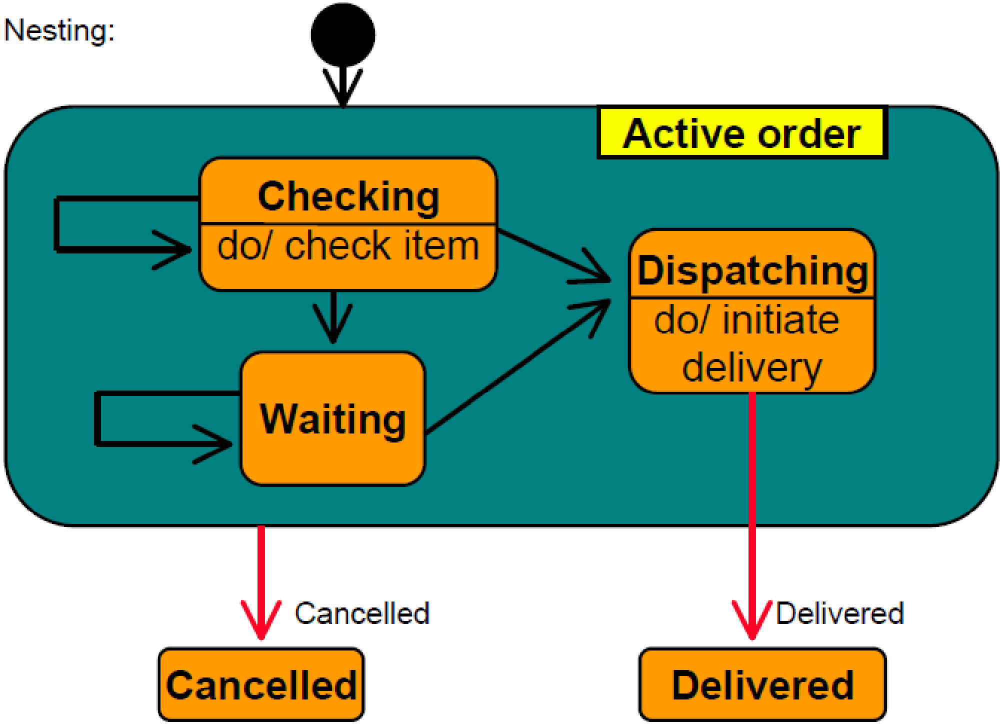


Nesting

- Superstates contain state diagrams or other superstates.
- Superstates allow to simplify multiple transitions from probably many source states to a single target state by
  - introducing a (superstate) name for a (nested) state diagram and
  - substituting each of the transitions between source states and the target state by a single transition between superstate and target state.

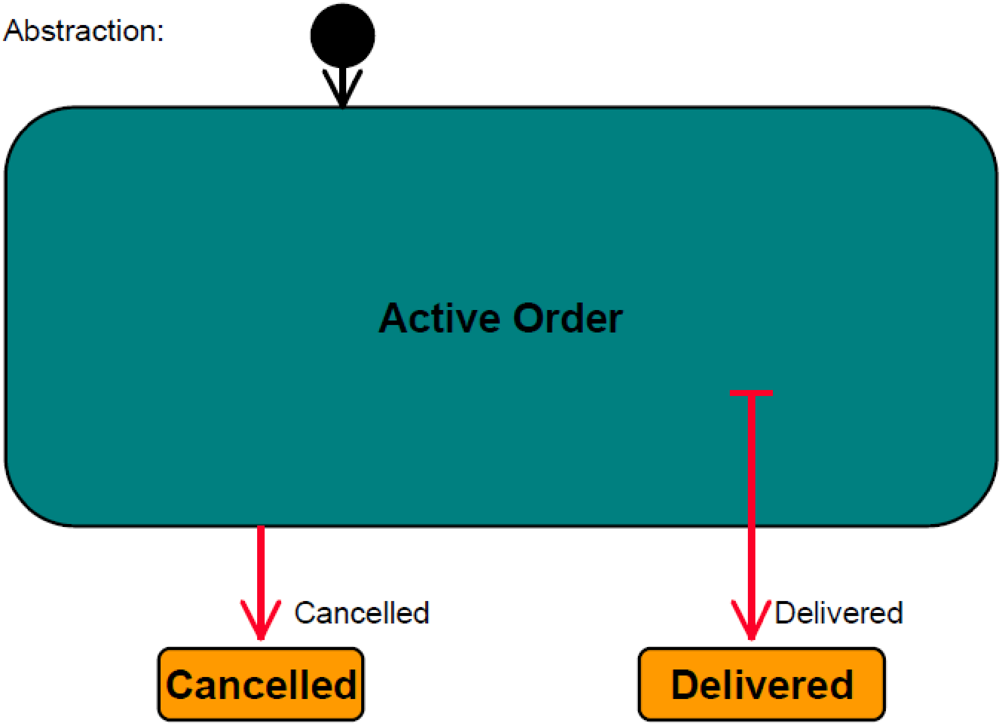
Superstates: Nesting

Nesting:



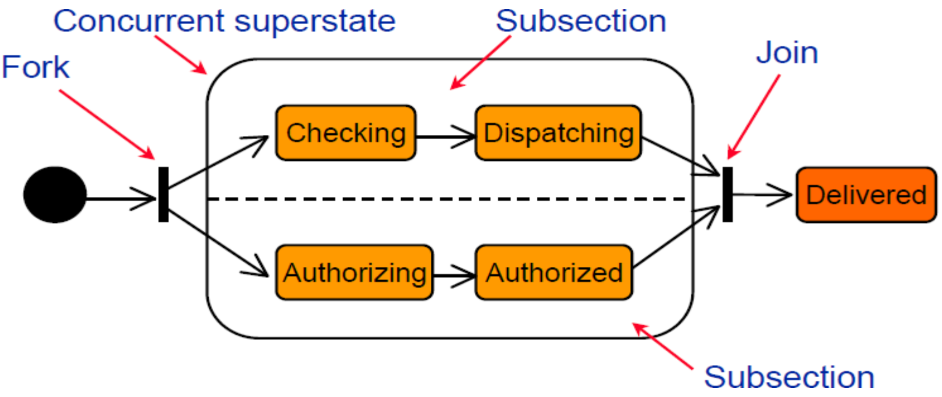
Superstates: Abstraction

Abstraction:



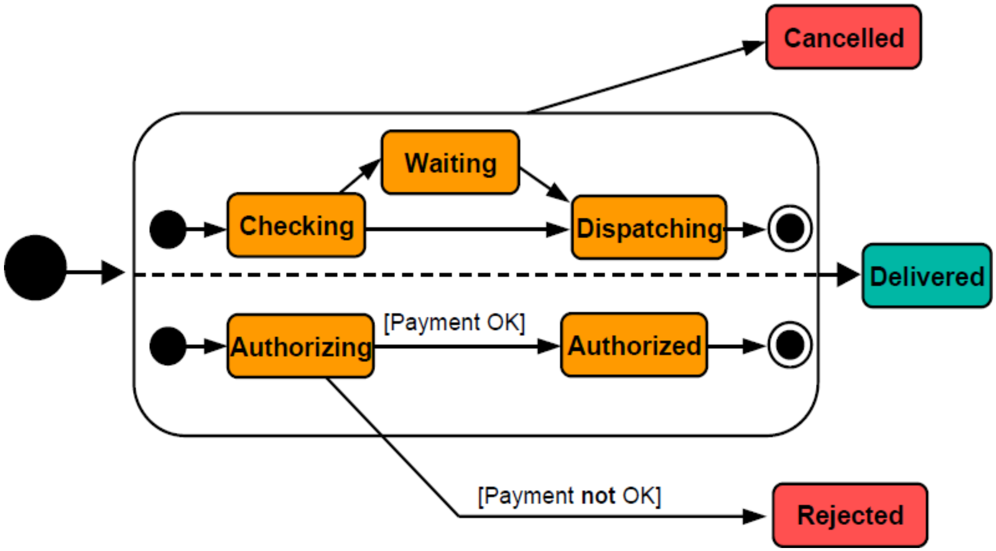
Concurrency in State Diagrams

- Concurrent state diagrams are useful when a given object has sets of independent behaviors.
- The concurrent sections of a state diagram are places in which, at any time, the given object is in a composite state defined by the given subsections.



Concurrency: Alternative Notation

Example: The authorization of a customer for a certain purchase is checked concurrently to the item dispatching actions.



When to use state diagrams?

- State diagrams are good at describing the behavior of an object across several use cases.
- Draw state diagrams especially for classes, which are not well understood and which need detailed description.
- If you have to describe several objects, which are involved in a single use case, use interaction diagrams.
- To show the general sequence for multiple use cases and multiple objects, use activity diagrams.
- State diagrams are not very good at describing behavior that involves a number of objects collaborating together.