Lecture 7
Statechart Diagram
aka: State Diagram, State Transition Diagram

Chapter 25
The Unified Modeling Language User Guide
SECOND EDITION
By Grady Booch, James Rumbaugh, Ivar Jacobson

Session 20-21
UML Weekend Crash Course
Thomas A. Pender

Introduction

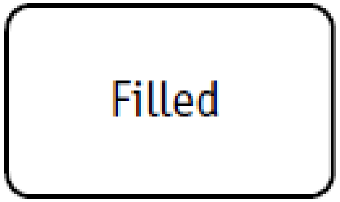
- **A State diagram addresses dynamic aspect of a system.** A state diagram shows a state machine to model the **lifetime** of an object.
- **A state diagram shows flow of control from state to state within a single object.**
- This involves modeling the behavior of reactive objects. A reactive object is one whose behavior is best characterized by its response to events.
- **The Statechart describes the life of an object in terms of the events that trigger changes in the object's *state.***
- **The *state* of the object is simply its current condition. That condition is reflected in the values of the attributes that describe that object.**
- There are behaviors in the system that alter those attribute values.

Elements

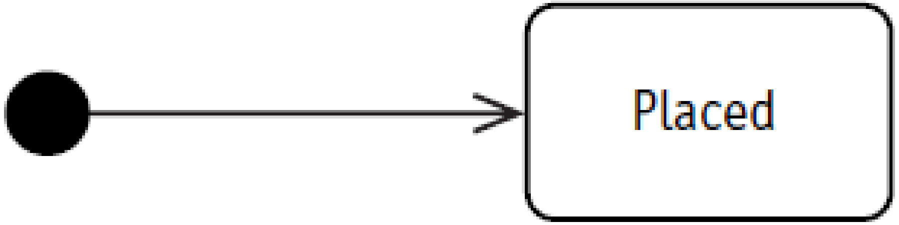
- State diagrams commonly contain:
  - State machine
  - Initial State
  - Transition
  - Event/ Action

State, Initial State, Transition

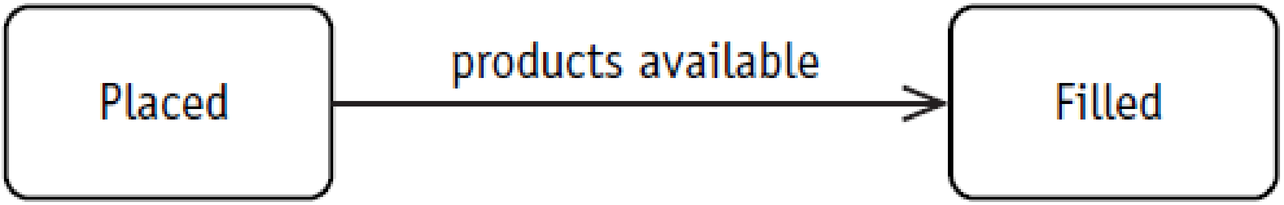
- State:



- Initial State: The initial state indicates the state in which an object is created or constructed.

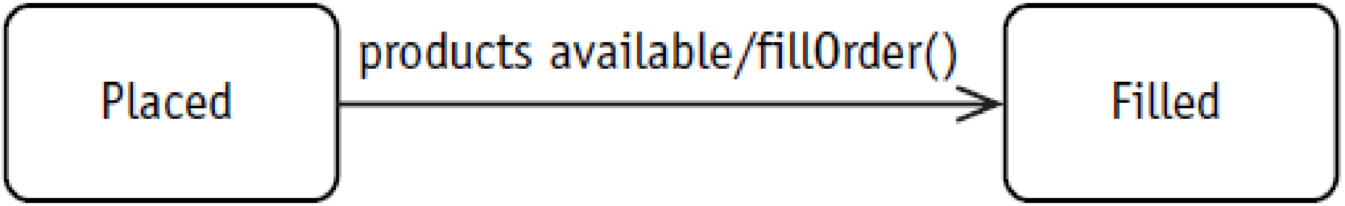


- Transition:



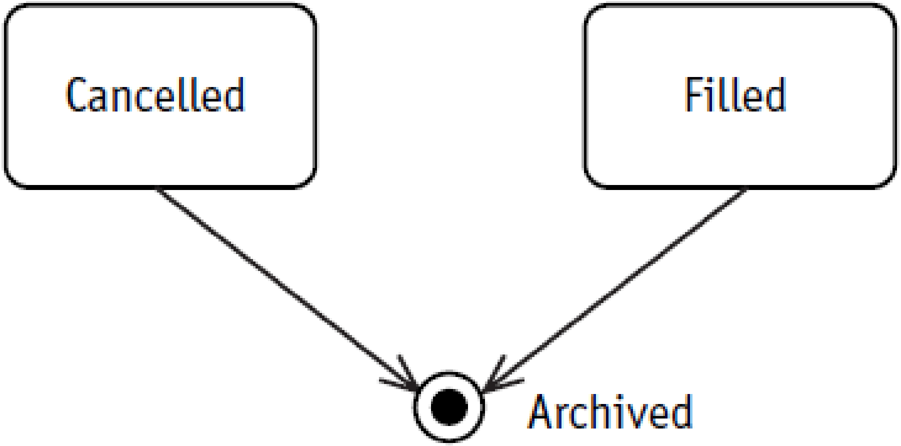
Event / Action

- Event: The reason of Transition
- Action: An action is associated with an event. An action is the behavior that is triggered by the event, and it is the behavior that actually changes the attributes that define the state of the object. **An action is an atomic task, and as such it cannot be broken into component tasks, nor can it be interrupted. There are no break points within it and, furthermore, stopping it midway would leave the object state undefined.**



Final State

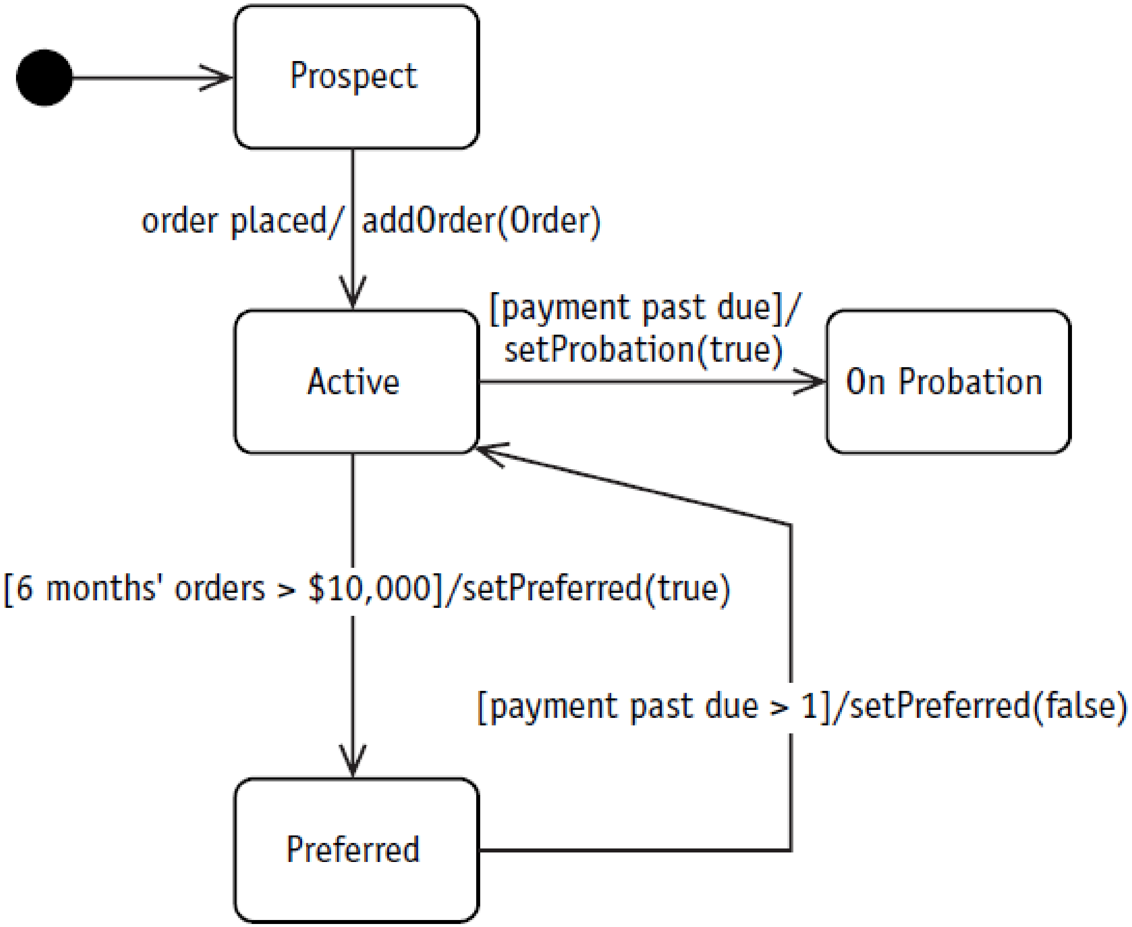
- An object may reach a *final* state from which it may not return to an active state. In other words, you would never see an arrow going out of this state. You may still see it and it may still exist, but you can no longer alter its state. The final state may also mean that the object has actually been deleted.



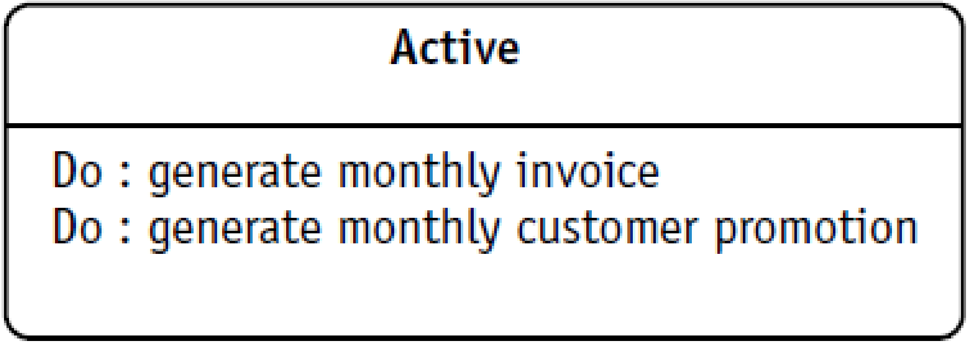
Example – Problem Statement (Page 206)

Case Study:

We track current customer status to help avoid uncollectable receivables and identify customers worthy of preferred treatment. All customers are initially set up as prospects, but when they place their first order, they are considered to be active. If a customer doesn't pay an invoice on time, he is placed on probation. If he does pay on time and has ordered more than $10,000 in the previous six months, he warrants preferred status. Preferred status may be changed only if the customer is late on two or more payments. Then he returns to active status rather than probation, giving him the benefit of the doubt based on his preferred history.


Prospect
order placed/ addOrder(Order)
Active
[payment past due]/ setProbation(true)
On Probation
[6 months' orders > $10,000]/setPreferred(true)
[payment past due > 1]/setPreferred(false)
Preferred

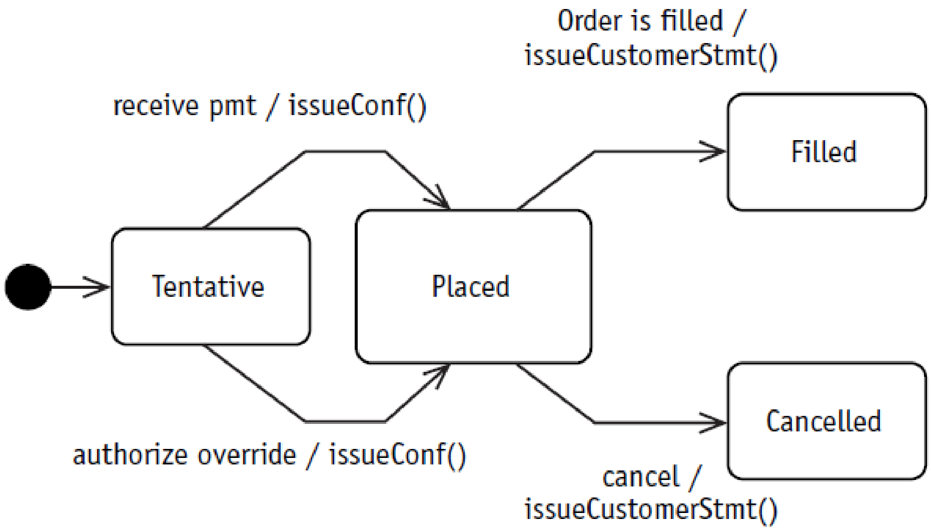
Internal Events/ Do Activity

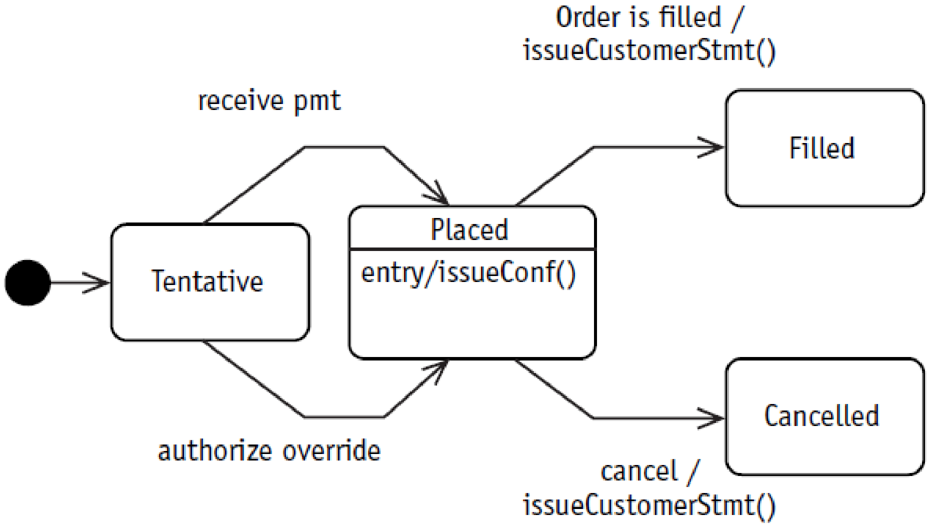


Active

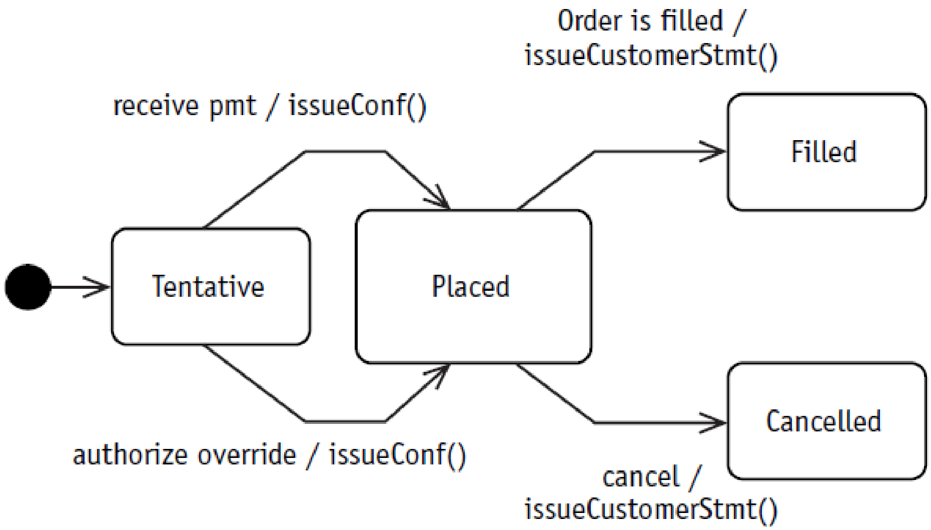
Do : generate monthly invoice
Do : generate monthly customer promotion

Entry Actions





Exit Actions



Order is filled / issueCustomerStmt()

receive pmt / issueConf()

Filled

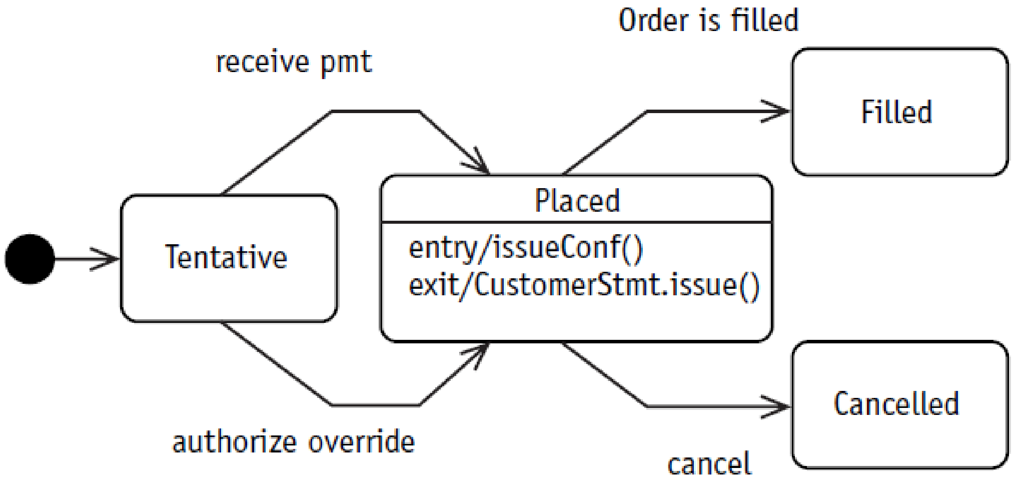
Tentative

Placed

Cancelled

authorize override / issueConf()

cancel / issueCustomerStmt()



Order is filled

receive pmt

Filled

Tentative

Placed

entry/issueConf()
exit/CustomerStmt.issue()

Cancelled

authorize override

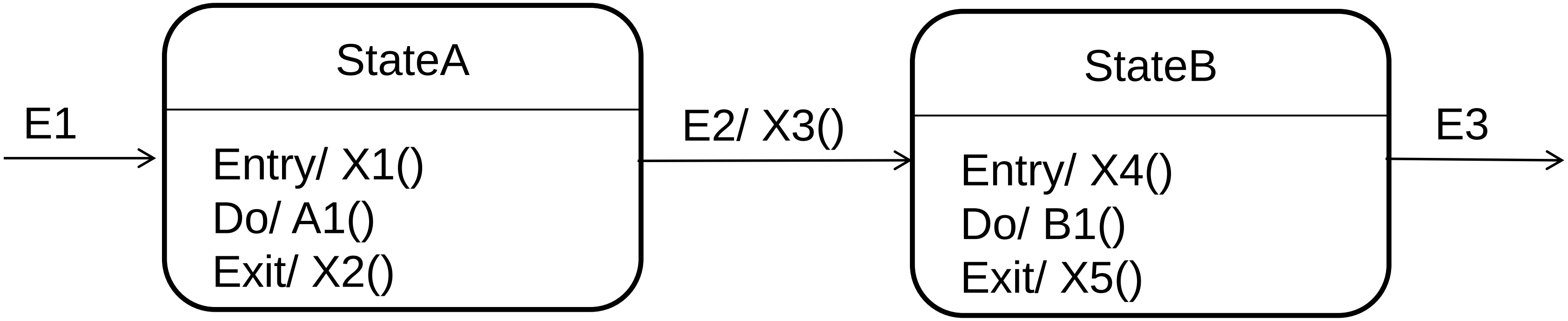
cancel

Do, Entry, Exit

- **Do Activity:** The regular operation(s) of the object while in a state
- **Entry Action:** Common to all incoming transition to a state. Must be completed while entering the state.
- **Exit Action:** Common to all outgoing transition from a state. Must be completed while exiting the state.

Order of Events (Example)

- When an event **(E2)** occurs, the order of execution runs like this:
  1. If an activity is in progress in the current state (**A1**), interrupt it (gracefully if possible).
  2. Execute the exit: action(s) – **X2**
  3. Execute the actions associated with the event that started it all – **X3**
  4. Execute the entry: action(s) of the new state – **X4**
  5. Execute the activity or activities of the new state – **B1**

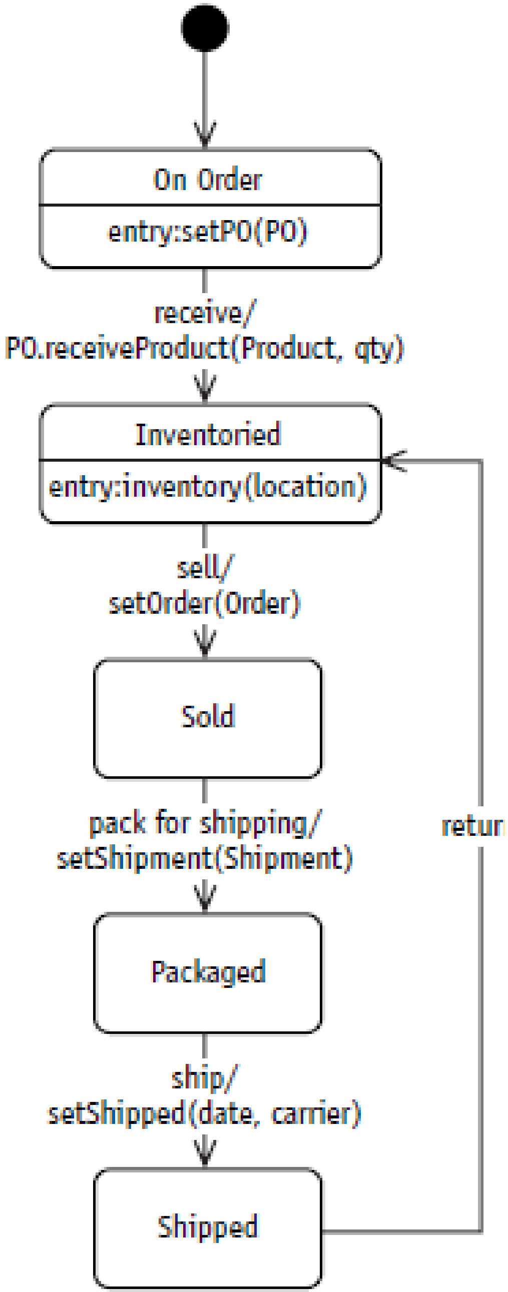


Problem Statement (Page 221)

Inventory control:

Products are first entered into our system when they are ordered using a purchase order (P.O.). Each product keeps a record of the originating P.O. When the product is received, it is placed into inventory by recording the location where it is placed. When the product is received, you have to update the P.O. to indicate that you have received the product.

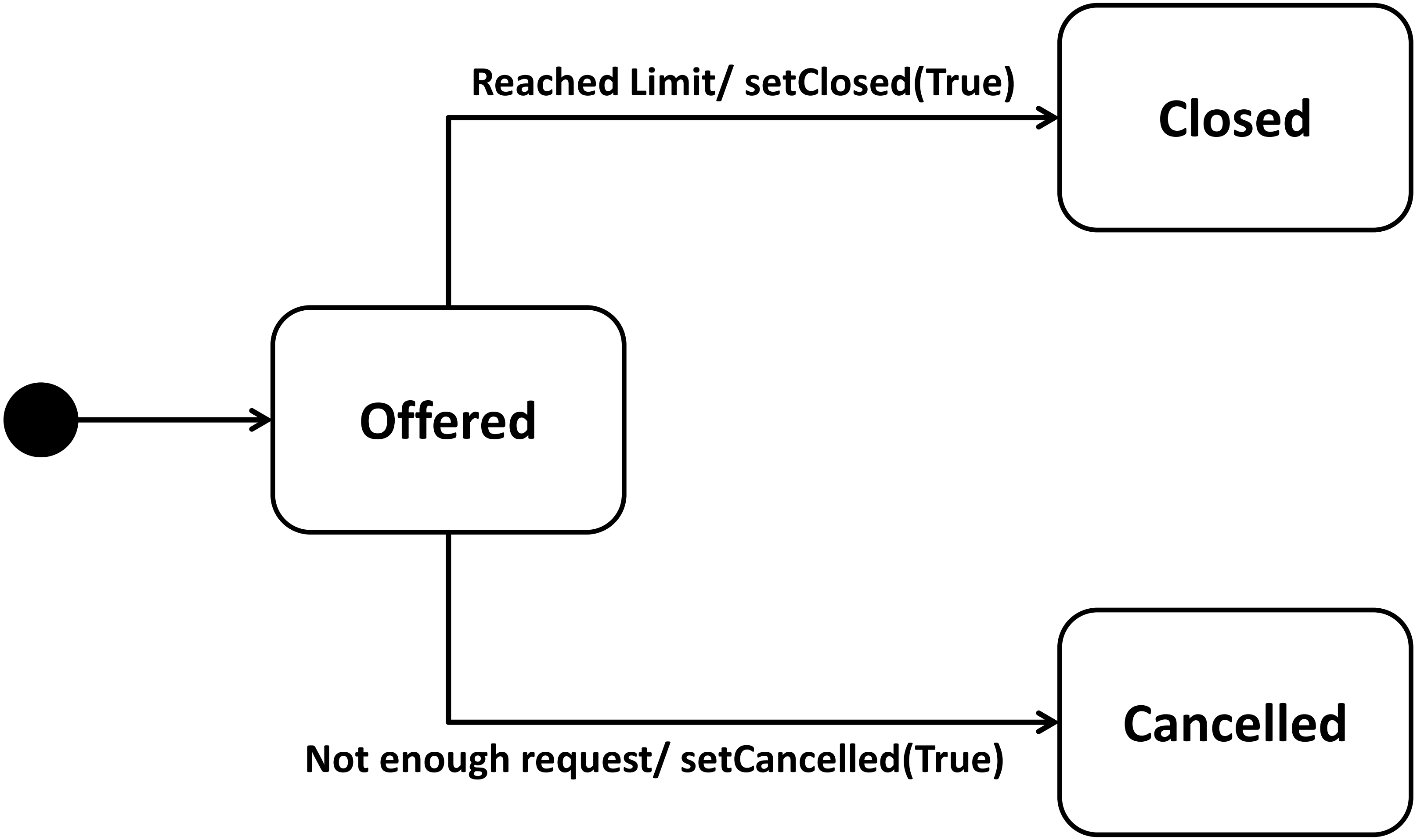
When a product is sold, the product tracks the order to which it belongs. When a product is sold, it is also packed for shipping and the associated shipment is recorded. When the product is shipped, you need to record the shipper and the date it was picked up. Occasionally, a product is returned. In that case, you put the product back into inventory and record the location.



Case Studies

- **Case 1: Statechart diagram for 'COURSE'**
- A course is initially created in offered state. Offered courses are then included by the students in their requests and the requests are placed for registration. Student requests are then checked for pre-requisite completion for some of the courses. If the pre-requisites are completed students are added in the courses. When a course reaches the limit of its capacity the course is then closed for further student requests. On the other hand if there are not enough students requesting for a course, the course is cancelled.

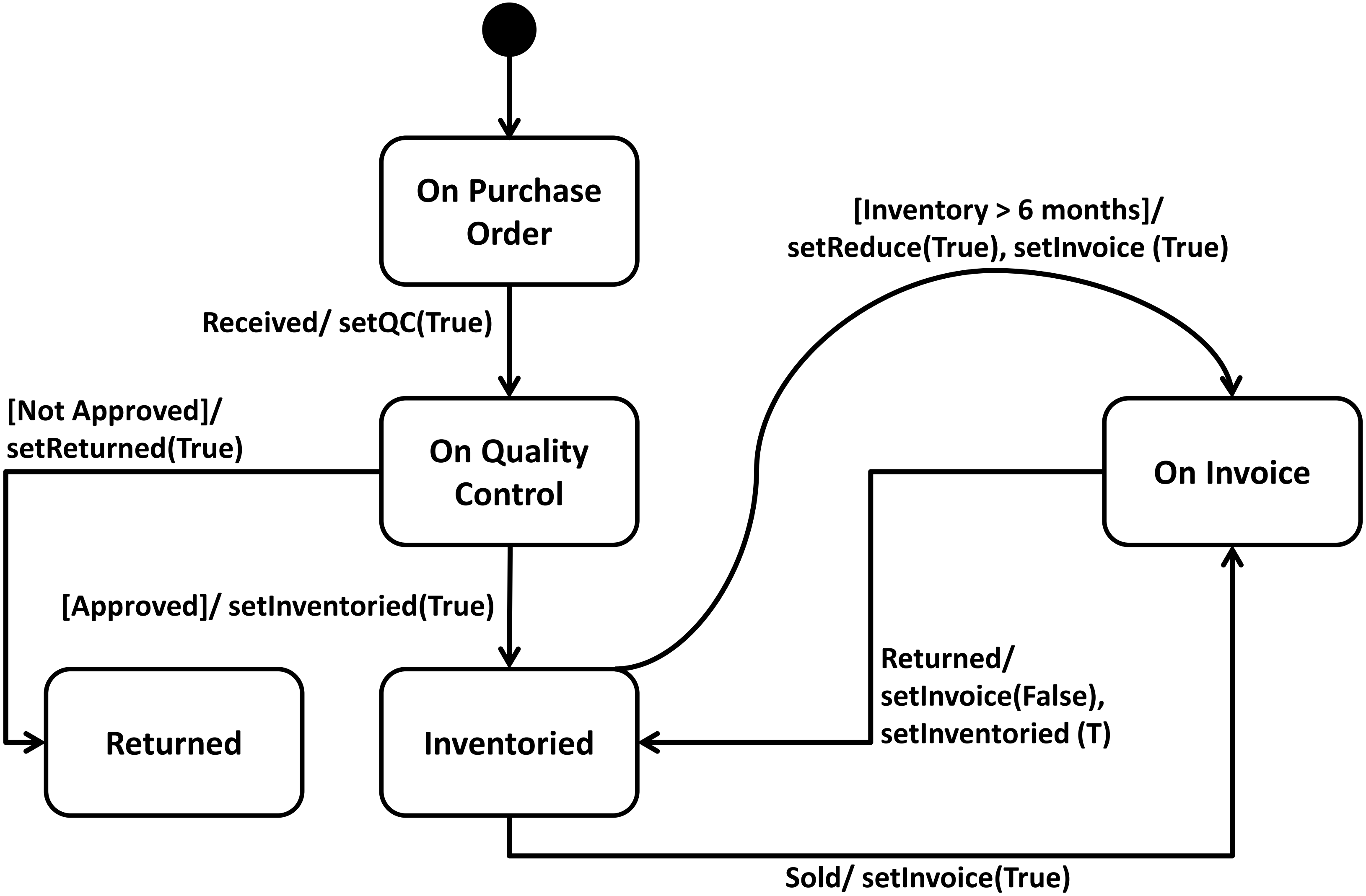
Case 1: Solution – ‘COURSE’ object


Reached Limit/ setClosed(True)
Closed
Offered
Cancelled
Not enough request/ setCancelled(True)

Case Studies

- **Case 2: Statechart diagram for ‘PRODUCT’**
- A Product is first entered into the system when a purchase order for that product is issued. When a purchase order generated for the product and it is recorded with that associated product. The purchase order is then sent to the vendor. When the product is received from the vendor, it is examined by the quality control department. If the product is approved by the quality control it is placed into inventory by recording the location where it is placed. If it is not approved the product is returned to the vendor. When a customer buys a product it is included in an invoice. Sometimes, a product is returned. In that case, you put the product back into inventory and record the location. If a product is lying in the inventory for more than six months it is returned to the vendor, but this time the vendor buys back the product in a reduced price, which means the product is included in an invoice.

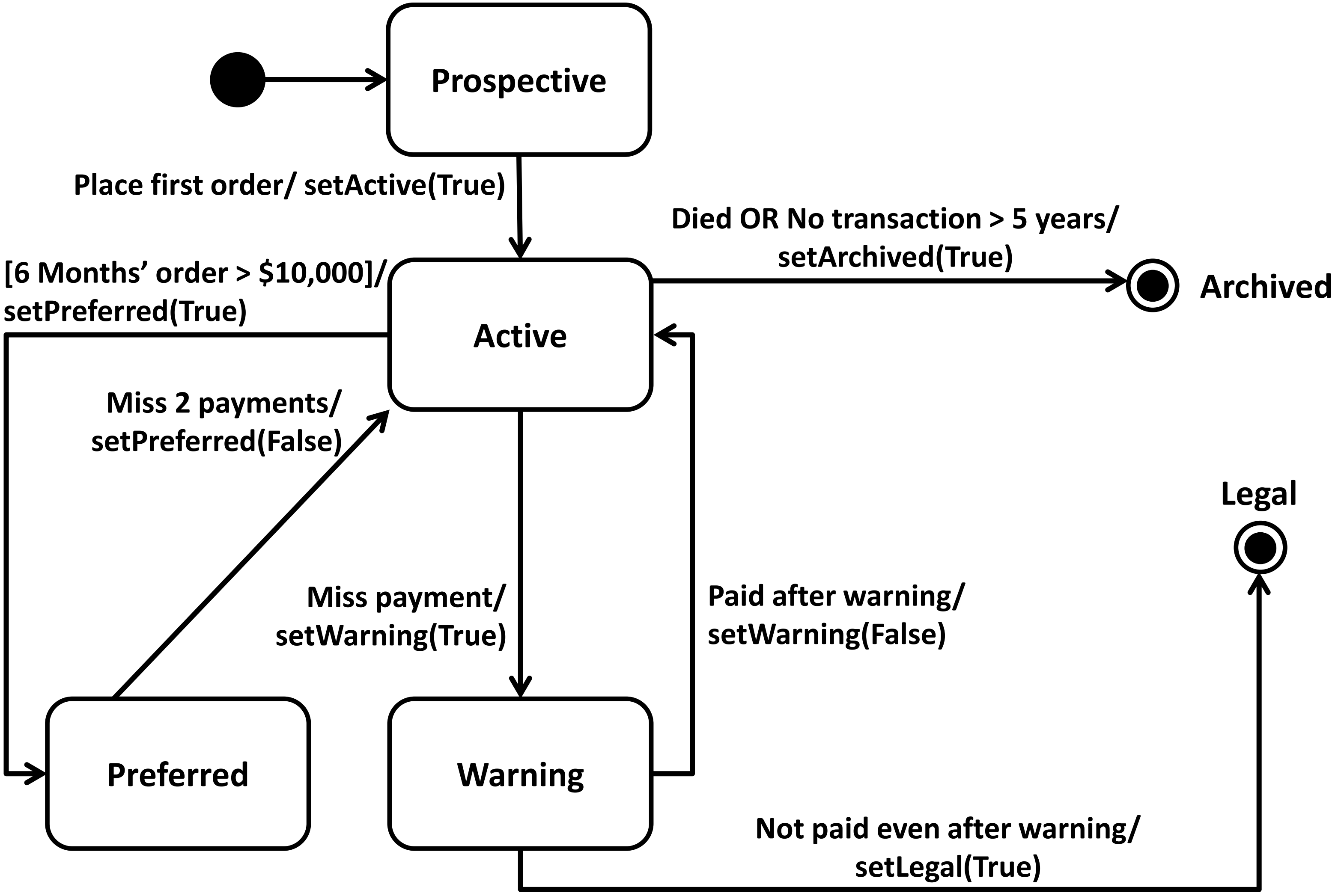
Case 2: Solution – ‘PRODUCT’ object


On Purchase Order
Received/ setQC(True)
On Quality Control
[Not Approved]/ setReturned(True)
[Approved]/ setInventoried(True)
Returned
Inventoried
[Inventory > 6 months]/ setReduce(True), setInvoice (True)
On Invoice
Returned/ setInvoice(False), setInventoried (T)
Sold/ setInvoice(True)

Case Studies

- **Case 3: Statechart diagram for 'CUSTOMER'**
- In an order processing system all customers are initially set up as **prospective** customers, but when they place their first order, they are considered to be **active**. If a customer doesn't pay an invoice on time, he is given **warning**. If the customer doesn't pay the dues within a month of the warning, **legal** actions are taken against the customer. Whatever the legal action is, these customers are never taken back in the system. If the customer pays after the warning, he is taken back into the system as a regular **active** customer. If an active customer does pay on time and has ordered more than $10,000 in the previous six months, he warrants **preferred** status. Preferred status may be changed only if the customer is late on two consecutive payments. Then he returns to **active** status rather than warning, giving him the benefit of the doubt based on his preferred history. Active customer information is **archived** for good if the customer dies or if there is no transaction with the customer for more than five years.

Case 3: Solution – 'CUSTOMER' object



Case Studies

- **Case 4: Statechart diagram for ‘STUDENT’**
- A student object is created as prospective student when he submits the admission form to a university. Student information is verified against the admission requirements and if he meets all requirements, he is asked to sit for an admission test as an examinee. But if the student doesn’t meet the requirements or fails in the admission test, he is rejected, and the student object is deleted permanently from the system. Students who pass in the admission exam are called for viva as viva candidates. If the student passes the viva and get admitted, the student is in freshman status. After successful completion of the first semester, the student becomes regular. Each semester the student grade is evaluated and in case of low CGPA (less than 2.5), a regular student becomes a probation student. If the CGPA of a probation student doesn’t improve the studentship is cancelled from the university. On the other hand if the student CGPA improves to 2.5 or above, they become regular student again. High CGPA students are granted scholarships provided that they apply for it and fulfill the requirements. After successful completion of all courses a regular (or scholarship) student becomes a graduate student and all the information of a graduate student is archived for ever.