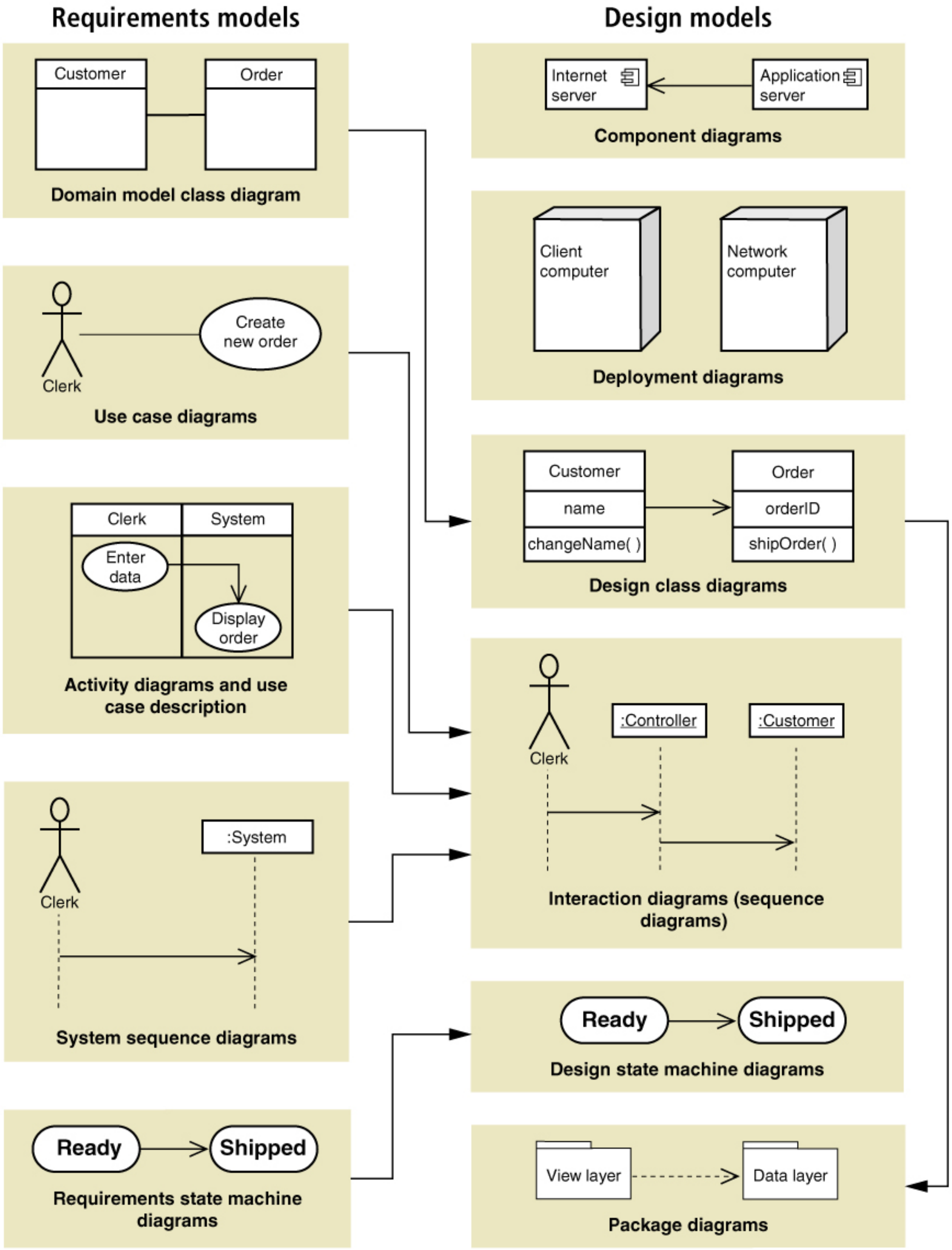
Agenda

- State Machine Diagram
- Sequence Diagram
- Detailed Class Diagram

From Analysis Models to Design Models



The State Machine Diagram

Object Behaviour and States

(Systems Analysis and Design in a Changing World, ©2016. Cengage Learning).

- Each class has objects that may have status conditions or “states”
- Object behavior consists of the various states and the movement between these states
- **State** – a condition during an object’s life when it satisfies some criterion, performs an action, or waits for an event
- **Transition** – the movement of an object from one state to another

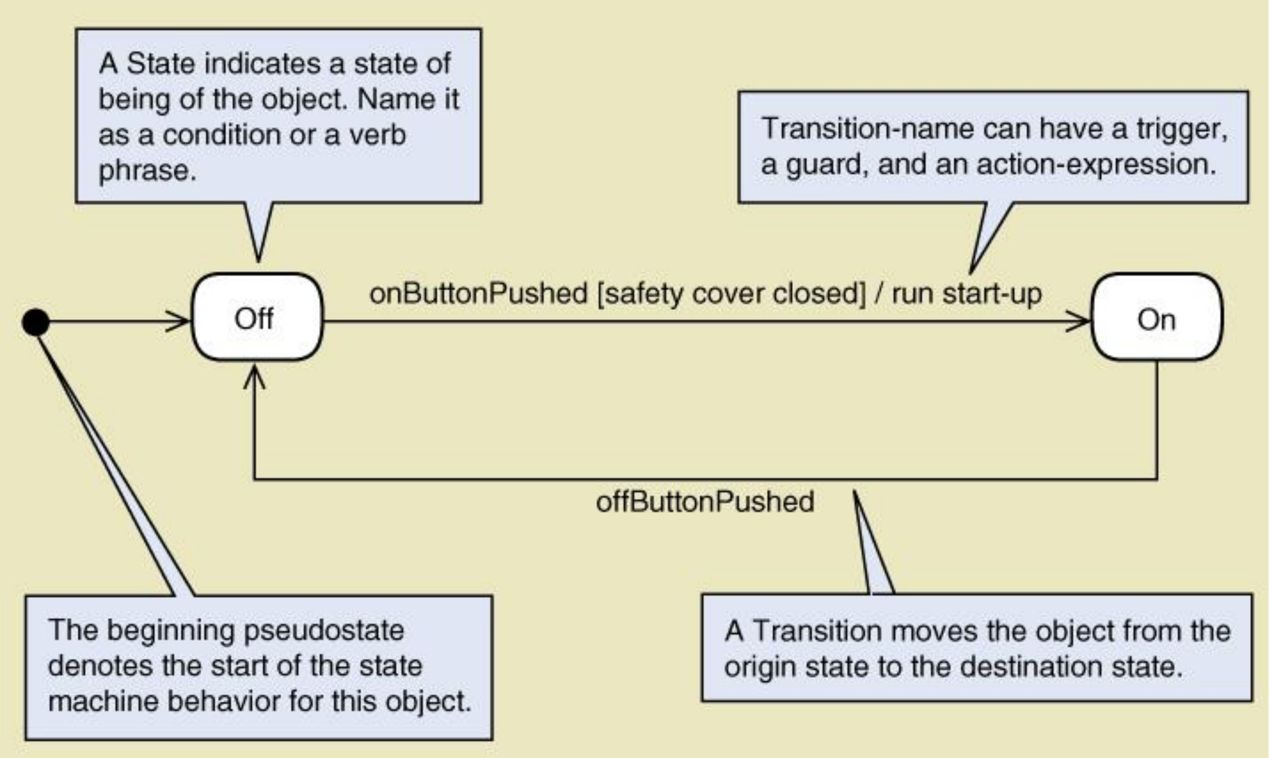
State Machine Diagram

(Systems Analysis and Design in a Changing World, ©2016. Cengage Learning).

- **State Machine Diagram** – a diagram which shows the life of an object in states and transitions
- **Origin state** – the original state of an object before it begins a transition
- **Destination state** – the state to which an object moves after completing a transition
- **pseudostate** – the starting point in a state machine diagram. Noted by a black circle.
- **action-expression** – some activity that must be completed as part of a transition
- **guard-condition** – a true/false test to see whether a transition can fire

State Machine for a Printer

(Systems Analysis and Design in a Changing World, ©2016. Cengage Learning).


A State indicates a state of being of the object. Name it as a condition or a verb phrase.
Transition-name can have a trigger, a guard, and an action-expression.
Off
onButtonPushed [safety cover closed] / run start-up
On
offButtonPushed
The beginning pseudostate denotes the start of the state machine behavior for this object.
A Transition moves the object from the origin state to the destination state.

- Syntax of transition statement
  - *transition-name (parameters, …) [guard-condition] / action-expression*

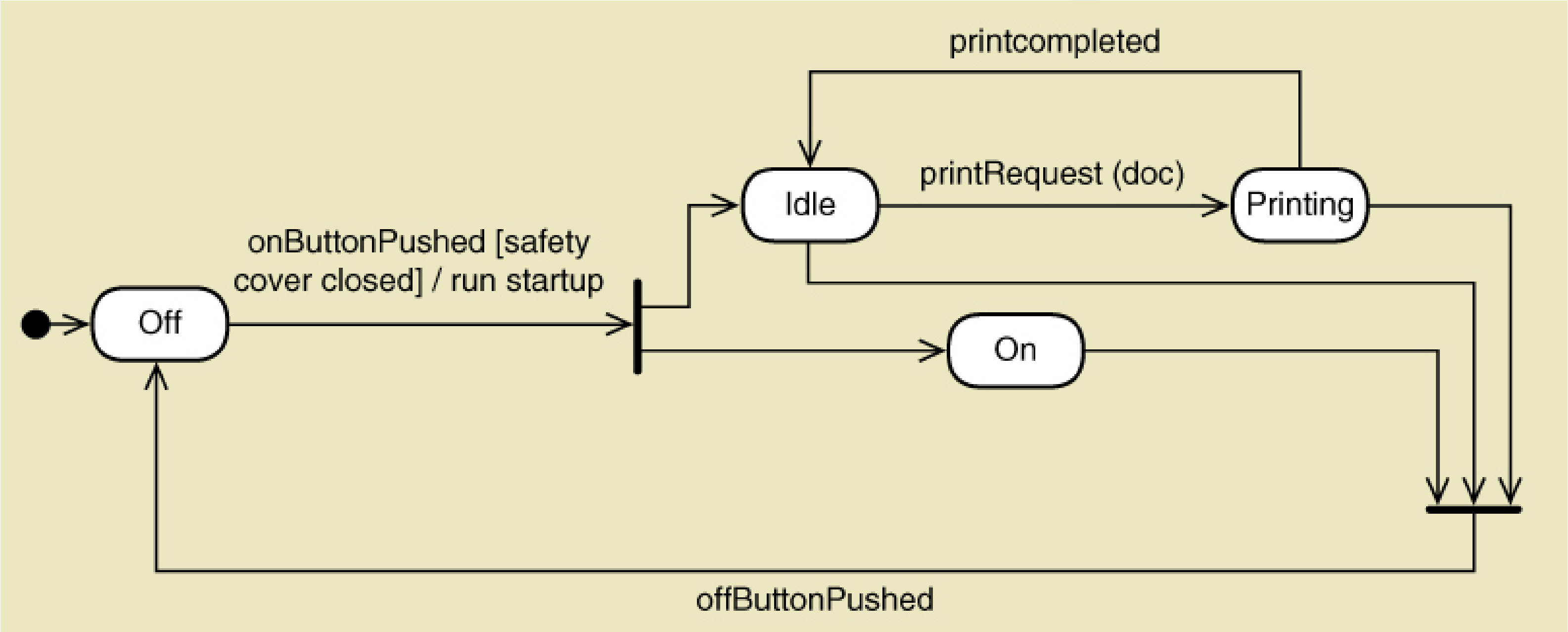
Concurrency in a State Machine Diagram

(Systems Analysis and Design in a Changing World, ©2016. Cengage Learning).

- **Concurrent states** – when an object is in one or more states at the same time
- **Path** – a sequential set of connected states and transitions
- **Concurrent paths** – when multiple paths are being followed concurrently, i.e. when one or more states in one path are parallel to states in another path

Printer with Concurrent Paths

(Systems Analysis and Design in a Changing World, ©2016. Cengage Learning).



- Concurrent paths often shown by synchronization bars (same as Activity Diagram)
- Multiple exits from a state is an "OR" condition.
- Multiple exits from a synchronization bar is an "AND" condition.

Steps in Creating a State Machine Diagram

(Systems Analysis and Design in a Changing World, ©2016. Cengage Learning).

1. Review the class diagram *(in our case, conceptual object model)* and select object classes that might require state machine diagrams
2. For each object class, make a list of status conditions (states) you can identify
3. Begin building diagram fragments by identifying transitions that cause an object to leave the identified state
4. Sequence these states in the correct order and aggregate combinations into larger fragments
5. Review paths and look for independent, concurrent paths

Steps in Creating a State Machine Diagram…

(Systems Analysis and Design in a Changing World, ©2016. Cengage Learning).

6. Look for additional transitions and test both directions
7. Expand each transition with appropriate message event, guard condition, and action expression
8. Review and test the state machine diagram for the object class
   - Make sure state are really state for the object in the class
   - Follow the life cycle of an object coming into existence and being deleted
   - Be sure the diagram covers all exception condition
   - Look again for concurrent paths and composite states

Case Study:

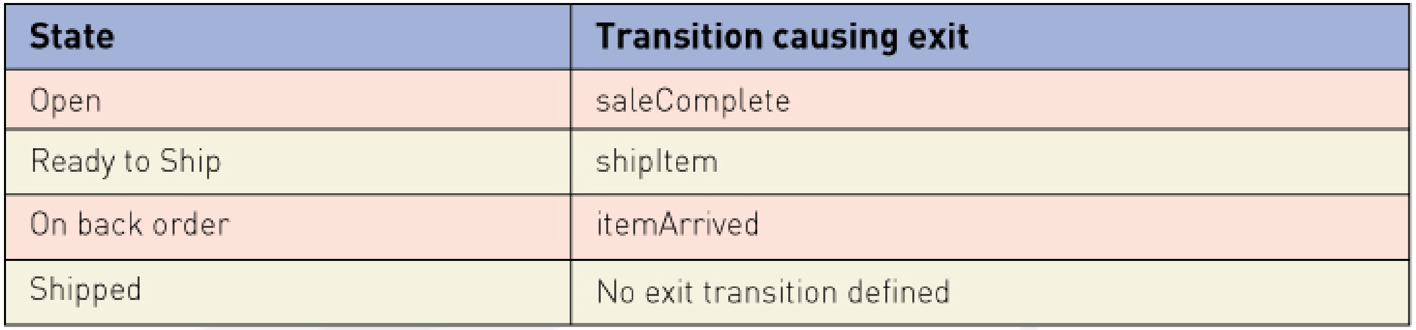
RMO CSMS Project

(State Machine Diagram)

RMO – Creating a State Machine Diagram

Steps -- SaleItem

1. Choose SaleItem. It has status conditions that need to be tracked
2. List the states and exit transitions

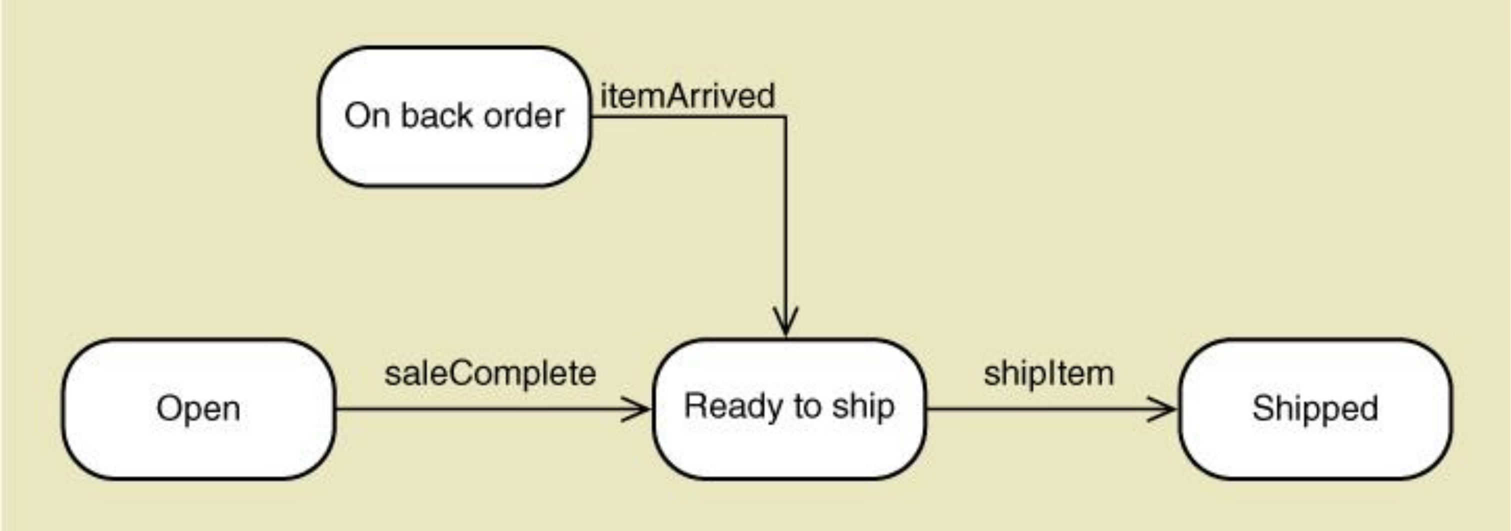


State	Transition causing exit
Open	saleComplete
Ready to Ship	shipItem
On back order	itemArrived
Shipped	No exit transition defined

RMO – Creating a State Machine Diagram

Steps -- SaleItem

3. Build fragments – see figure below
4. Sequence in correct order – see figure below
5. Look for concurrent paths – none

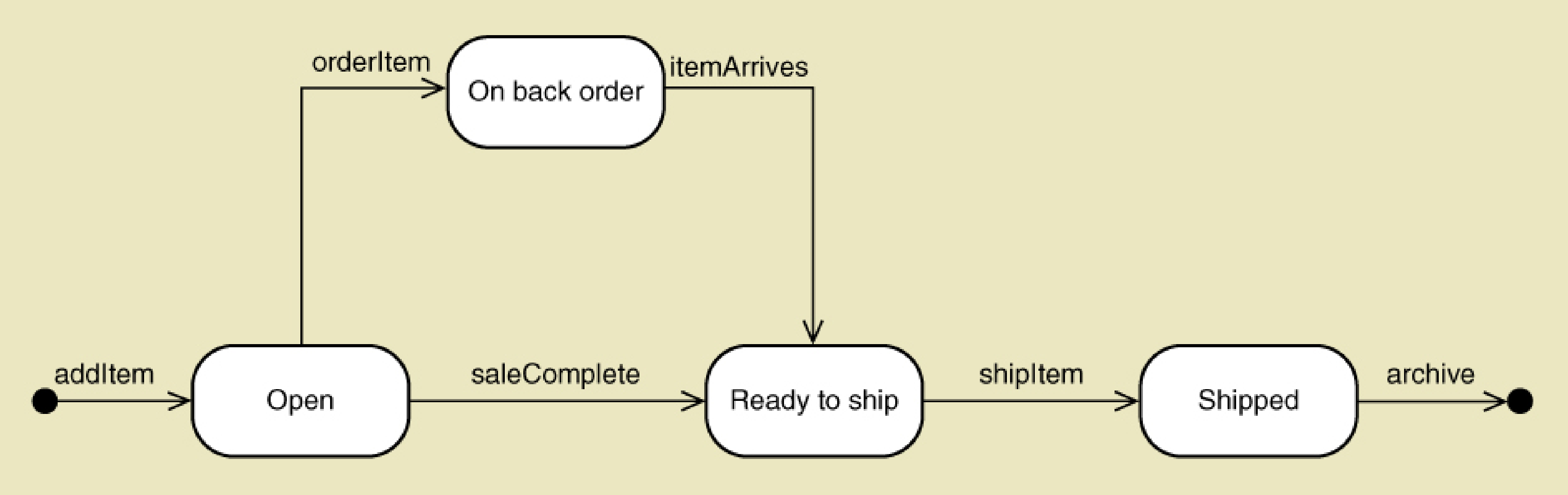


RMO – Creating a State Machine Diagram

Steps -- SaleItem

6. Add other required transitions
7. Expand with guard, action-expressions etc.
8. Review and test

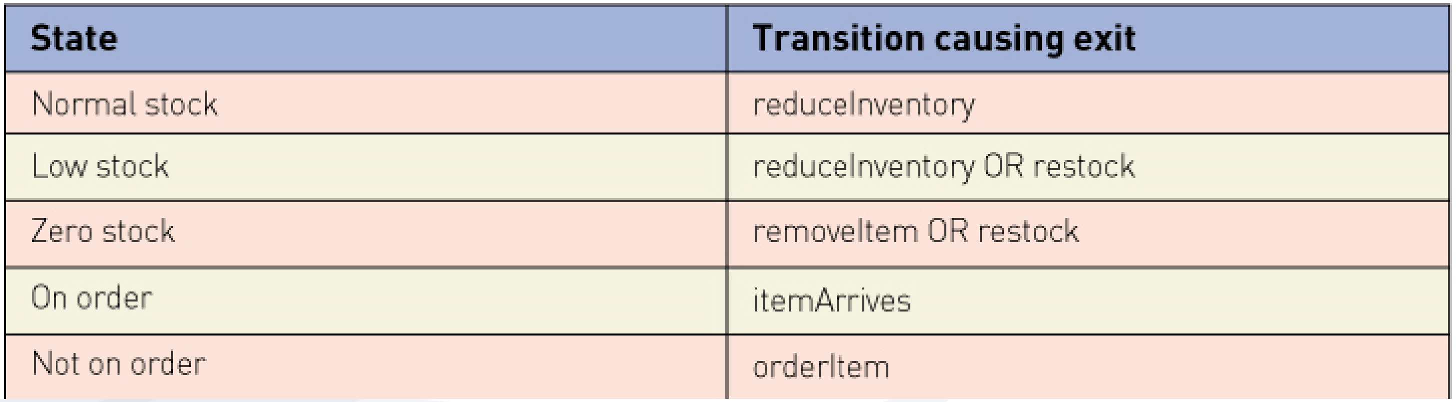
Below is the final State Machine Diagram



RMO – Creating a State Machine Diagram

Steps -- InventoryItem

1. Choose InventoryItem. It has status conditions that need to be tracked
2. List the states and exit transitions

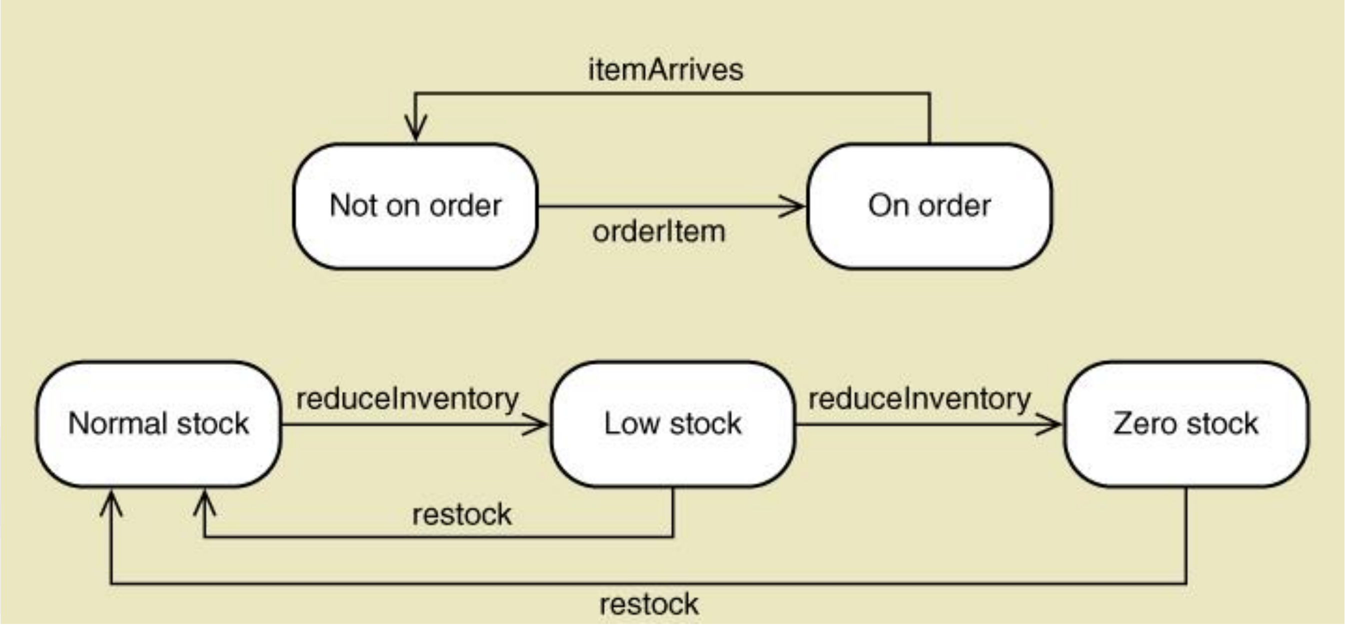


State	Transition causing exit
Normal stock	reduceInventory
Low stock	reduceInventory OR restock
Zero stock	removeItem OR restock
On order	itemArrives
Not on order	orderItem

RMO – Creating a State Machine Diagram

Steps -- InventoryItem

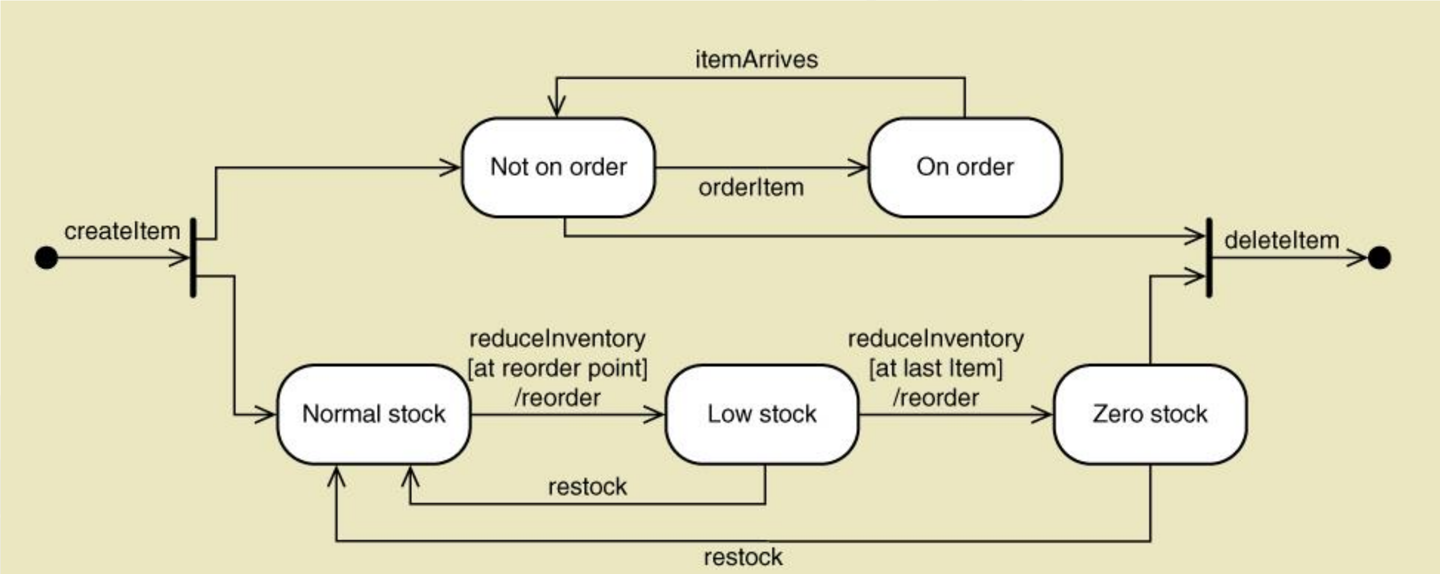
3. Build fragments – see figure below
4. Sequence in correct order – see figure below
5. Look for concurrent paths – see figure below



RMO – Creating a State Machine Diagram
Steps -- InventoryItem

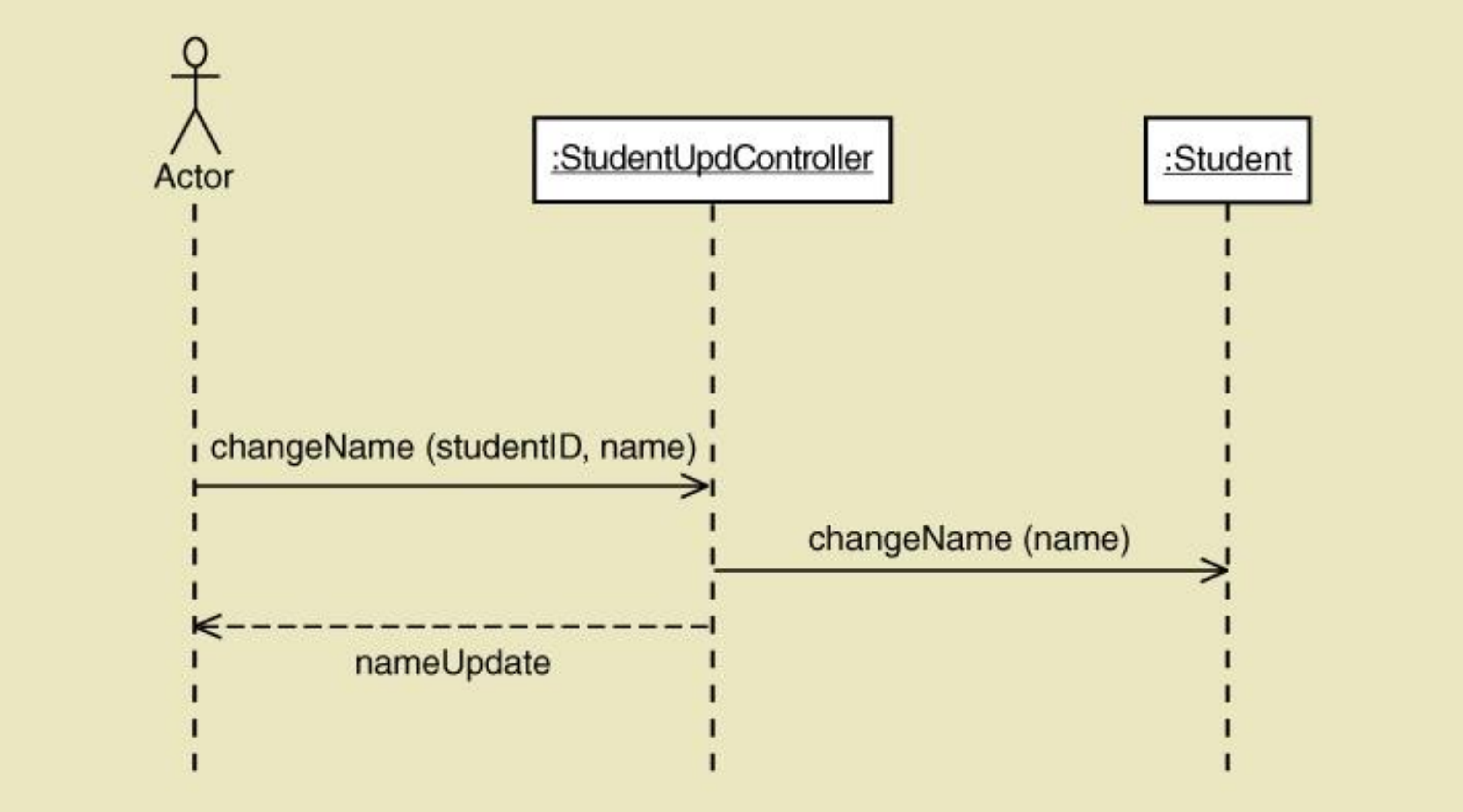
6. Add other required transitions
7. Expand with guard, action-expressions etc.
8. Review and test

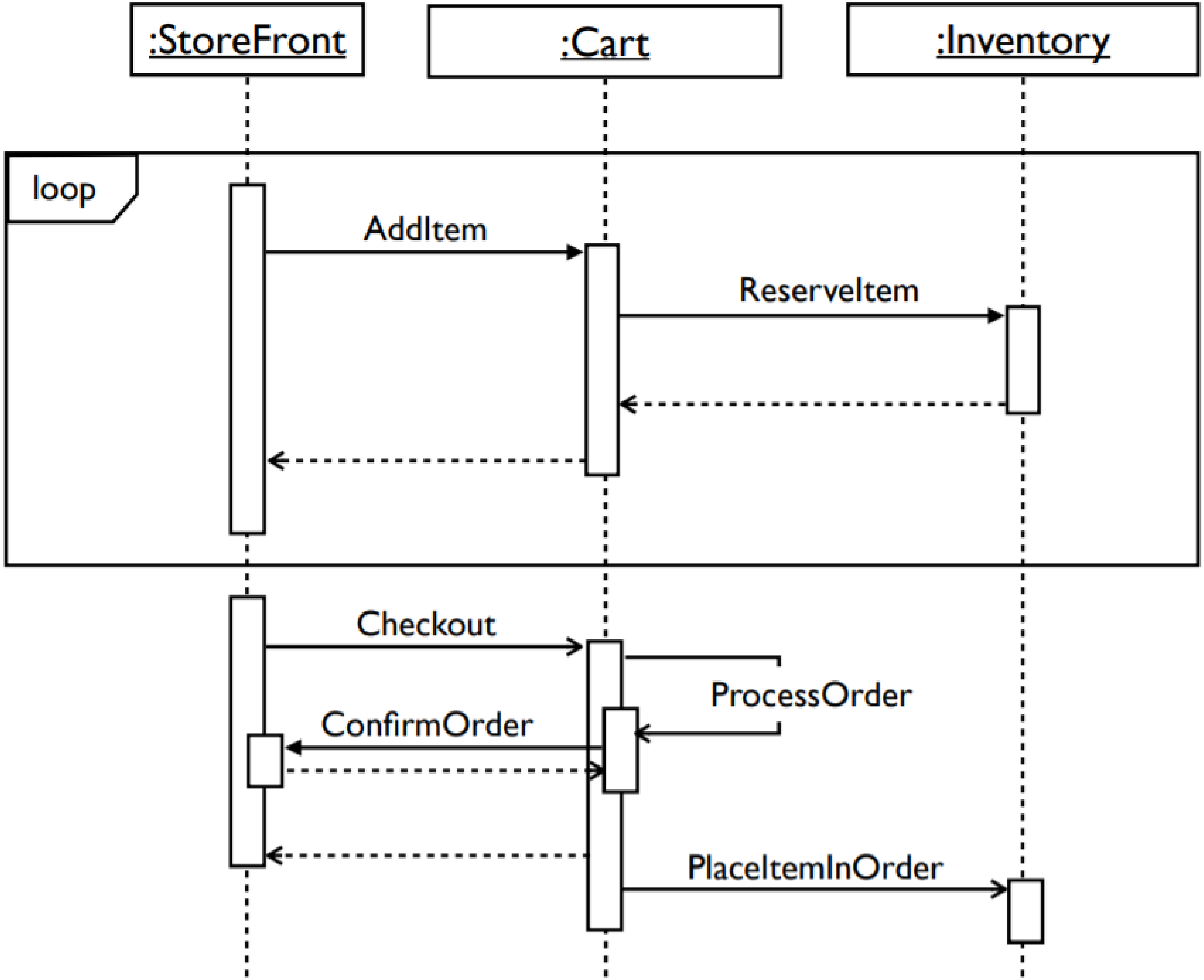
Below is the final State Machine Diagram



Sequence Diagrams

Student record update -Sequence Diag.




:StoreFront
:Cart
:Inventory
loop
AddItem
ReserveItem
Checkout
ProcessOrder
ConfirmOrder
PlaceItemInOrder

Detailed Class Diagram

Key Steps

- Finding entities in your system
  - These entities will become classes
- Assigning responsibilities to these classes
  - Often become operations that a class support
- Identifying collaborations between classes
  - Represent relationships between classes
  - Often describe dependence

Relation to Previous Phases

Analysis: Domain Model

- Identify conceptual classes
- Identify attributes of conceptual classes
- Identify association between conceptual classes

Design: Class Diagrams

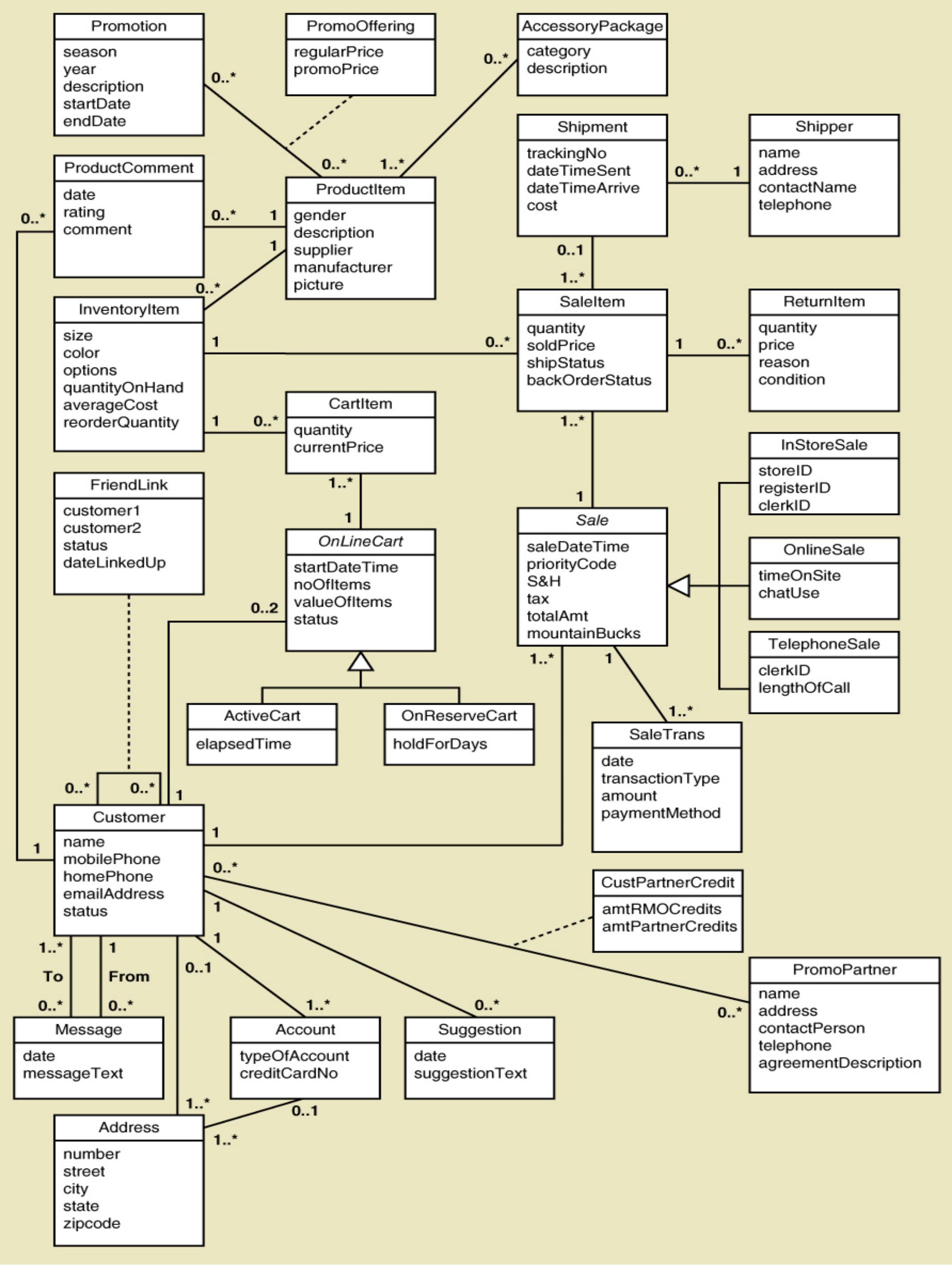
- Identify solution classes
- Identify responsibilities of solution classes
- Identify collaboration between solution classes

Finding Solution Classes

- Conceptual Classes in Domain Model
  - Not every conceptual class may be required
    - Example: Back Order, a conceptual class, but may not be a solution class
  - Some conceptual classes may become attributes
    - Example: Manufacturer, a conceptual class, but may just be an attribute of product item

Example: Domain Model Class Diagram

Classes with no responsibilities (methods)



CRC Card Method

- Although advocates of the object paradigm often say that identifying objects is a simple and intuitive process, a number of noted experts admit that this is not always true! …
- The solution is to use the CRC process to determine the classes necessary to the system as part of the design process for the application.
- CRC (classes, responsibility, and collaboration) cards can be used to visualize and test different class-based models during the design phase.
- It is a proven technique used and advocated by leading methodologists.

CRC Card

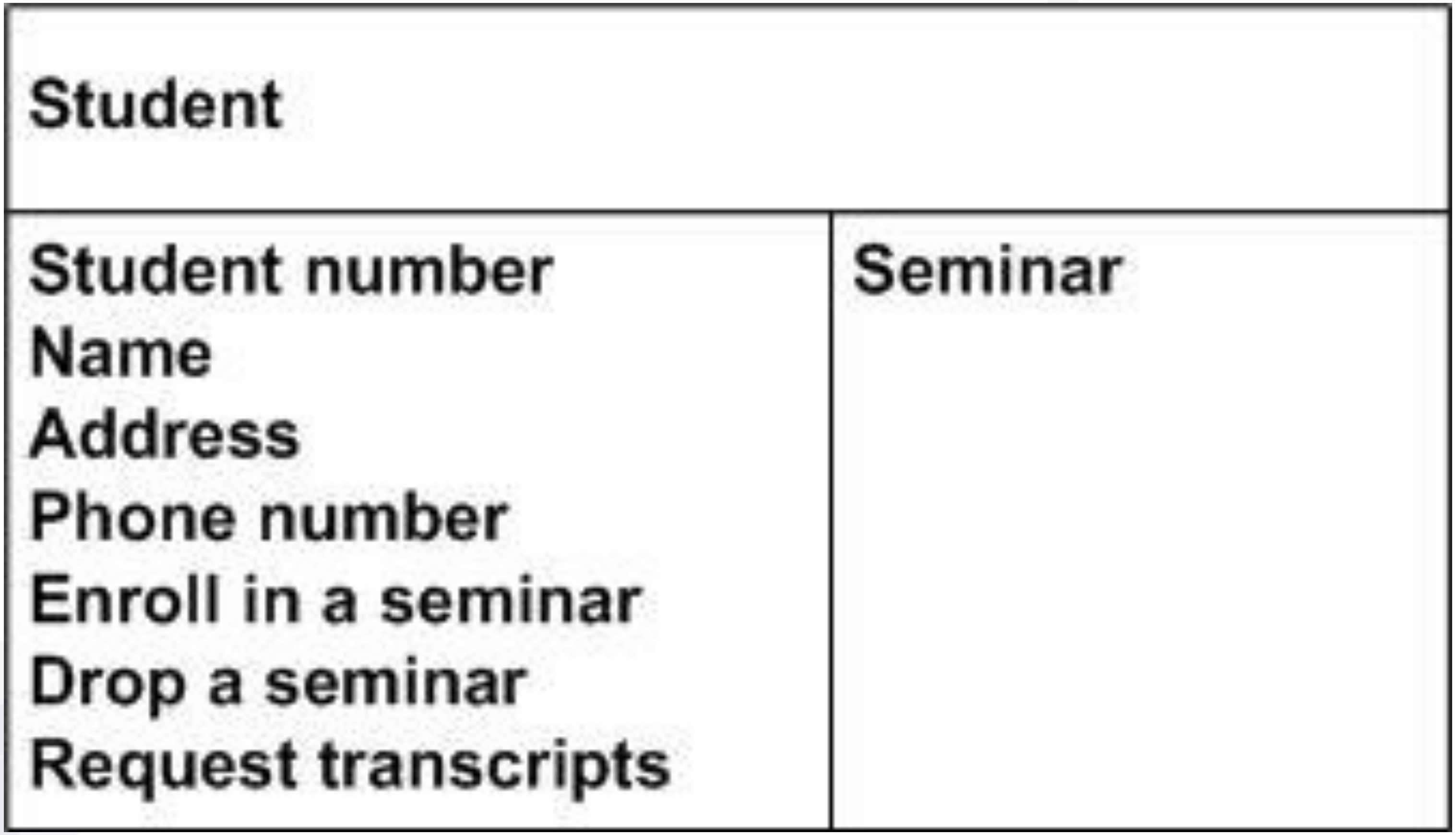
Class: Abstract/Concrete

Superclass(es):

Subclasses(es):

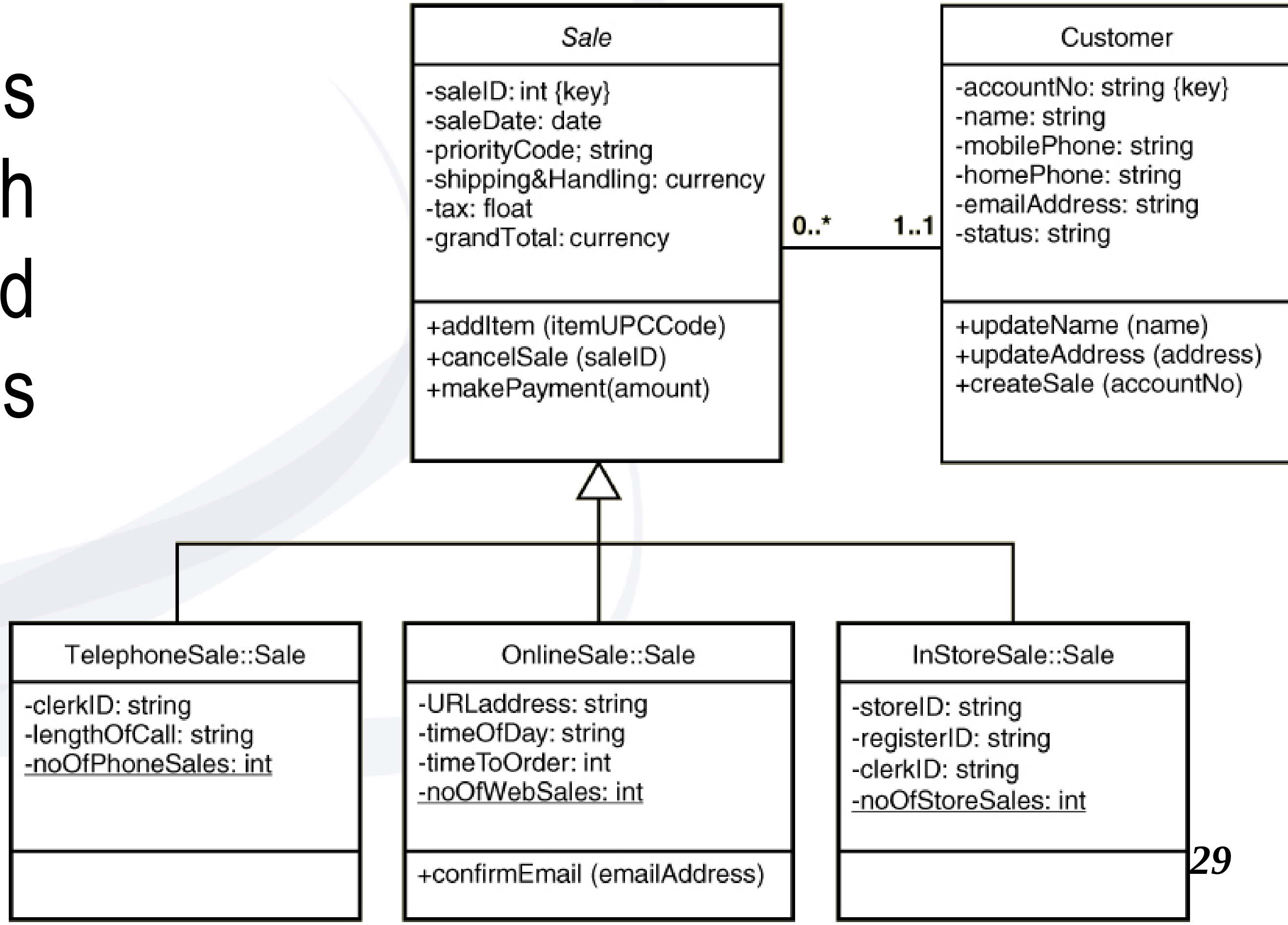
Responsibilities	Collaborators
(what class does or knows)	*(which classes help it perform each responsibility)*

Student CRC card



Student	
Student number Name Address Phone number Enroll in a seminar Drop a seminar Request transcripts	Seminar

Detailed Class Diagram with attributes and methods



Sale

-saleID: int {key}
-saleDate: date
-priorityCode; string
-shipping&Handling: currency
-tax: float
-grandTotal: currency

+addItem (itemUPCCode)
+cancelSale (saleID)
+makePayment(amount)

0..*

1..1

Customer

-accountNo: string {key}
-name: string
-mobilePhone: string
-homePhone: string
-emailAddress: string
-status: string

+updateName (name)
+updateAddress (address)
+createSale (accountNo)

TelephoneSale::Sale

-clerkID: string
-lengthOfCall: string
-noOfPhoneSales: int

OnlineSale::Sale

-URLaddress: string
-timeOfDay: string
-timeToOrder: int
-noOfWebSales: int

+confirmEmail (emailAddress)

InStoreSale::Sale

-storeID: string
-registerID: string
-clerkID: string
-noOfStoreSales: int

Thank You

Review Questions

Review Questions

1. In UML, what are three types of relationships found on a class diagram?
2. What is a generalization/specialization relationship, and what object-oriented terms does it illustrate?
3. Compare/contrast superclass and subclass. Compare/contrasts abstract class and concrete class.
4. What is a whole-part relationship, and why does it show multiplicity?
5. Compare/contrast aggregation with composition for a whole part relationship.
6. What is an object state?

Review Questions

7. What is a state transition?
8. When considering requirements, states and state transitions are important for understanding which other diagram?

9. What UML diagram is used to show the states and transitions for an object?
10. List the elements that make up a transition description. Which elements are optional?
11. What is a composite state? What is it used for?
12. What is meant by the term path?
13. What is the purpose of a guard-condition?
14. What are the steps in creating a state machine diagram?

Q & A