Systems Analysis and Design in a Changing World, Fifth Edition

CHAPTER 7

THE OBJECT-ORIENTED APPROACH TO REQUIREMENTS

7

Learning Objectives

- Understand the models and processes of defining object-oriented requirements
- Develop use case diagrams and activity diagrams
- Develop system sequence diagrams
- Develop state machine diagrams to model object behavior
- Explain how UML diagrams work together to define functional requirements for the object-oriented approach

Overview

- The objective of requirements definition is understanding – understanding the users' needs, the business processes, and the systems to support business processes
- Understand and define requirements for a new system using object-oriented analysis models and techniques
- Line between object-oriented analysis and object-oriented design is somewhat fuzzy
  - Iterative approach to development
  - Models built in analysis are refined during design

Object-Oriented Requirements

- Object-oriented modeling notation is Unified Modeling Language (UML 2.0)
- UML was accepted by Object Management Group (OMG) as standard modeling technique
- Purpose of Object Management Group
  - Promote theory and practice of object-oriented technology for development of distributed systems
  - Provide common architectural framework for OO

Object-Oriented Requirements (continued)

7

- Object-oriented system requirements are specified and documented through process of building models
- Modeling process starts with identification of use cases and problem domain classes (things in users' work environment)
- Business events trigger elementary business processes (EBP) that new system must address as use cases
- Use cases define functional requirements

Object-Oriented Requirements Models

7

- Use case model – a collection of models to capture system requirements
- Use case diagram – identify actors and their roles and how the actor roles utilize the system
- Systems sequence diagrams (SSDs) – define inputs and outputs and sequence of interactions between user and system for a use case

Object-Oriented Requirements Models (continued)

- Message – the communication between objects within a use case
- Domain model – describes the classes of objects and their states
- State machine diagrams – describe states of each object

Requirements Models—Traditional vs OO

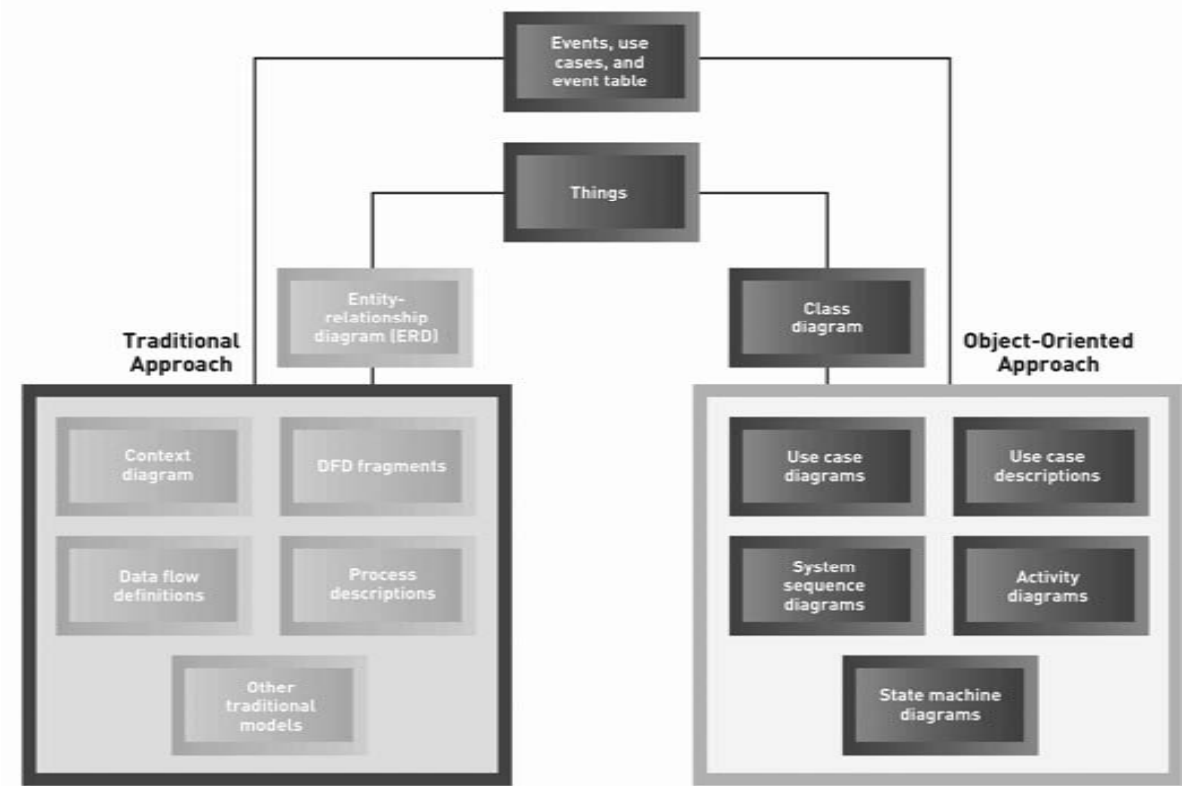

Figure 7-1

The System Activities—A Use Case/Scenario View

- Use case analysis used to identify and define all business processes that system must support
- Use case – an activity a system carried out, usually in response to a user request
- Actor
  - Role played by user
  - Outside automation boundary

Techniques for Identifying Use Cases (Review from Chapter 5)

- Identify user goals
  - Each goal at the elementary business process (EBP) level is a use case
  - EBP – task performed by one user in one place and in response to business event that adds measurable business value, and leaves system and data in consistent state
- Event decomposition technique (event table)
- CRUD analysis technique (create, read/report, update, delete) to ensure coverage

Use Case Diagram

- Graphical UML diagram that summarizes information about actors and use cases
- Simple diagram shows overview of functional requirements
- Can have multiple use case diagrams
  - By subsystem
  - By actor

Simple Use Case with an Actor

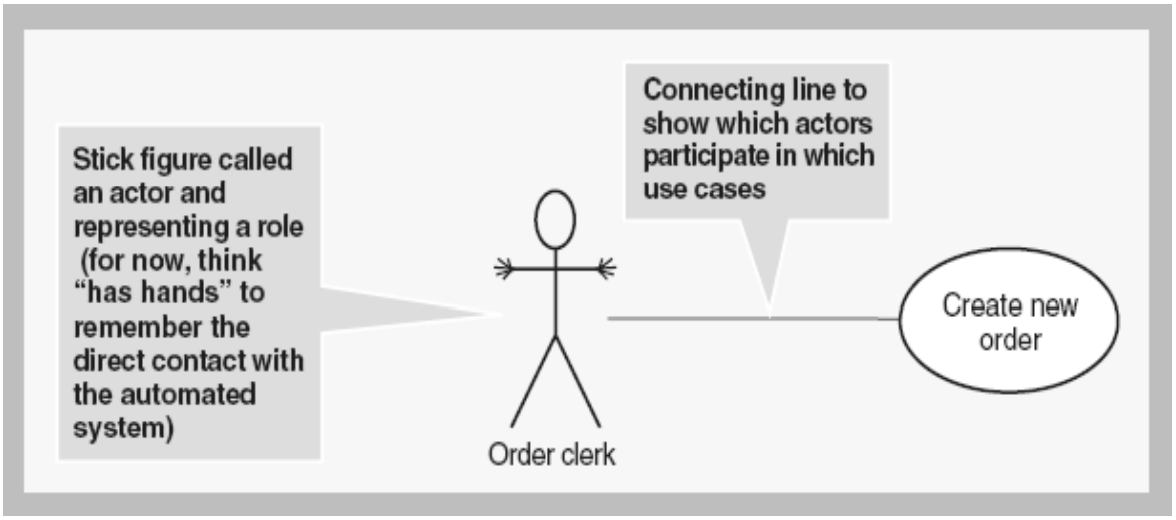

Figure 7-2

Use Case Diagram with Automation Boundary and Alternate Actor Notation

7

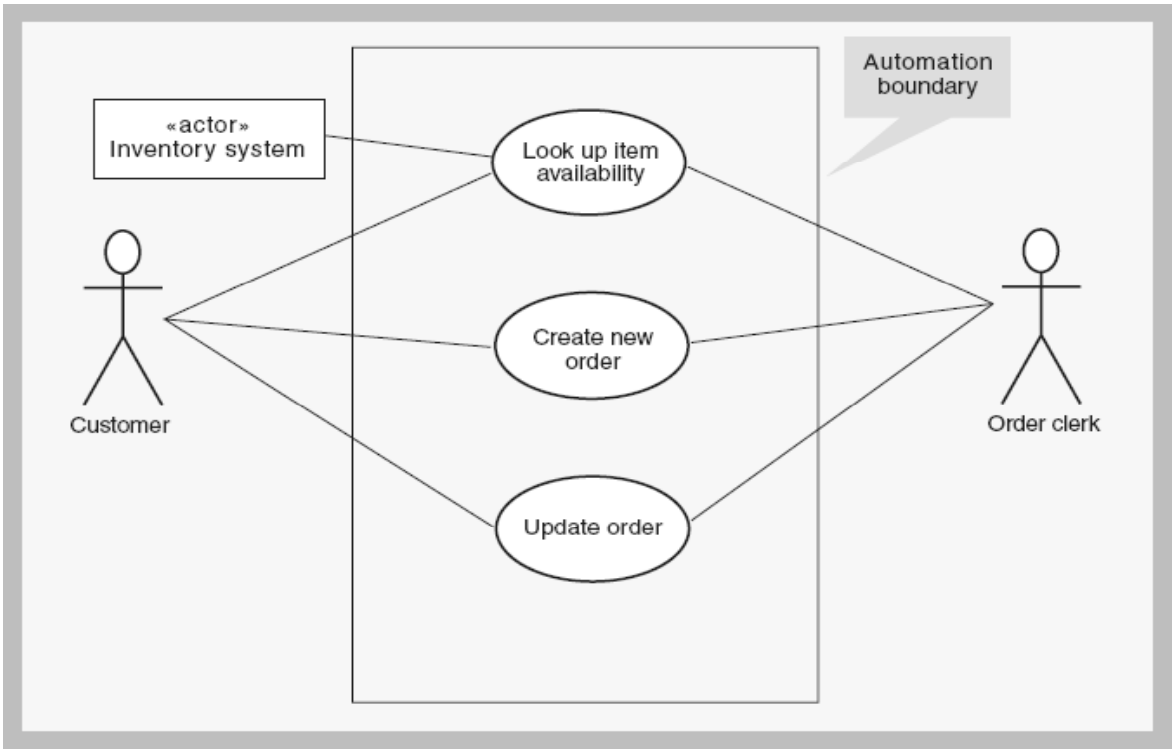

Figure 7-3

All Use Cases Involving Customer as Actor

7

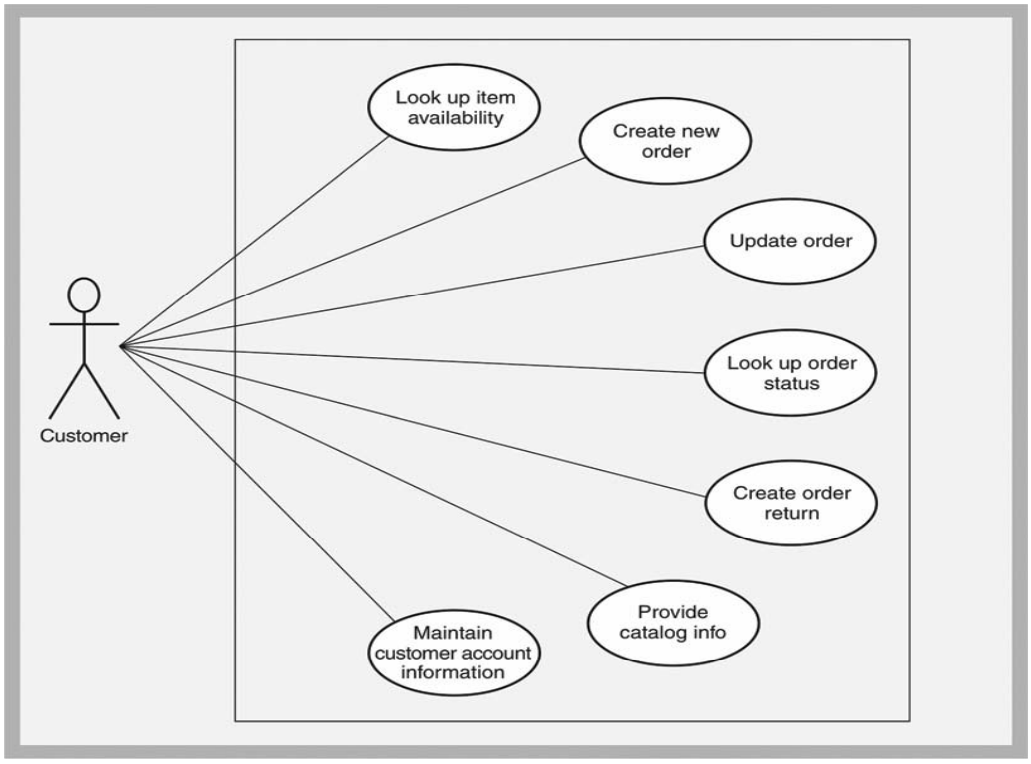

Figure 7-4

Use Cases of RMO Order Entry Subsystem

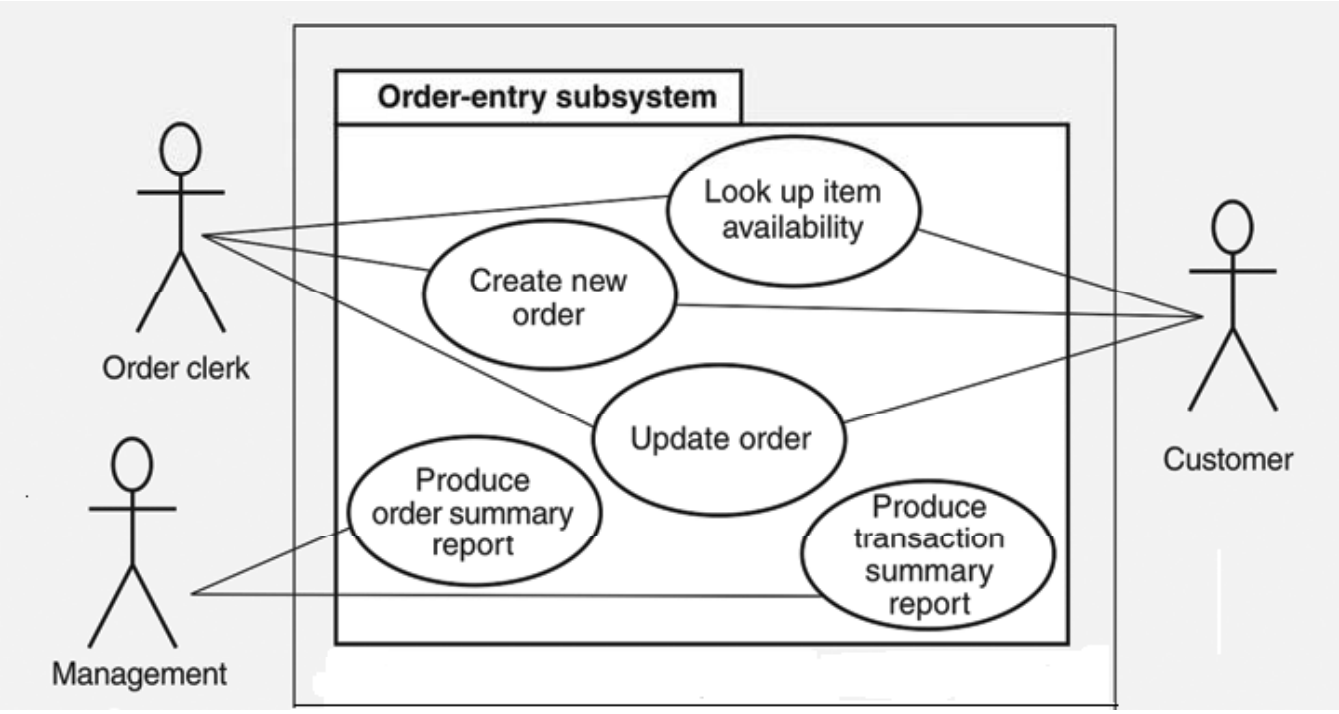

Figure 7-5 (partial figure)

<<Includes>> Relationship

- Documents situation in which one use case requires the services of a common subroutine
- Another use case is developed for this common subroutine
- A common use case can be reused by multiple use cases

Example of Order-Entry Subsystem with <<Includes>> Use Cases

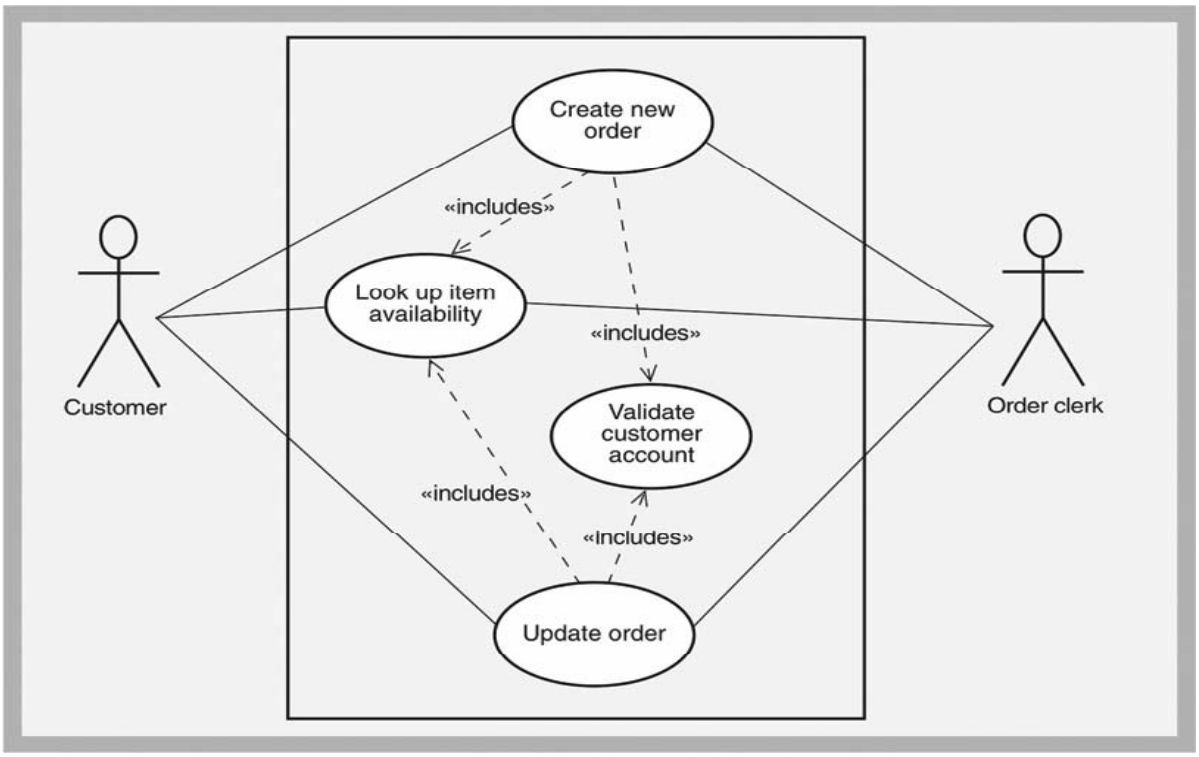

Figure 7-6

Developing a Use Case Diagram

- Underlying conditions for describing use cases
  - Based on automated system, e.g. users “touch” the system
  - Assume perfect technology condition
- Iterate through these two steps
  - Identify actors as roles
  - List goals, e.g. use cases, for each actor. A goal is a unit of work.
- Finalize with a CRUD analysis to ensure completeness

Activity Diagrams

- Used to document workflow of business process activities for each use case or scenario
- Standard UML 2.0 diagram as seen in Chapter 4
- Can support any level of use case description; a supplement to use case descriptions
- Helpful in developing system sequence diagrams

Activity Diagram—Telephone Order Scenario

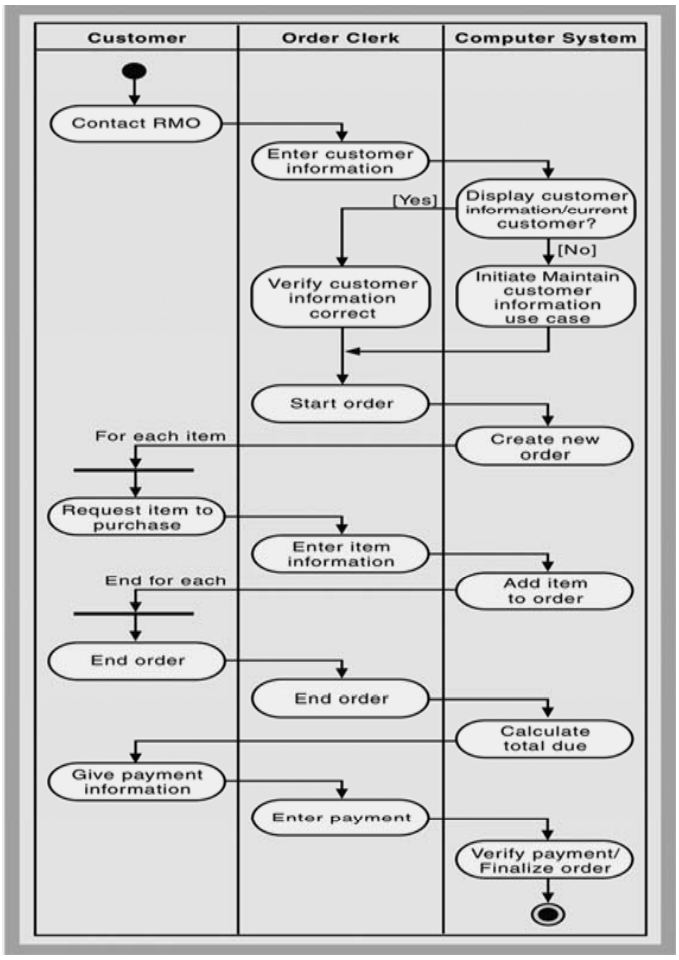

Figure 7-8

Activity Diagram—Web Order Scenario

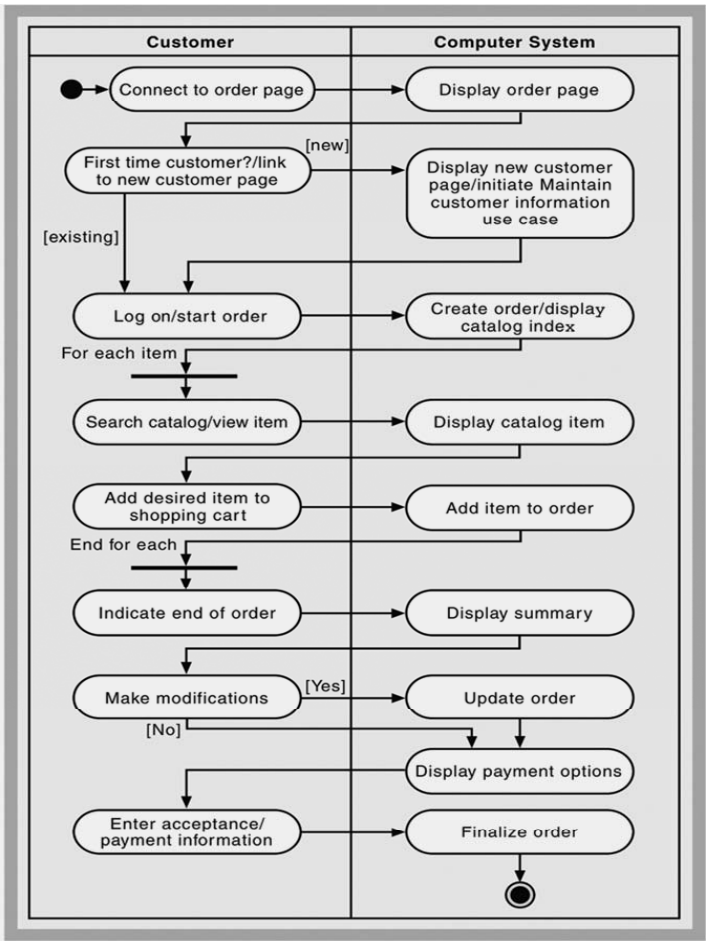

Figure 7-9

Identifying Inputs and Outputs—The System Sequence Diagram

- Interaction diagram – a communication diagram or a sequence diagram
- System sequence diagram (SSD) is type of UML 2.0 interaction diagram
- Used to model input and output messaging requirements for a use case or scenario
- Shows sequence of interactions as messages during flow of activities
- System is shown as one object: a “black box”

SSD Notation

- Lifeline or object lifeline is a vertical line under object or actor to show passage of time for object
- Message is labeled on arrows to show messages sent to or received by actor or system
- Actor is role interacting with the system with messages
- Object is the component that interacts with actors and other objects

System Sequence Diagram (SSD) Notation

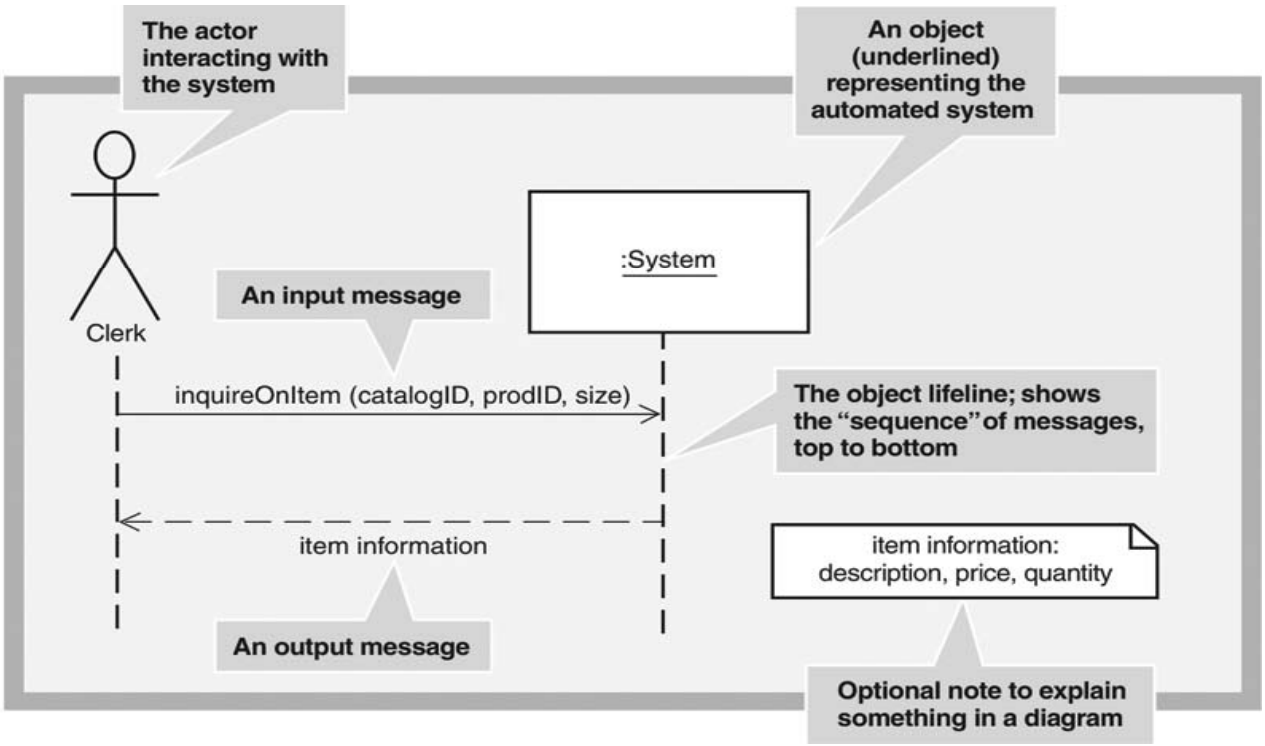

Figure 7-10

SSD Lifelines

- Vertical line under object or actor
  - Shows passage of time
- If vertical line dashed
  - Creation and destruction of thing is not important for scenario
- Long narrow rectangles
  - Activation lifelines emphasize that object is active only during part of scenario

SSD Messages

- Internal events identified by the flow of objects in a scenario
- Requests from one actor or object to another to do some action
- Invoke a particular method

Repeating Message

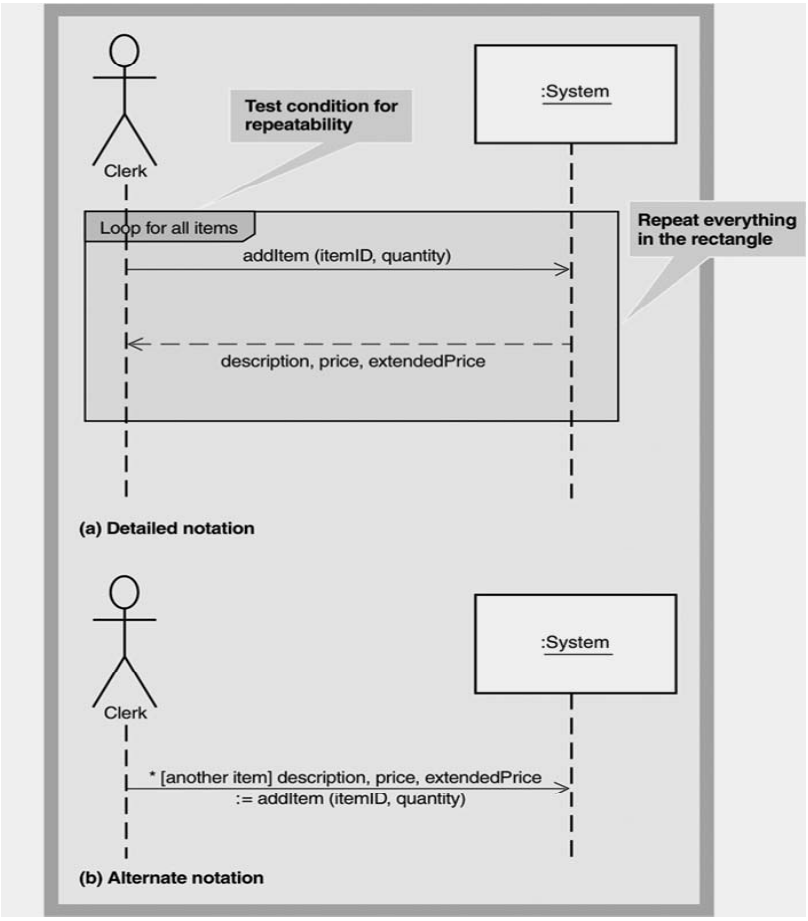

Figure 7-11

Developing a System Sequence Diagram

- Begin with detailed description of use case from fully developed form or activity diagram
- Identify input messages
- Describe message from external actor to system using message notation
- Identify and add any special conditions on input message, including iteration and true/false conditions
- Identify and add output return messages

Activity Diagram of the Telephone Order Scenario

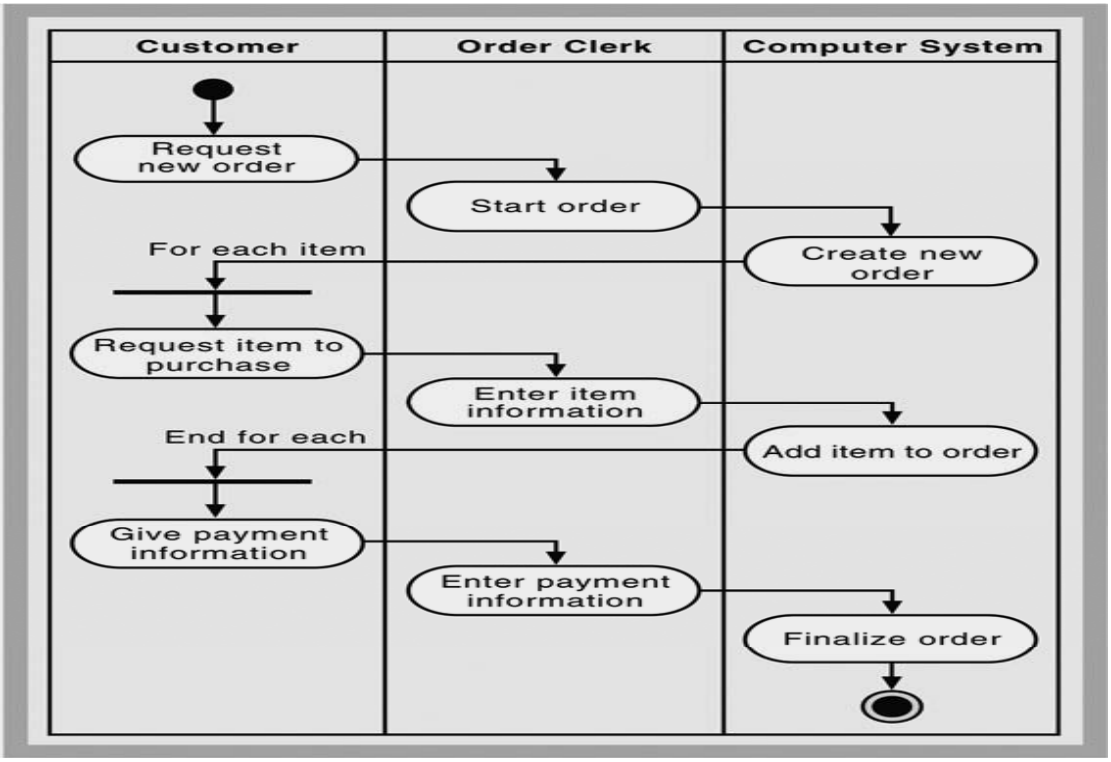

Figure 7-12

Resulting SSD for the Telephone Order Scenario

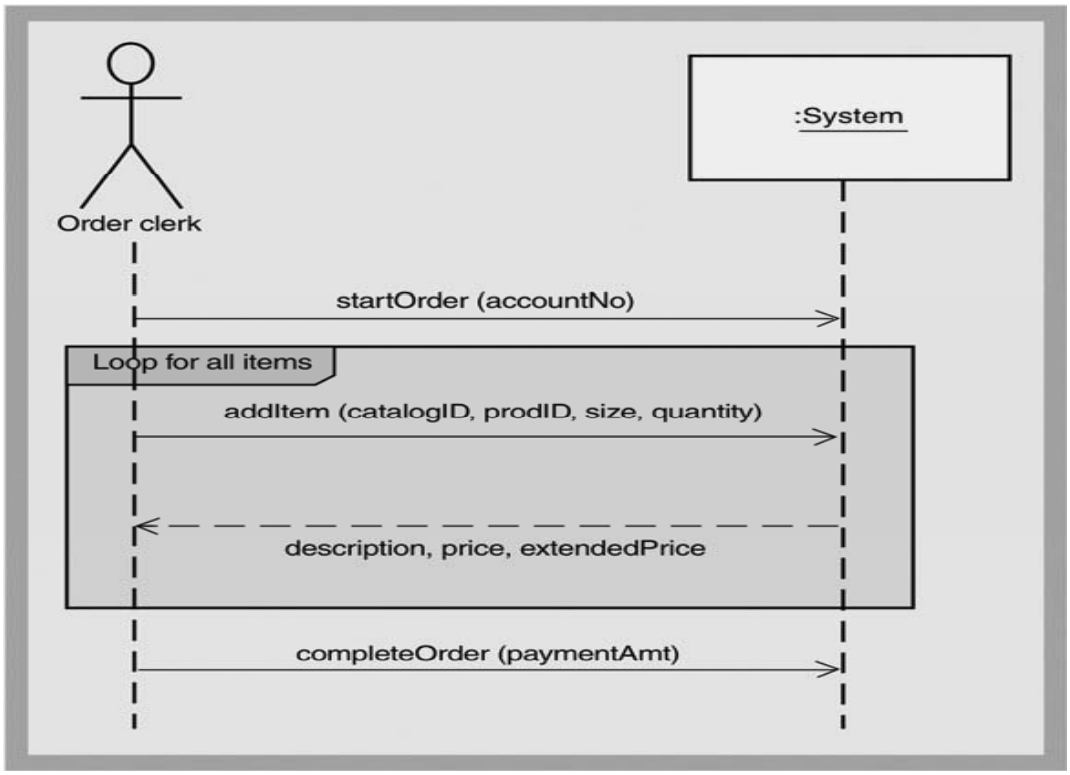

Figure 7-13

SSD of the Web Order Scenario for the Create New Order Use case

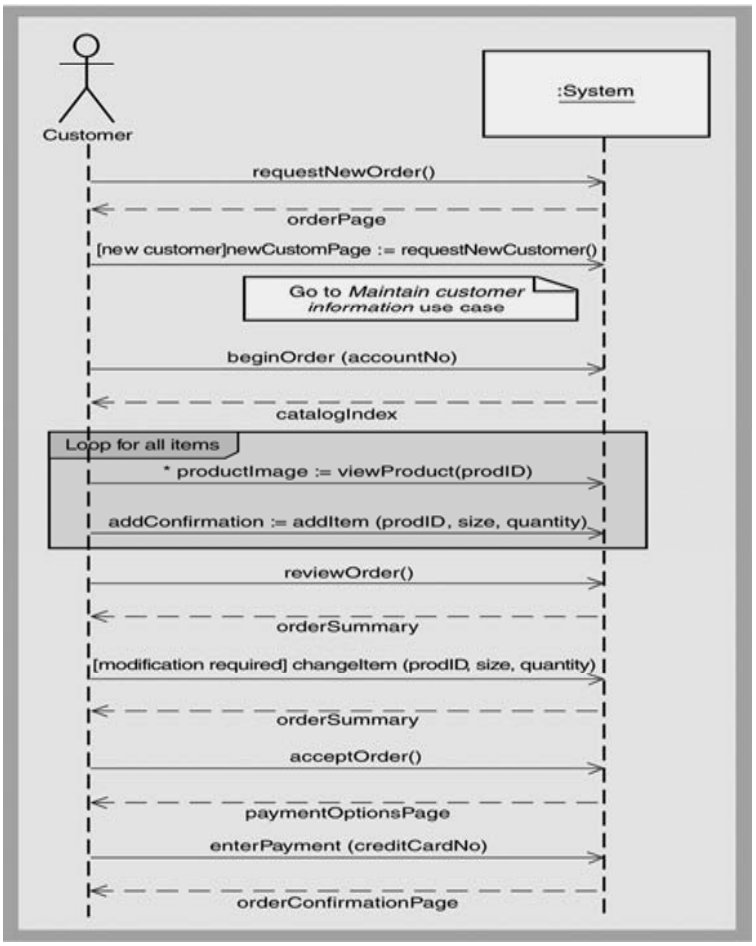

Figure 7-14

Identifying Object Behavior— The State Machine Diagram

- State machine diagram is UML 2.0 diagram that models object states and transitions
  - Complex problem domain classes can be modeled
- State of an object
  - A condition that occurs during its life when it satisfies some criterion, performs some action, or waits for an event
  - Each state has unique name and is a semipermanent condition or status
- Transition
  - The movement of an object from one state to another state

Simple State Machine Diagram for a Printer

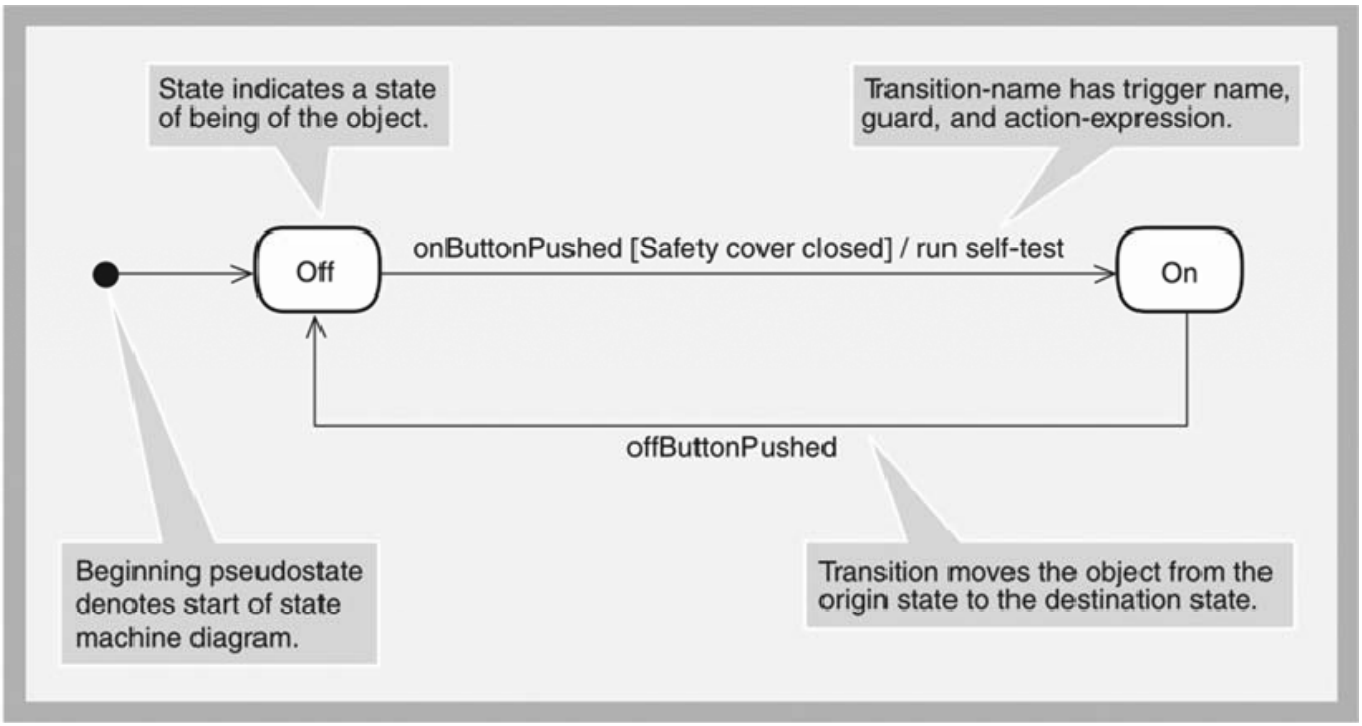

Figure 7-15

State Machine Terminology

- Pseudostate – the starting point of a state machine, indicated by a black dot
- Origin state – the original state of an object from which the transition occurs
- Destination state – the state to which an object moves after the completion of a transition
- Message event – the trigger for a transition, which causes the object to leave the origin state
- Guard condition – a true/false test to see whether a transition can fire
- Action expression – a description of the activities performed as part of a transition

Composite States and Concurrency—States within a State

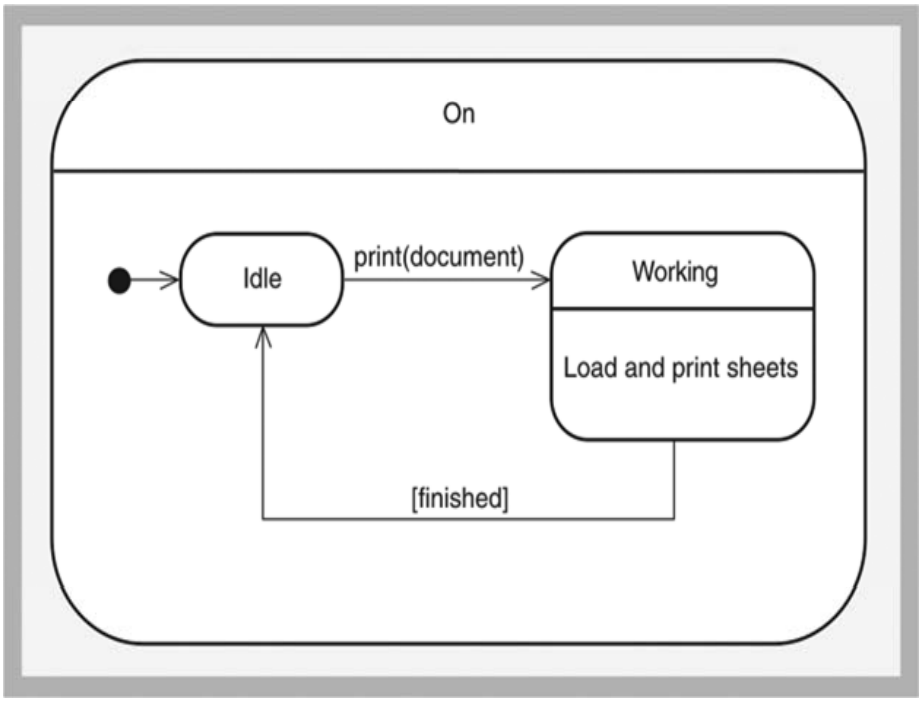

Figure 7-16

Concurrent Paths for Printer in the On State

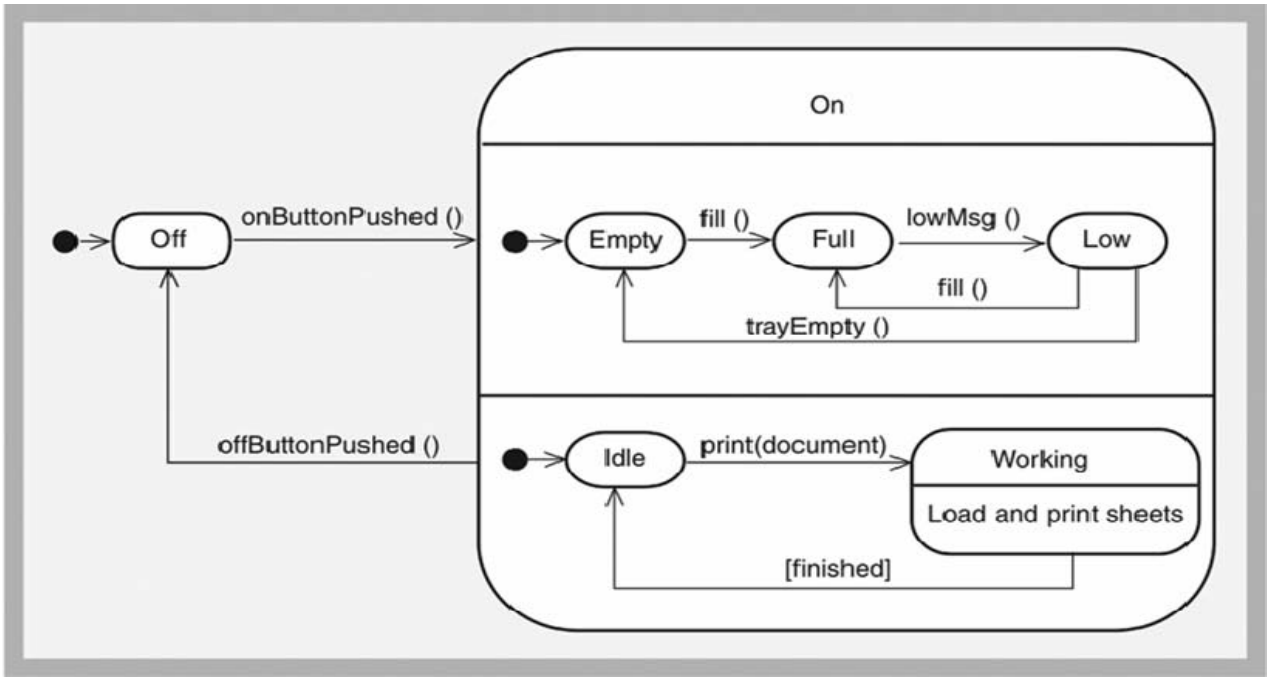

Figure 7-17

Rules for Developing State Machine Diagram

- Review domain class diagram, select important ones, and list all state and exit conditions
- Begin building state machine diagram fragments for each class
- Sequence fragments in correct order and review for independent and concurrent paths
- Expand each transition with message event, guard-condition, and action-expression
- Review and test each state machine diagram

States and Exit Transitions for OrderItem

State	Transition causing exit from state
Newly added	finishedAdding
Ready to ship	shipItem
On back order	itemArrived
Shipped	No exit transition defined

Figure 7-18

Partial State Machine for OrderItem

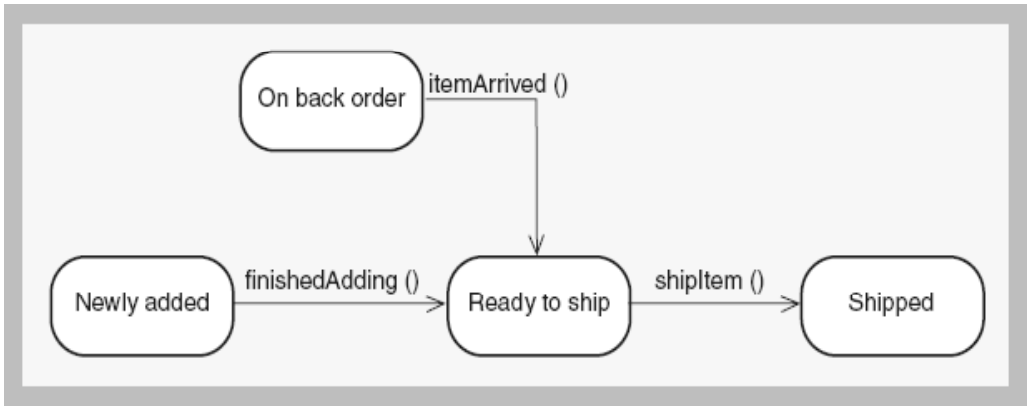

Figure 7-19

Final State Machine for OrderItem

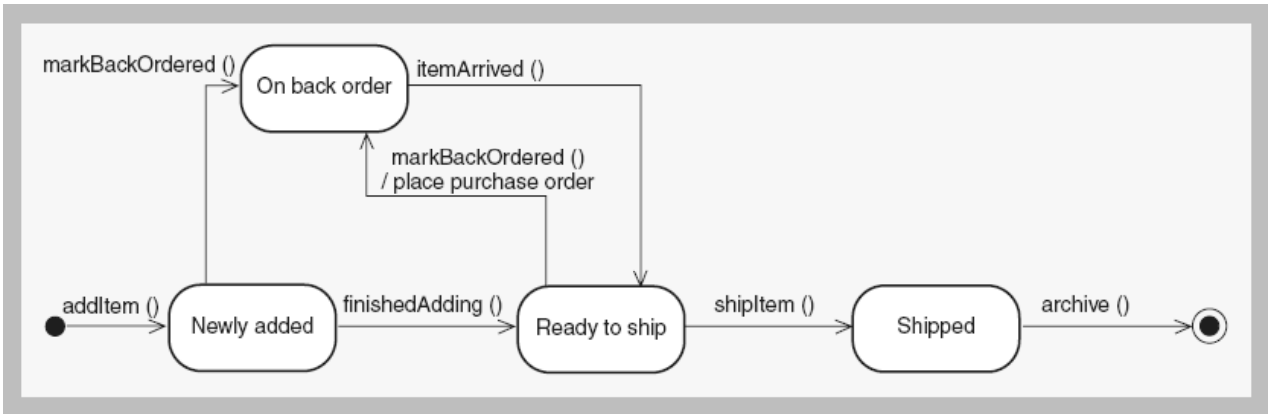

Figure 7-20

Order Domain Class for RMO—States and Exit Transitions

State	Exit transition
Open for item adds	completeOrder
Ready for shipping	beginShipping
In shipping	shippingComplete
Waiting for back orders	backOrdersArrive
Shipped	paymentCleared
Closed	archive

Figure 7-21

First-Cut State Machine Diagram for Order

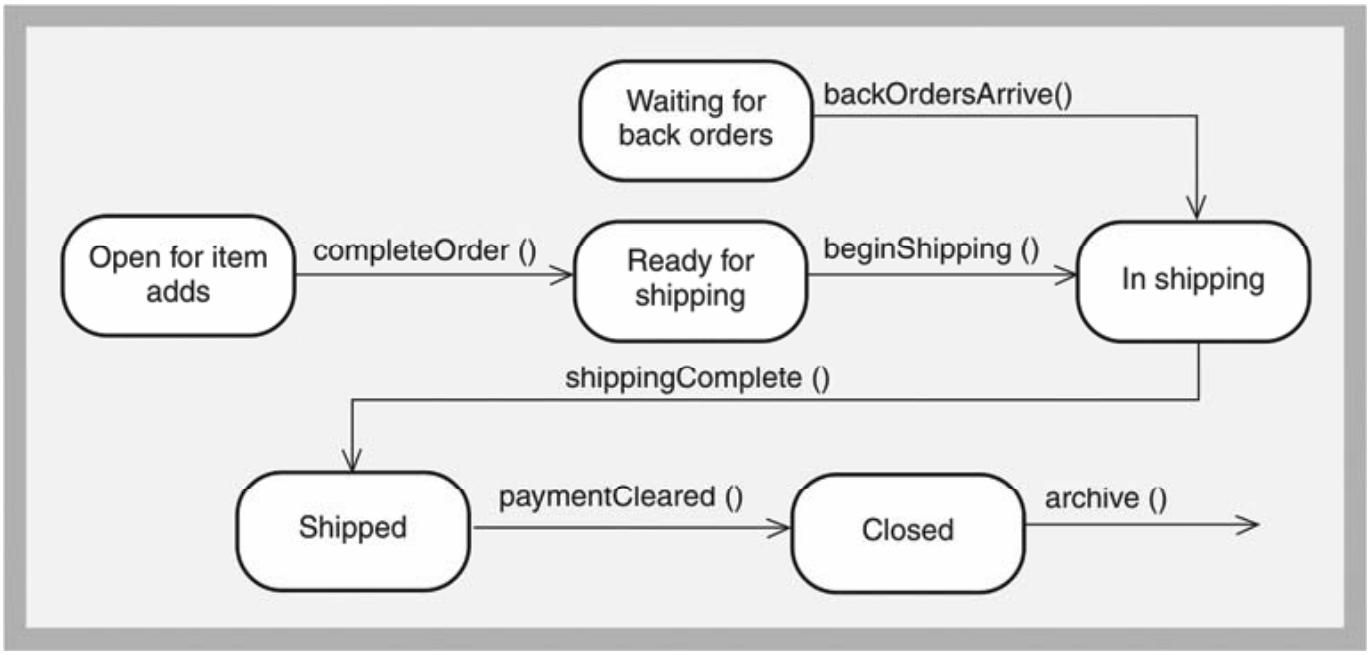

Figure 7-22

Second-Cut State Machine Diagram for Order

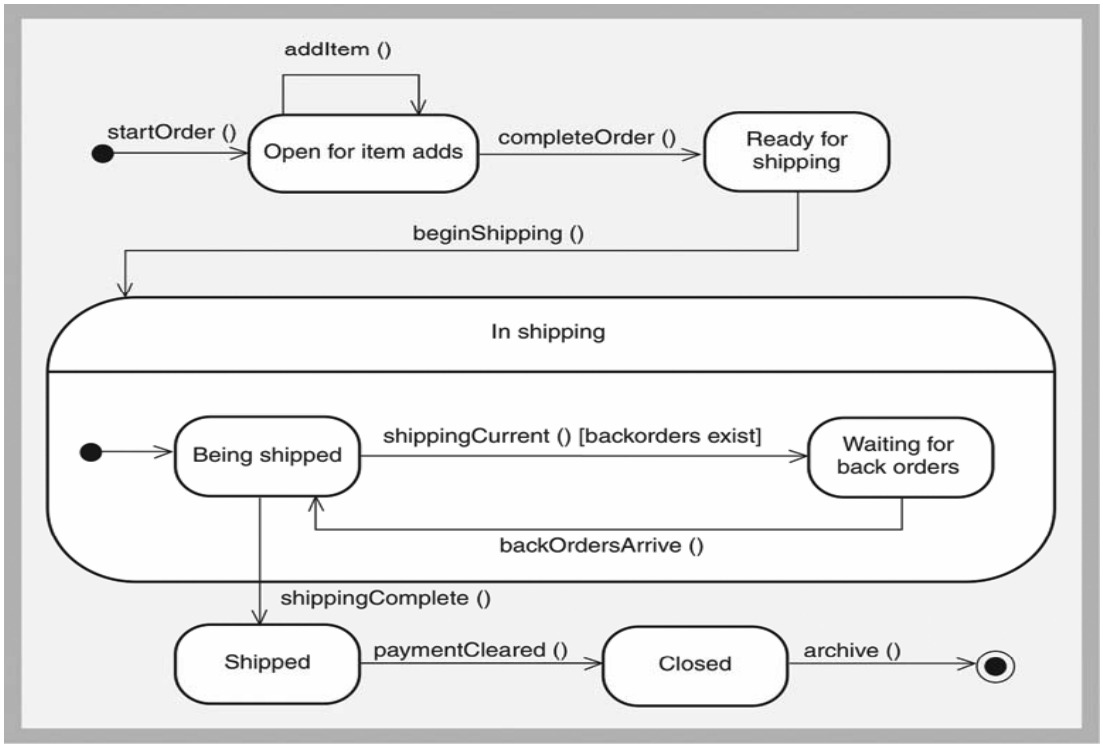

Figure 7-23

Integrating Object-Oriented Models

- Complete use case diagram is needed to understand total scope of new system
- Domain model class diagrams should also be as complete as possible for entire system
- With iterative approach, only construct use case descriptions, activity diagrams, and system sequence diagrams for use cases in iteration
- Development of a new diagram often helps refine and correct previous diagrams

Relationships Between OO Requirements Models

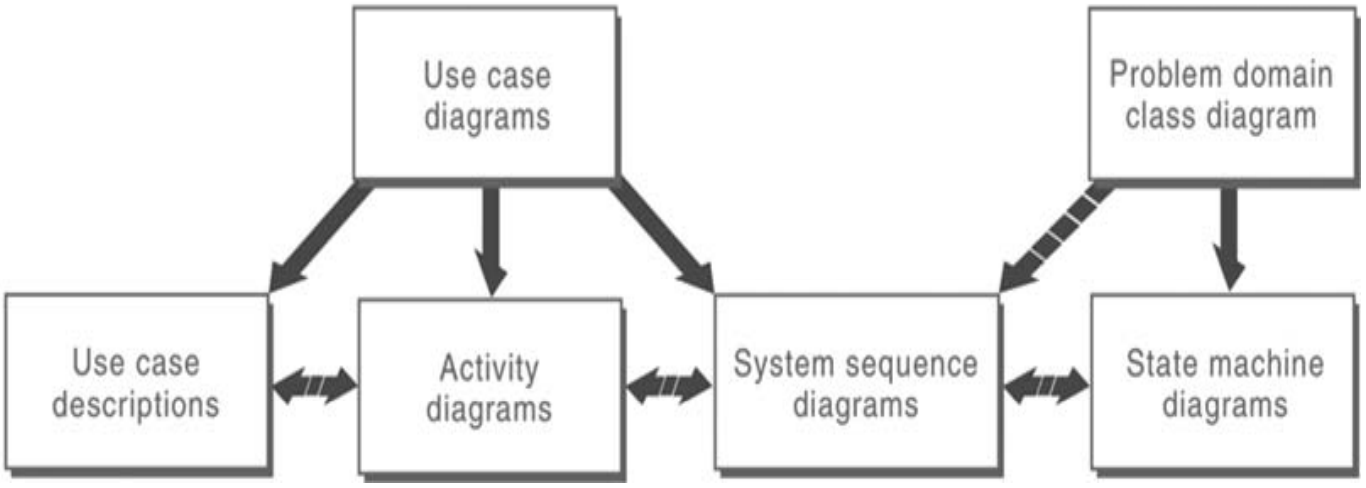

Figure 7-24

Summary

- Object-oriented approach has complete set of diagrams that define system requirements
- Requirements specified using following models
  - Domain model class diagram (Chapter 5)
  - Use case diagrams (Chapters 7)
  - Use case detailed models, either descriptive formats or activity diagrams (Chapter 5 & 7)
  - System sequence diagrams (Chapter 7)
  - State machine diagrams (Chapter 7)