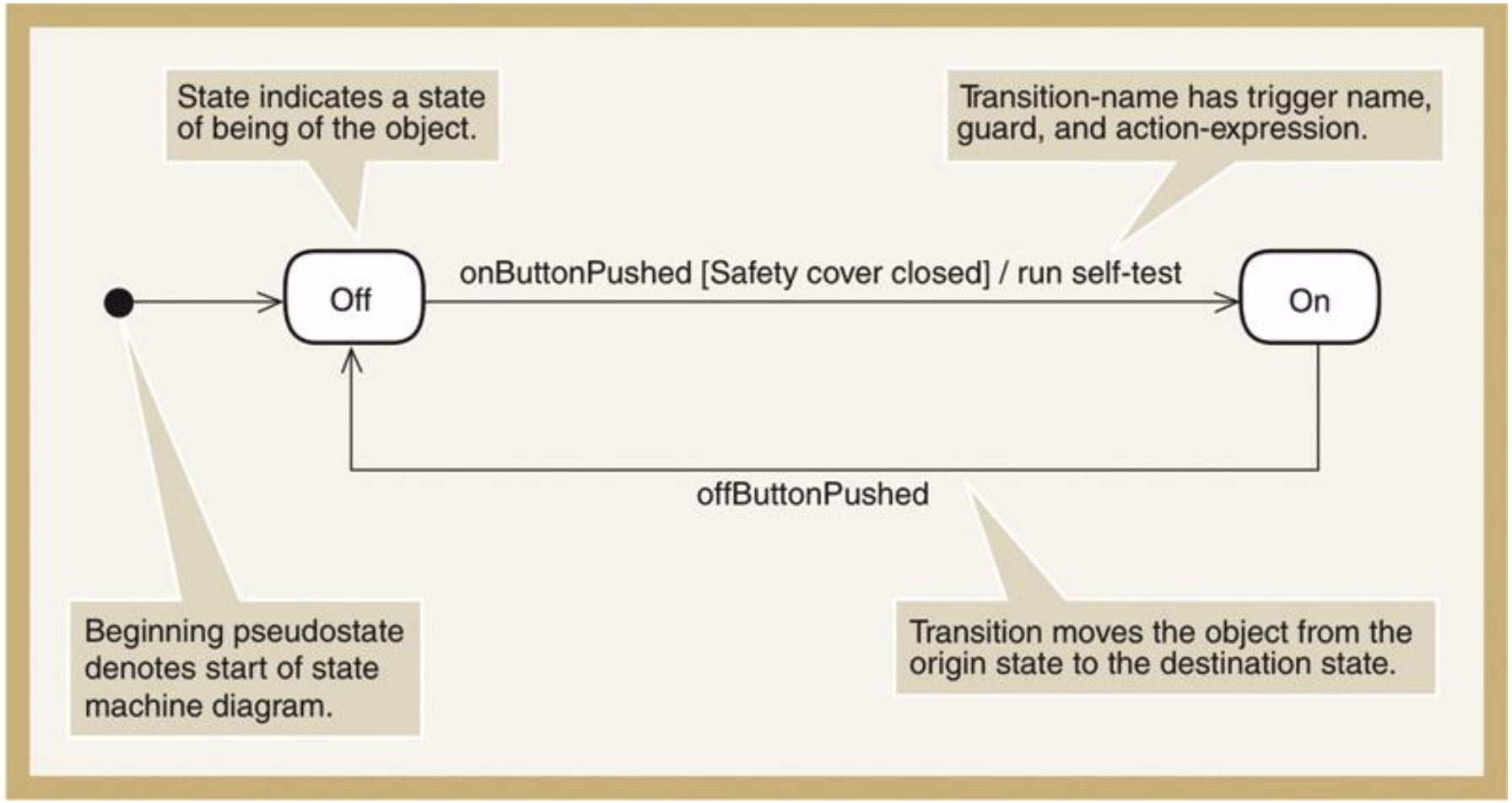
State Machine Diagram

Identifying Object Behaviour

State Machine Diagram

- State machine diagram is UML 2.0 diagram that models object states and transitions
  - Complex problem domain classes can be modelled
- 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 semi permanent condition or status
- Transition
  - The movement of an object from one state to another state

Simple State Machine Diagram for a Printer

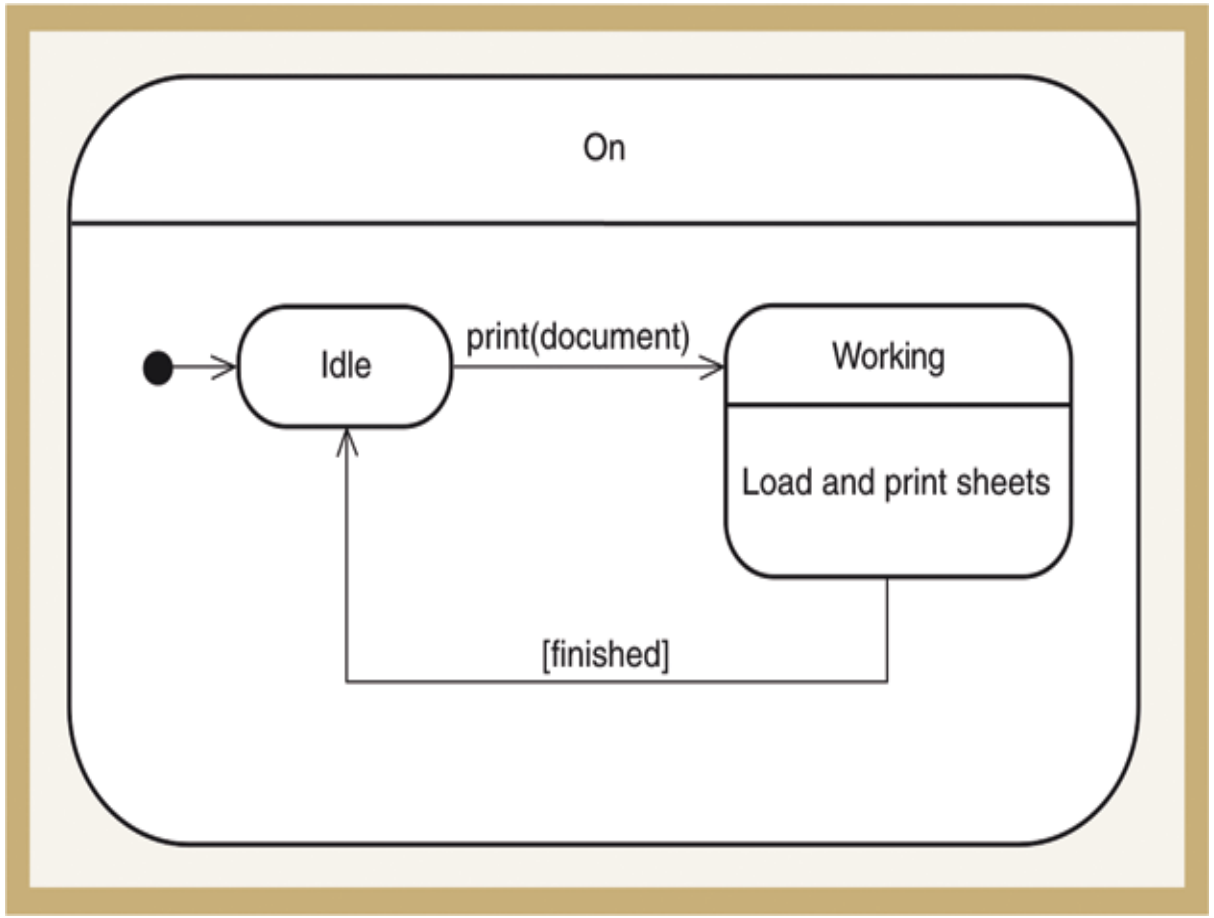


State Machine Terminology

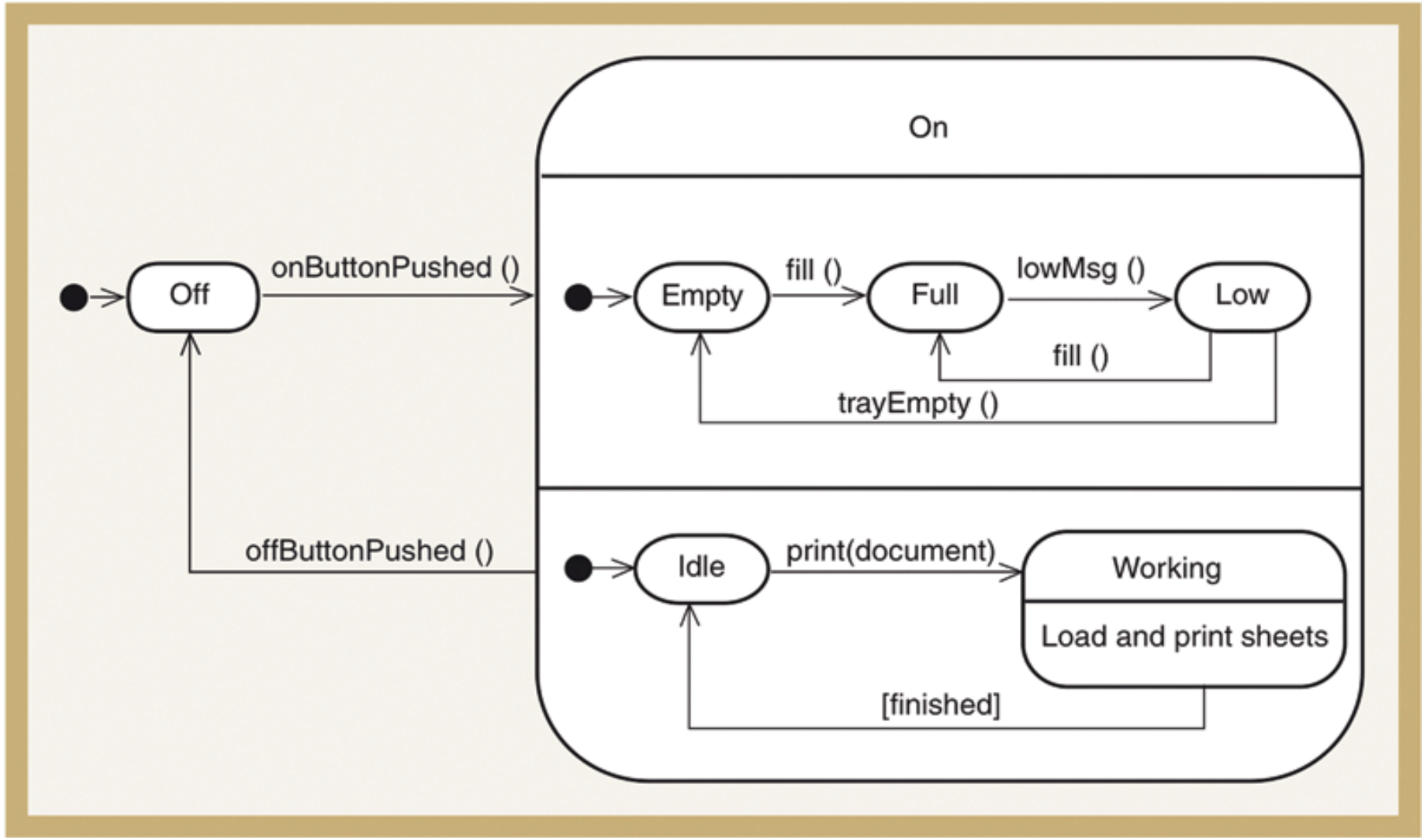
- Pseudo state – the starting point of a state machine, indicated by a black dot
- Origin state – the original state of an object from which the transition occurs
- Destination state – the state to which an object moves after the completion of a transition
- Message event – the trigger for a transition, 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



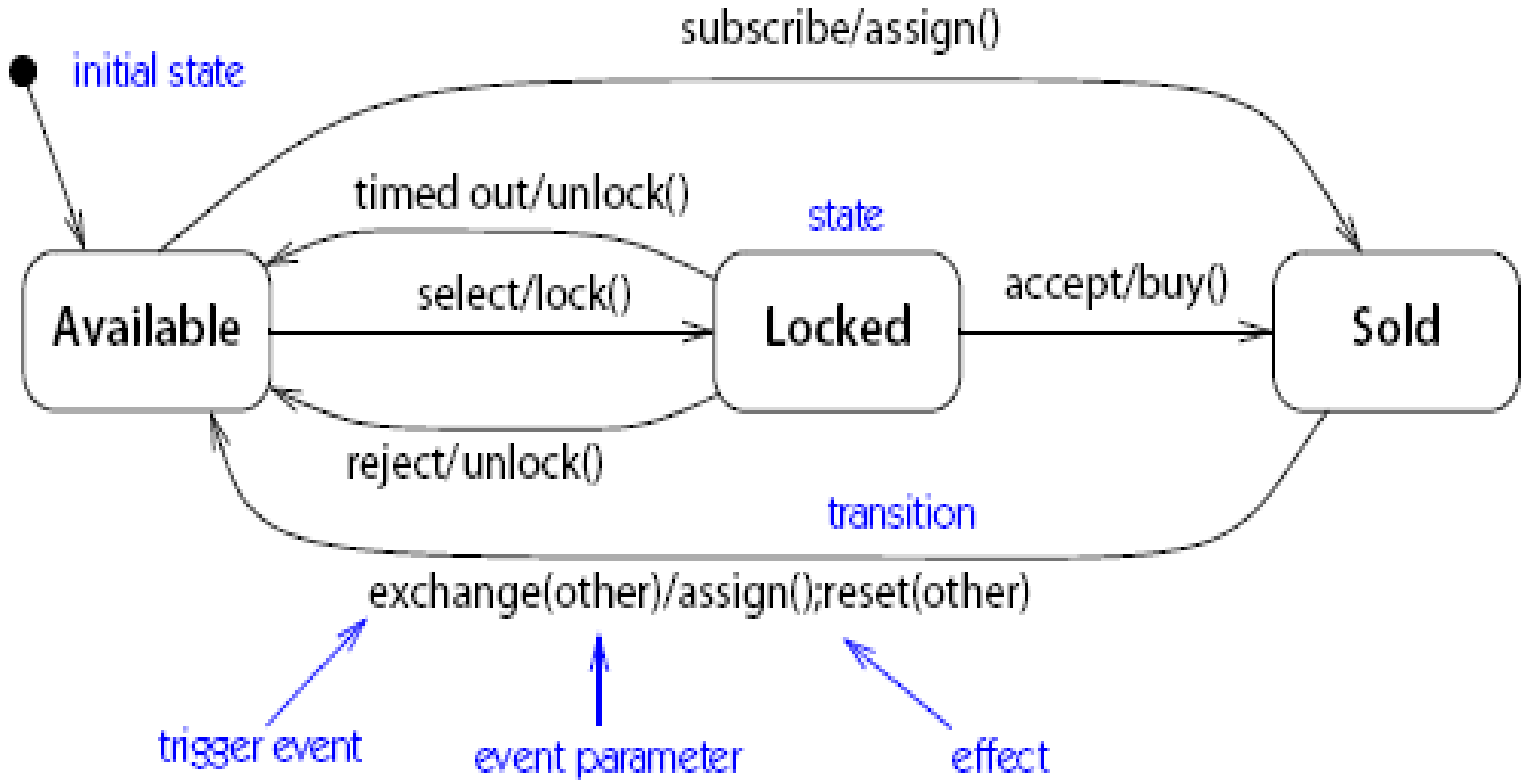
Concurrent Paths for Printer in the On State



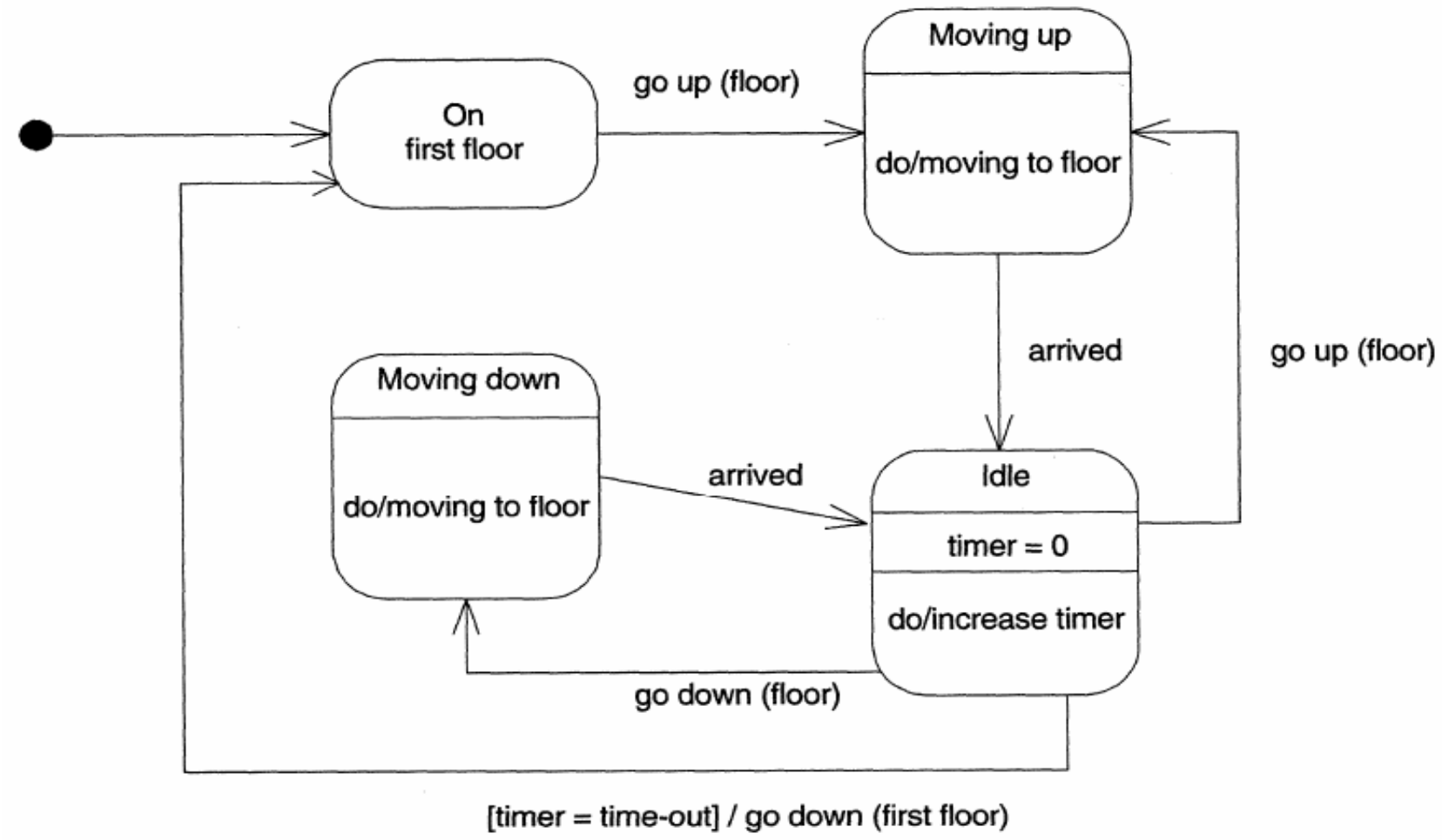
Rules for Developing State Machine Diagram

- 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

State machine for Ticket Object

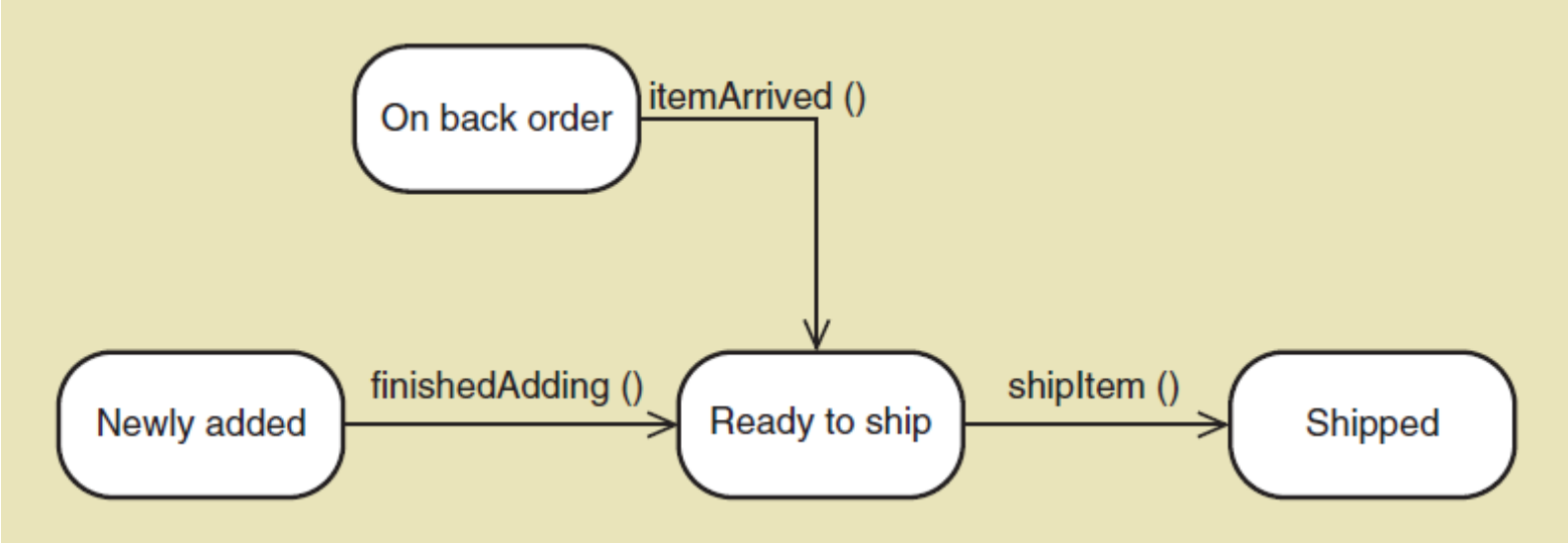


State machine for Lift



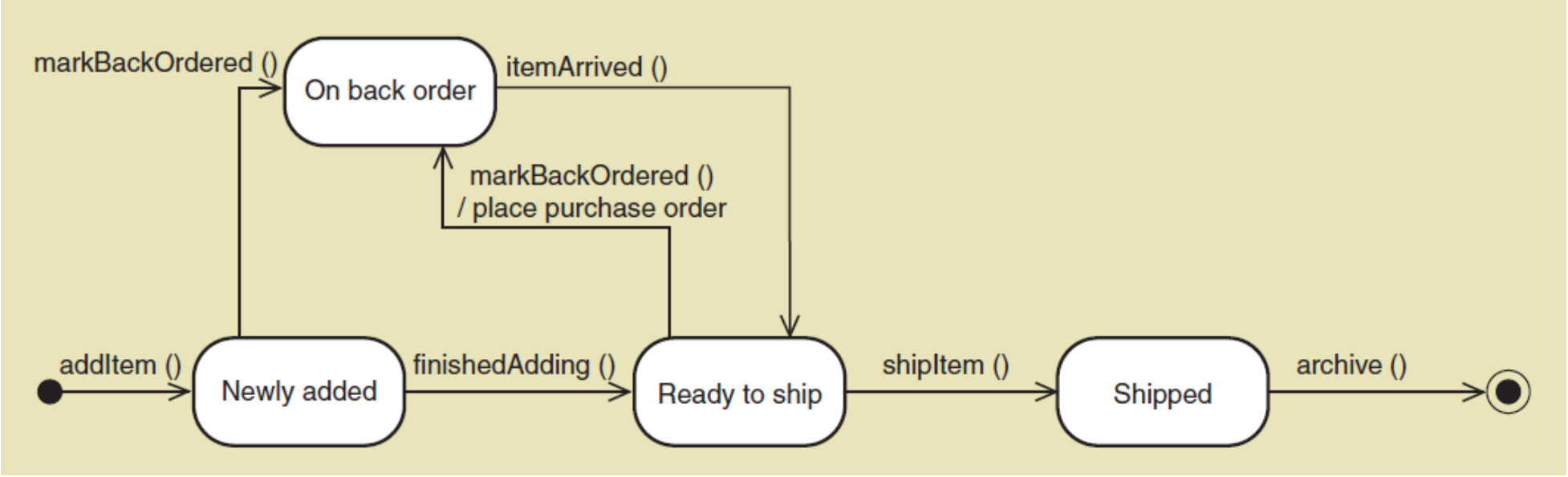
RMO Domain Class States for SaleItem Object

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



Final State Machine Diagram for SaleItem Object

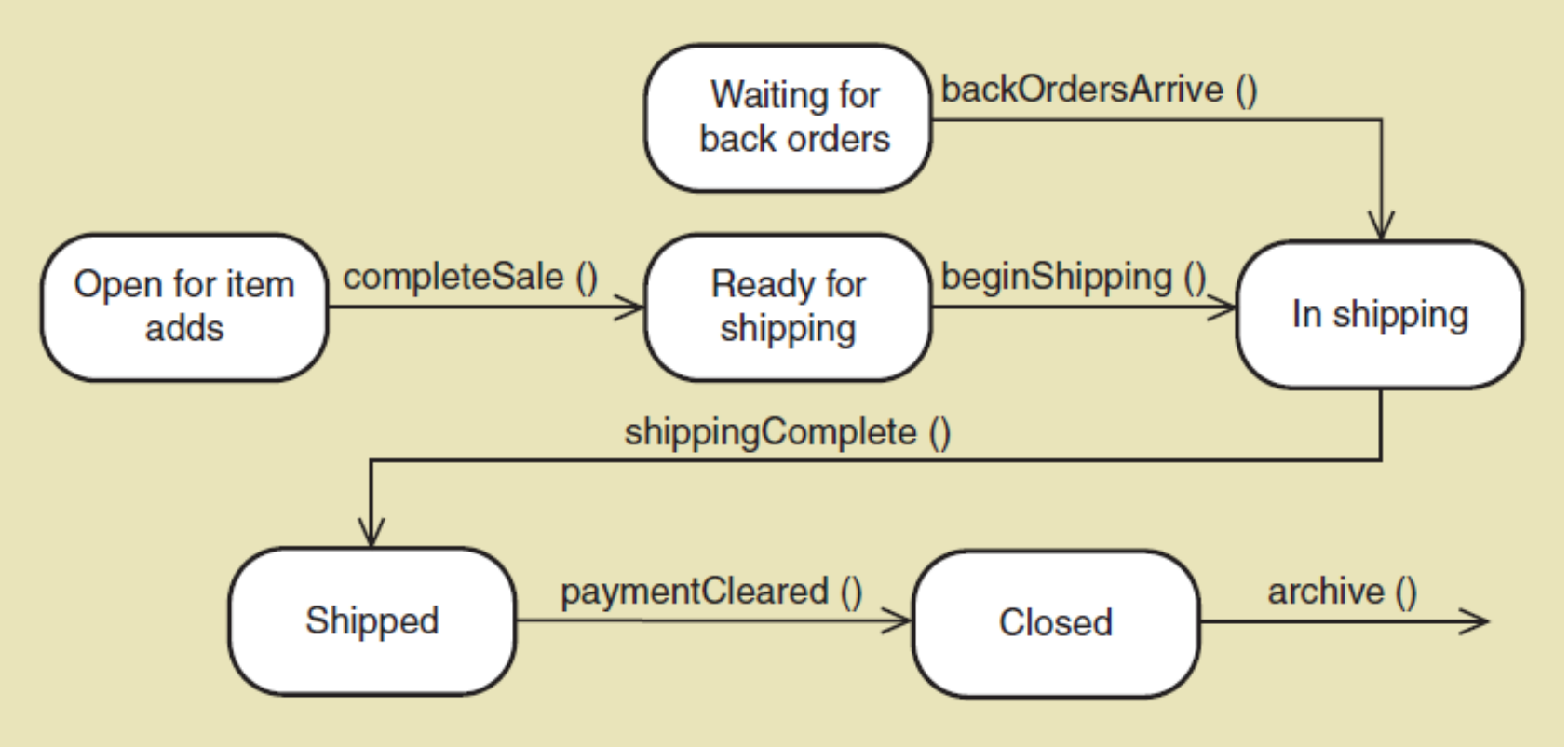
- addItem() and archive() transitions added
- markBackOrdered() transition added



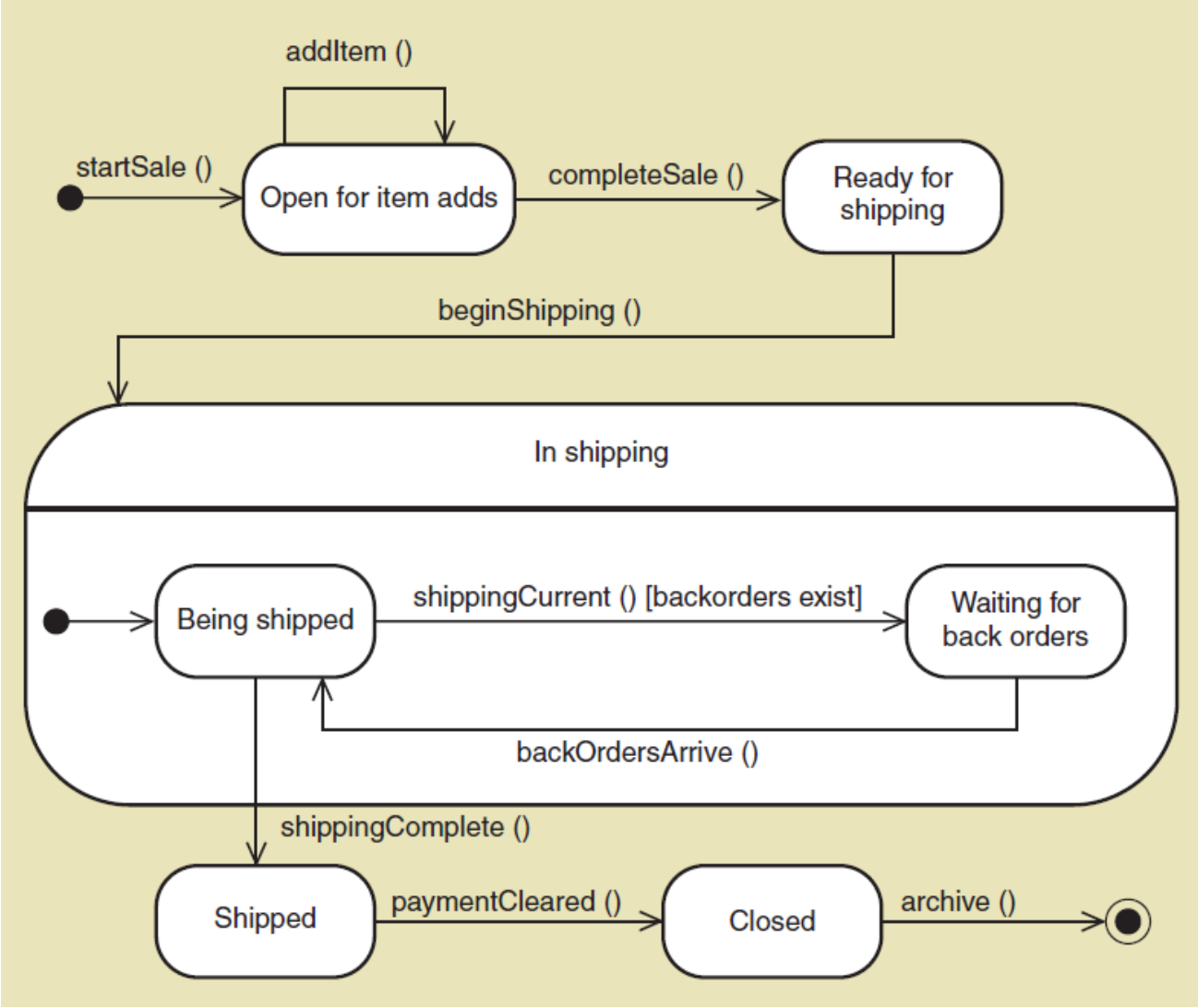
RMO Domain Class States for Sale Object

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

Initial State Machine Diagram for RMO Sale Object



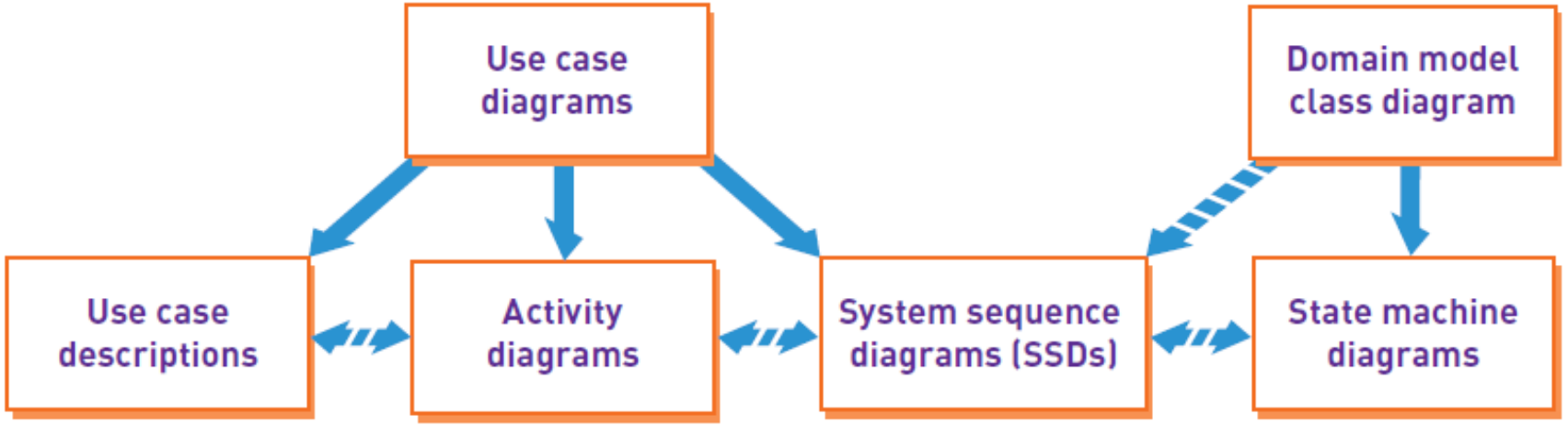
Final State Machine Diagram for Sale Object



Extending and Integrating Requirements Models

- Use cases
  - Use case diagram
    - Use case description
    - Activity diagram
    - System sequence diagram (SSD)
- Domain Classes
  - Domain model class diagram
    - State machine diagram

Integrating Requirements Models



Summary

- Chapters 3 and 4 identified and modeled the two primary aspects of functional requirements: *use cases* and *domain classes*
- This chapter focuses on additional techniques and models to extend the requirements models to show more detail
- Fully *developed use case descriptions* provide information about each use case, including actors, stakeholders, preconditions, post conditions, the flow of activities and exceptions conditions
- *Activity diagrams* (first shown in Chapter 2) can also be used to show the flow of activities for a use case

Summary (continued)

- *System sequence diagrams* (SSDs) show the inputs and outputs for each use case as messages
- *State machine diagrams* show the states an object can be in over time between use cases
- Use cases are modeled in more detail using fully developed use case descriptions, activity diagrams, and system sequence diagrams
- Domain classes are modeled in more detail using state machine diagrams
- Not all use cases and domain classes are modeled at this level of detail. Only model when there is complexity and a need to communicate details