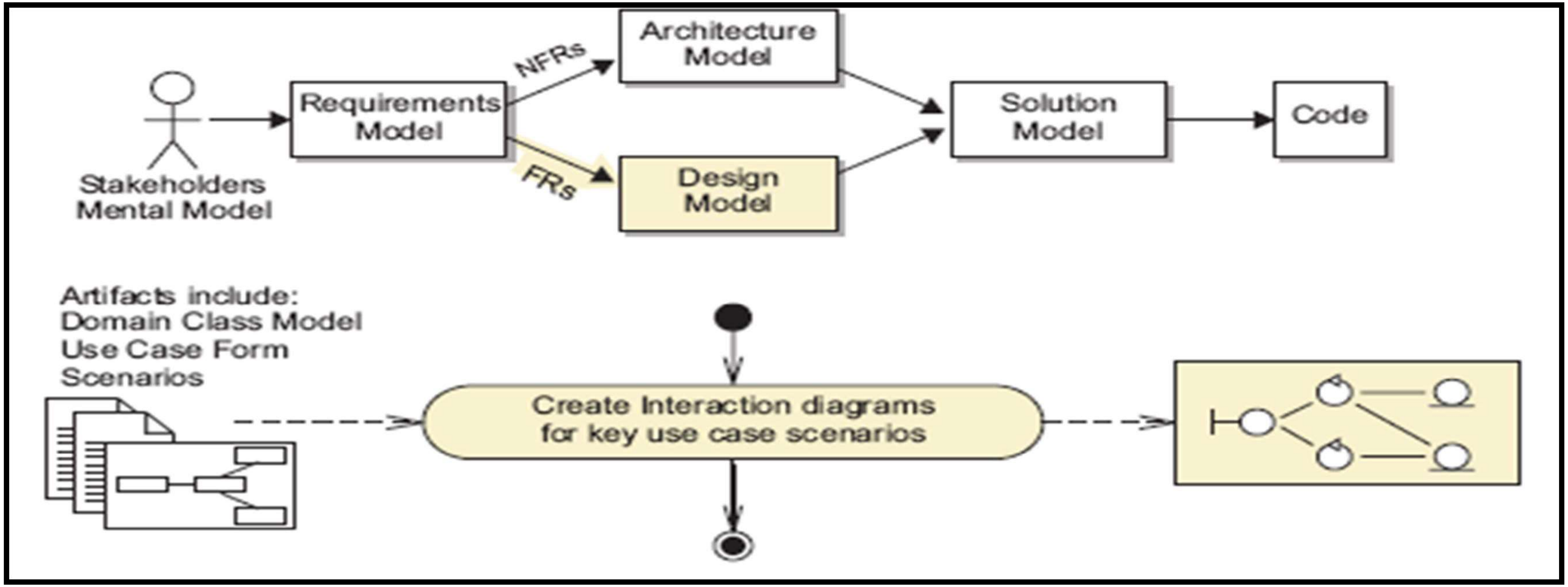
8

Other UML Diagrams

Objectives

After completing this lesson, you should be able to do the following:

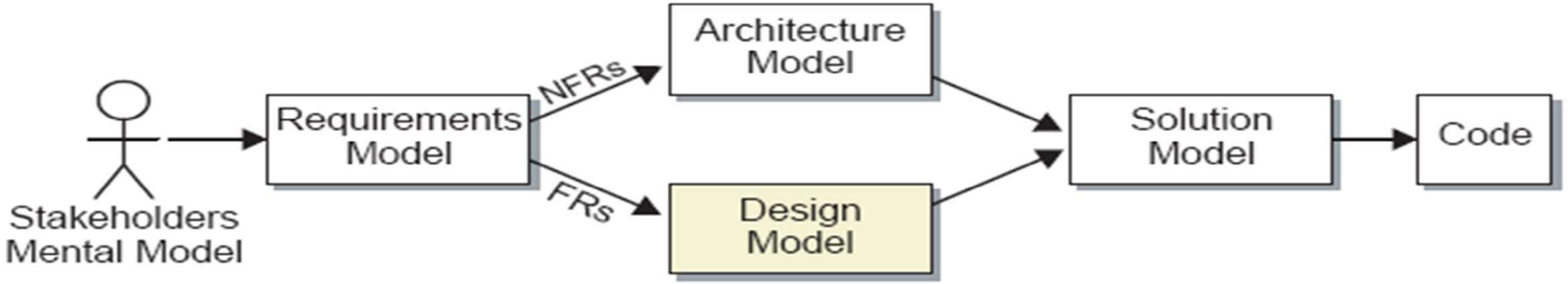
- Explain the purpose and elements of the Design model
- Identify the essential elements of a UML Communication diagram
- Create a Communication diagram view of the Design model
- Identify the essential elements of a UML Sequence diagram
- Create a Sequence diagram view of the Design model



Stakeholders Mental Model
Requirements Model
NFRs
FRs
Architecture Model
Design Model
Solution Model
Code
Artifacts include:
Domain Class Model
Use Case Form
Scenarios
Create Interaction diagrams for key use case scenarios

Introducing the Design Model

- The Design model is created from the Requirements model (use cases and Domain model).
- The Design model is merged with the Architecture model to produce the Solution model.



Interaction Diagrams

UML Interaction diagrams are the collective name for the following diagrams:

- Sequence diagrams
- Communication diagrams
  - Formerly known as Collaboration diagrams
- Interaction Overview diagrams
  - A combination of Activity diagram and Sequence diagram fragments

Interaction Diagrams

- Interaction diagrams depict interactions of objects and their relationships.
- They also include the messages passed between them.

There are two types of interaction diagrams:

1. Sequence Diagrams
2. Collaboration Diagrams

Interaction diagrams are used for modeling:

- The control flow by time ordering using sequence diagrams.
- The control flow of organization using collaboration diagrams.

Interaction Diagrams

Each UML Interaction diagram is used to show the sequence of interactions that occur between objects during:

- One or two use case scenarios
- A fragment of one use case scenario

UML Interaction diagrams may also be used to show the sequence of interactions that occur between:

- Systems
- Subsystems

Comparing Analysis and Design

Analysis helps you model *what is known about a business* process that the system must support:

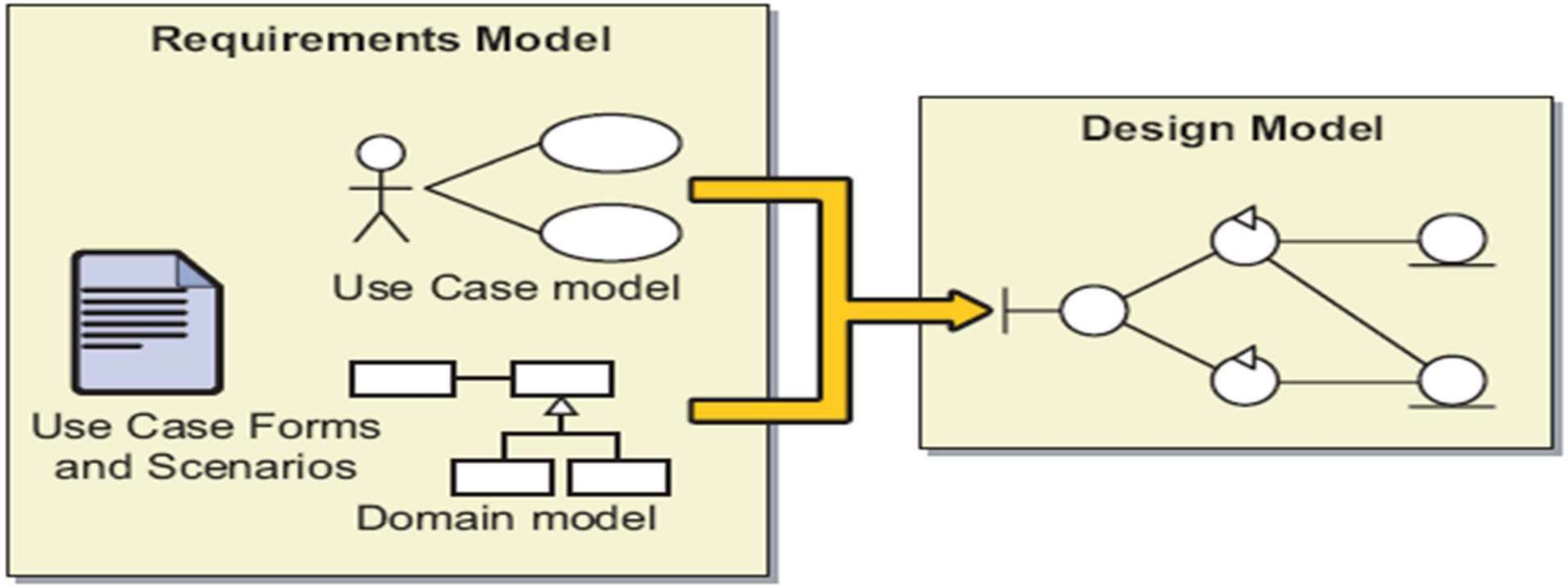
- Use cases
- Domain model

Design helps you model *how the system will support the* business processes. The Design model consists of:

- Boundary (UI) components
- Service components
- Entity components

Robustness Analysis

- Robustness analysis is a process that assists in identifying design components that would be required in the Design model:



Robustness Analysis

Inputs to Robustness Analysis:

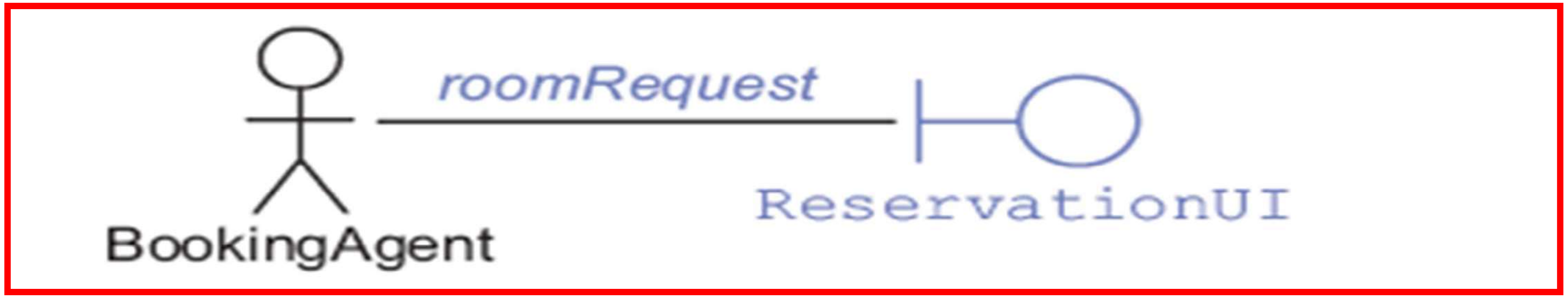
- A use case
- The use case scenarios for that use case
- The use case Activity diagram (if available) for that use case
- The Domain model

Output from Robustness Analysis:

- The Design model is usually captured in UML
- Interaction diagrams with design components such
- as Boundary, Service, and Entity components.

Boundary Components

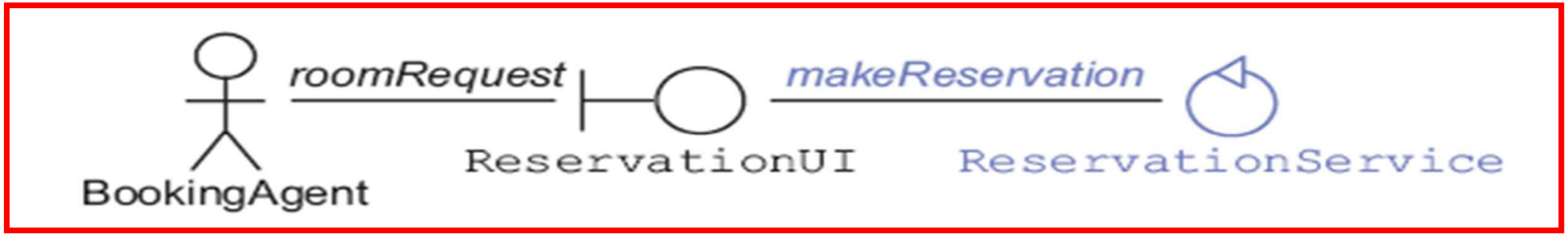
"A boundary class (component) is used to model interaction between the system and its actors (that is, users and external systems)."



- Abstractions of UI screens, sensors, communication interfaces, and so on.
- High-level UI components.
- Every boundary component must be associated with at least one actor.

Service Components

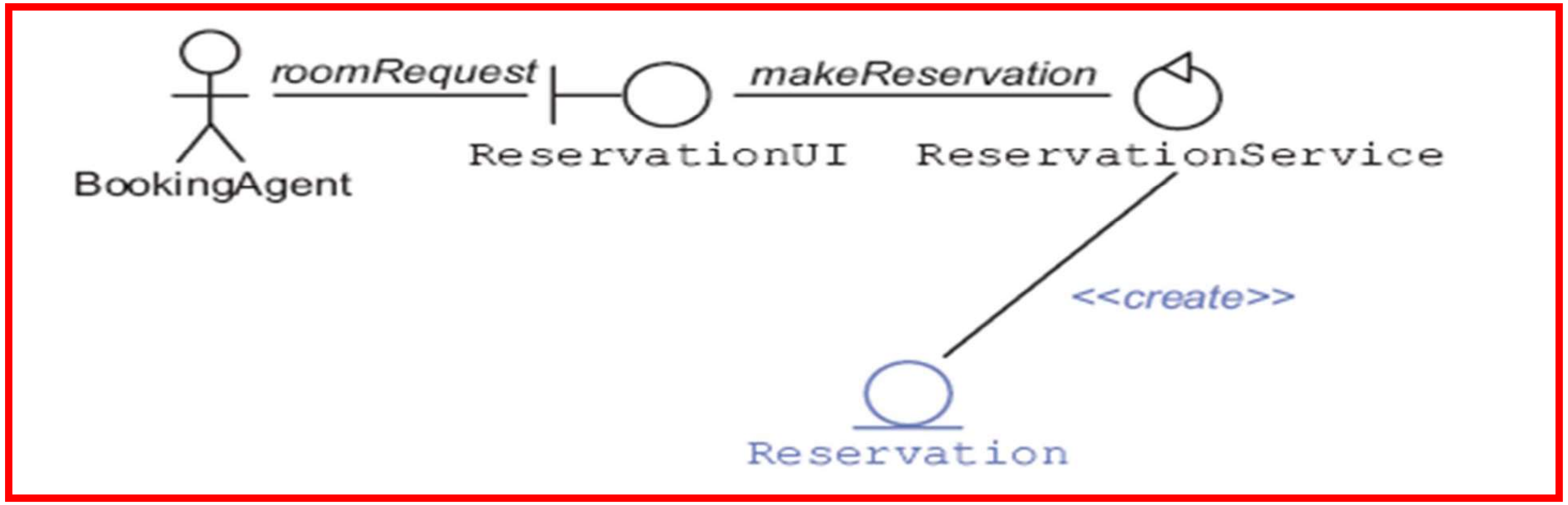
“Control (Service) classes (components) represent coordination, sequencing, transactions, and control of other objects and are often used to encapsulate control related to a specific use case.”



- Coordinate control flow
- Isolate any changes in workflow from the boundary and entity components

Entity Components

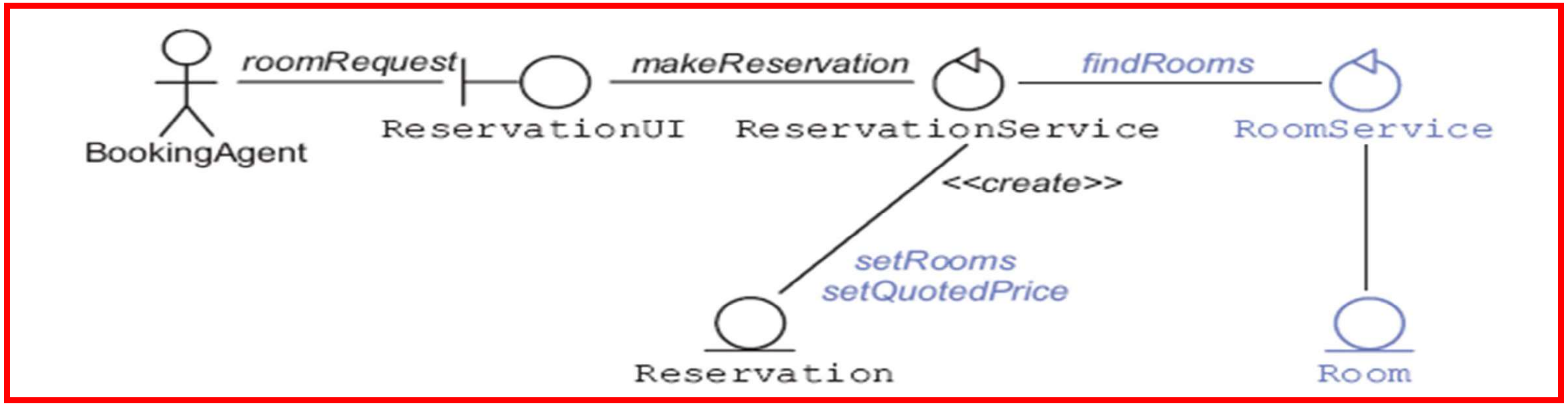
"An entity class (component) is used to model information that is long-lived and often persistent."



- Entities usually correspond to domain objects.
- Most entities are persistent.
- Entities can have complex behavior.

Service and Entity Components

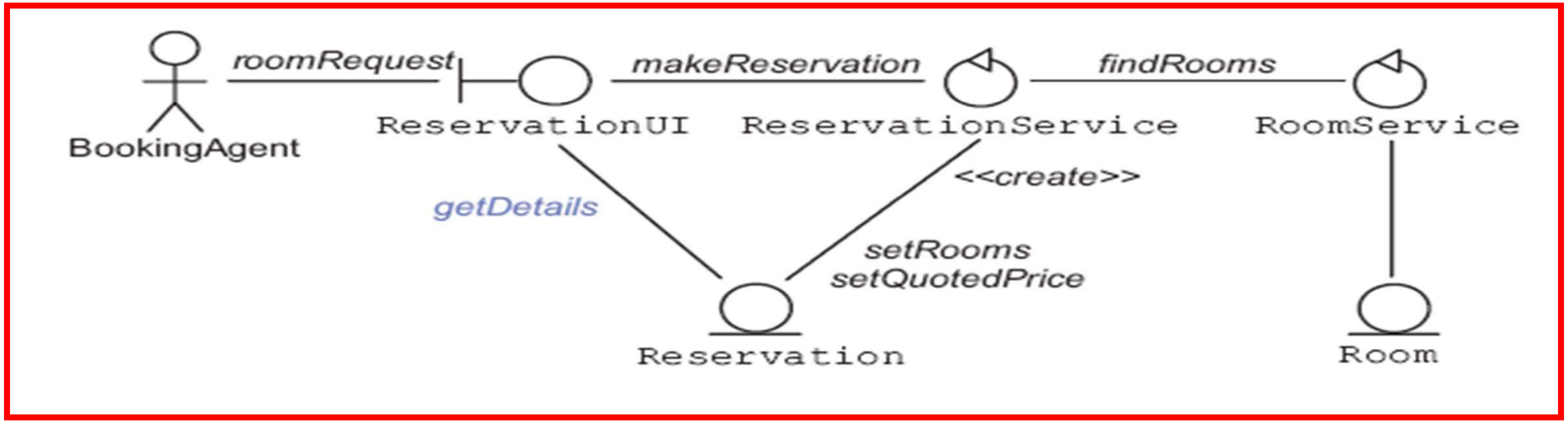
An Entity component will often have a corresponding Service component



- A service object will often control its corresponding entity object
- A service object can delegate to another service object

Boundary And Entity Components

- A Boundary component can often retrieve the attributes of an Entity component



Describing the Robustness Analysis

1. Select a use case.
2. Construct a Communication diagram or a Sequence diagram that satisfies the activities of the use case.

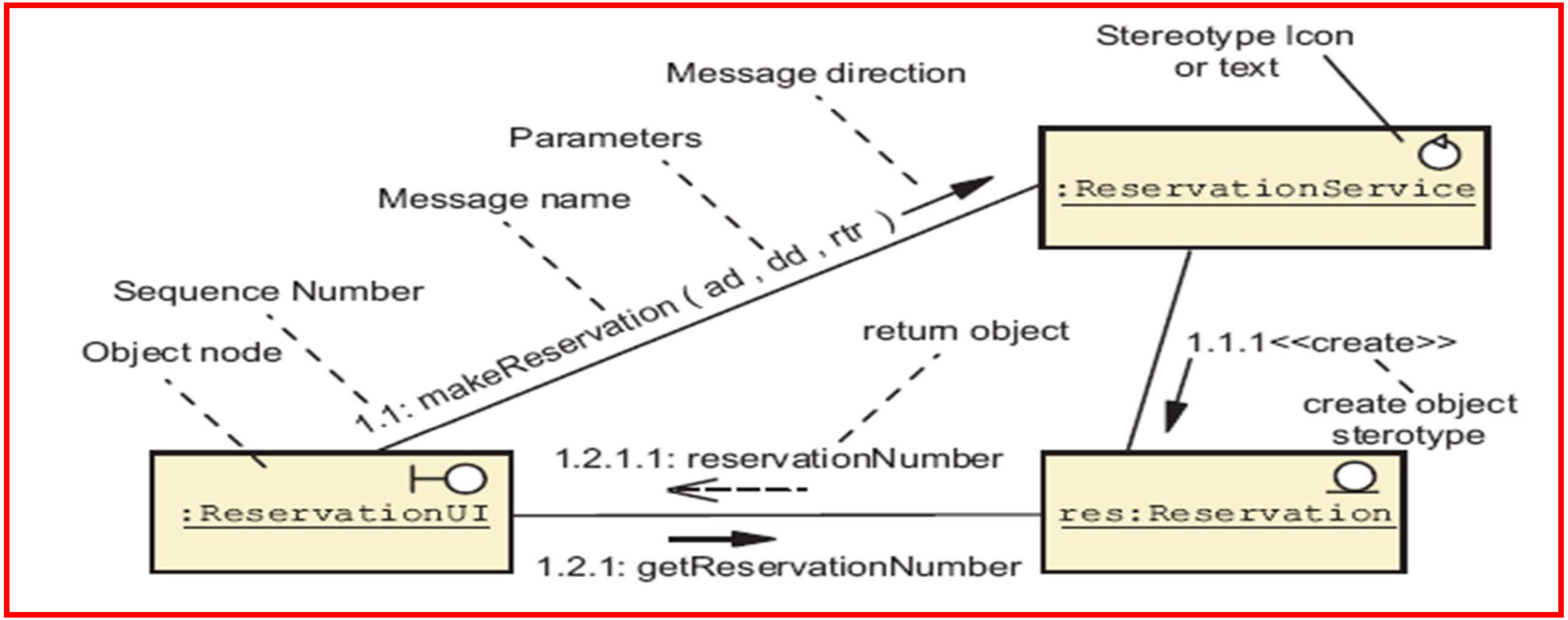
   a. Identify Design components that support the activities of the use case.

   b. Draw the associations between these components.

   c. Label the associations with messages.
3. Convert the Communication diagram into a Sequence diagram, or vice versa, for an alternate view (optional).

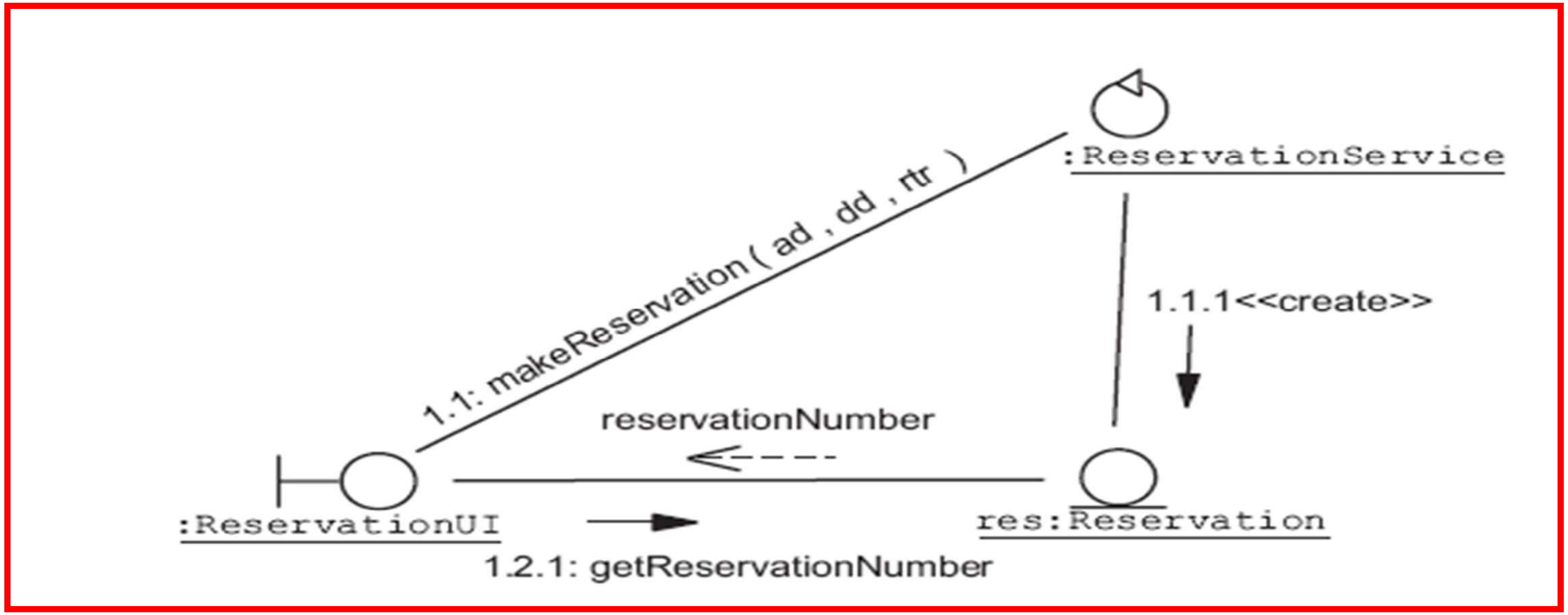
Identifying the Elements of a

- A UML Communication diagram is composed of the following elements:



Identifying the Elements of a

- A variation of the previous Communication diagram:



Identifying the Elements of a

A message can indicate:

- A message name
- A direction arrow
- An solid arrowhead is a synchronous message
- An open arrowhead is an asynchronous message
- A sequence number describing the order of the message
- A list of parameters passed to the receiving object
- A guard condition indicating a conditional message
- A return parameter

Creating a Communication Diagram

Select an appropriate use case.

1. Place the actor in the Communication diagram.
2. Analyze the Use Case form or the Activity diagram for the use case.

For every action in the use case:

a. Identify and add a Boundary component.

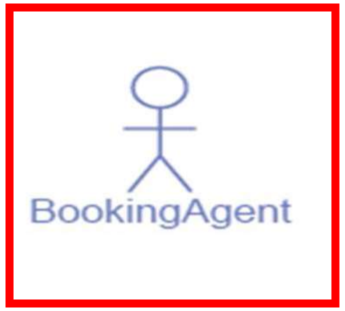
b. Identify and add a Service component.

c. Identify and add an Entity component.

d. Identify and add further Interactions, discovering new Methods, Boundary, Service and Entity components.

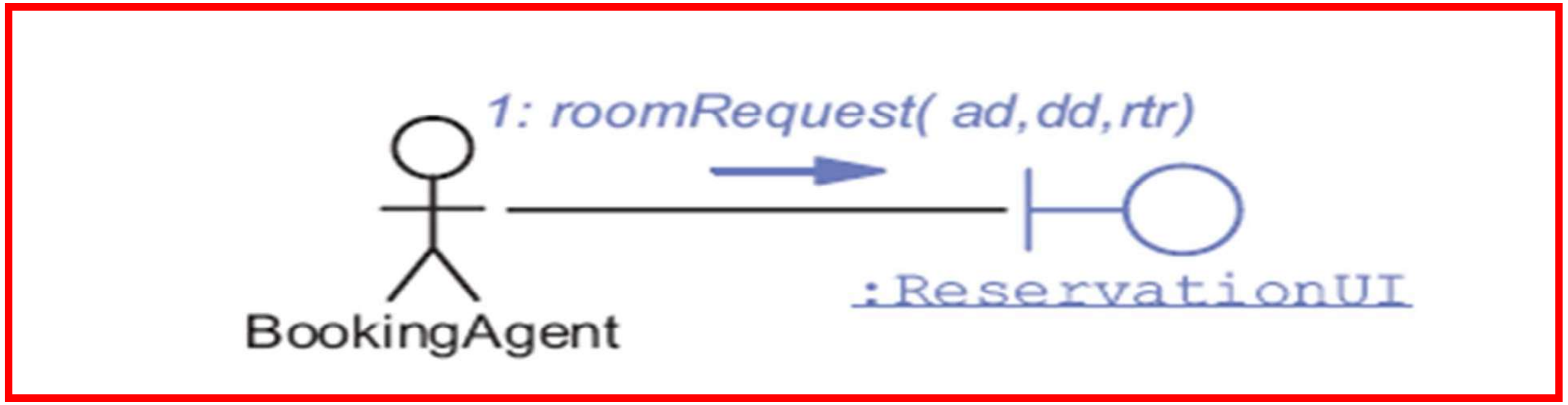
Step 1— Place the Actor in the Diagram

- Place the actor in the Communication diagram:



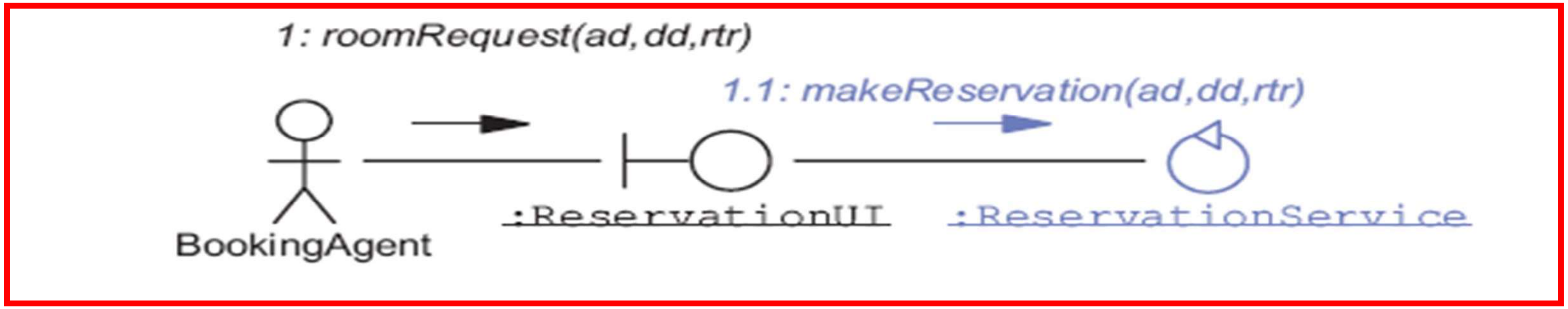
Step 2a—Identify Boundary Components

- BookingAgent makes a room request passing the arrival date (ad), departure date (dd), requested types of room (rtr):



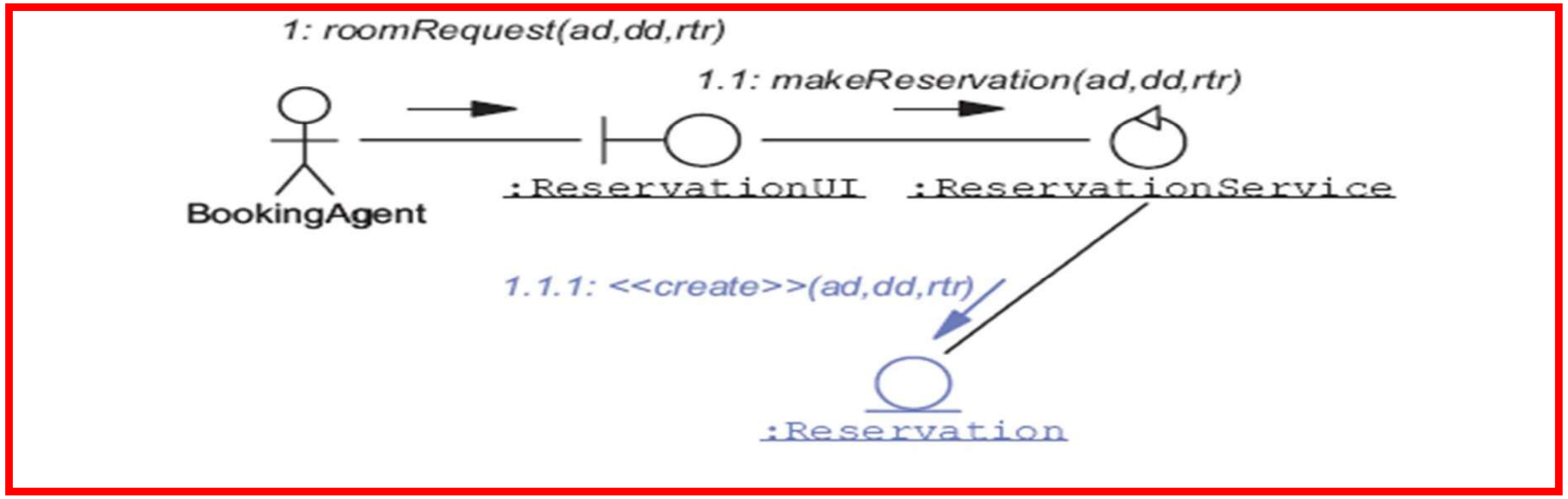
Step 2b—Identify Service Components

- The ReservationUI boundary object uses ReservationService object to make the Reservation:



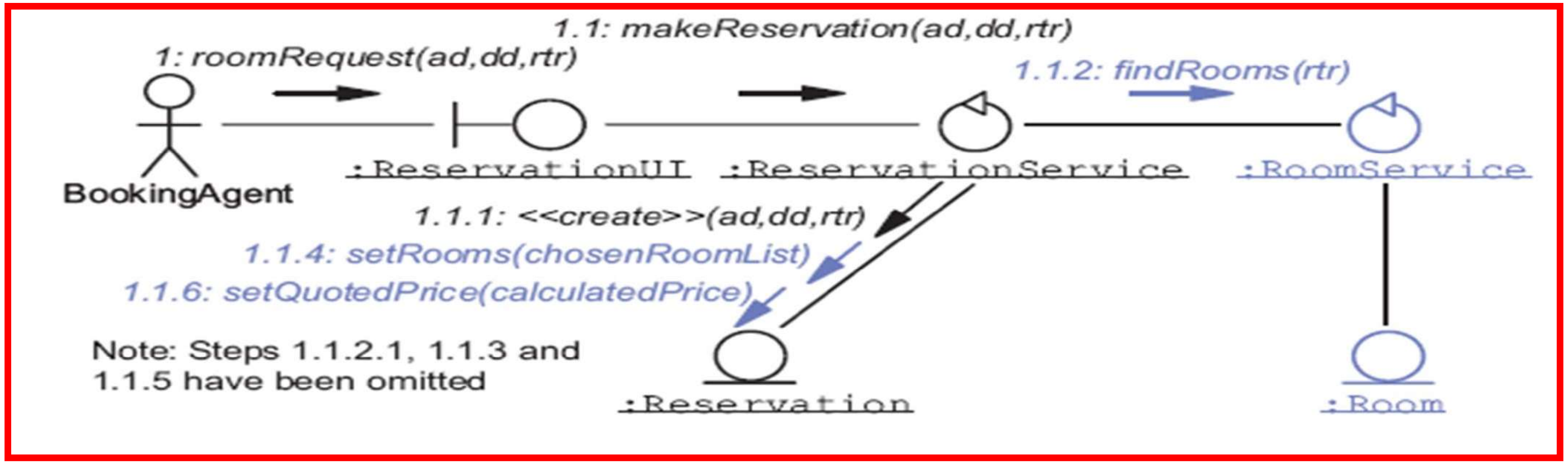
Step 2c—Identify Entity Components

- The makeResevation method in the ReservationService object creates the Reservation entity object:


1: roomRequest(ad,dd,rtr)
1.1: makeReservation(ad,dd,rtr)
BookingAgent
:ReservationUI
:ReservationService
1.1.1: <<create>>(ad,dd,rtr)
:Reservation

Step 2d—Identify Additional Interactions

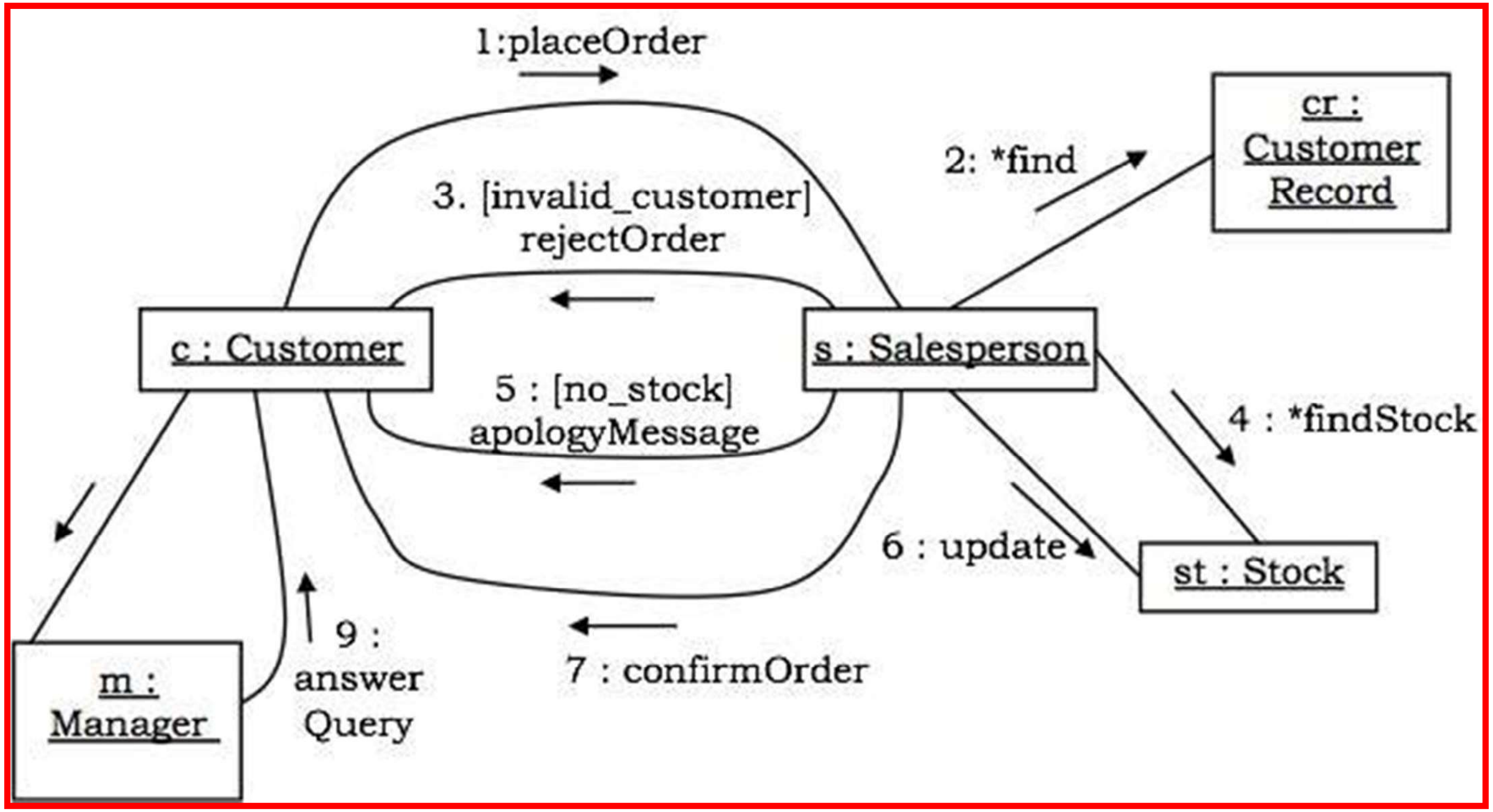
- The makeReservation method uses the findRoom method to find rooms of the required type. After finding and choosing free rooms (not shown) the setRooms method assigns the rooms to the Reservation. The calculated price (not shown) is assigned to the Reservation:


1.1: makeReservation(ad,dd,rtr)
1: roomRequest(ad,dd,rtr)
1.1.2: findRooms(rtr)
BookingAgent
:ReservationUI
:ReservationService
:RoomService
1.1.1: <<create>>(ad,dd,rtr)
1.1.4: setRooms(chosenRoomList)
1.1.6: setQuotedPrice(calculatedPrice)
Note: Steps 1.1.2.1, 1.1.3 and
1.1.5 have been omitted
:Reservation
:Room

Collaboration Diagrams

- Collaboration diagrams are interaction diagrams that illustrate the structure of the objects that send and receive messages.
- **Notations :**
  - In these diagrams, the objects that participate in the interaction are shown using vertices.
  - The links that connect the objects are used to send and receive messages.
  - The message is shown as a labeled arrow.

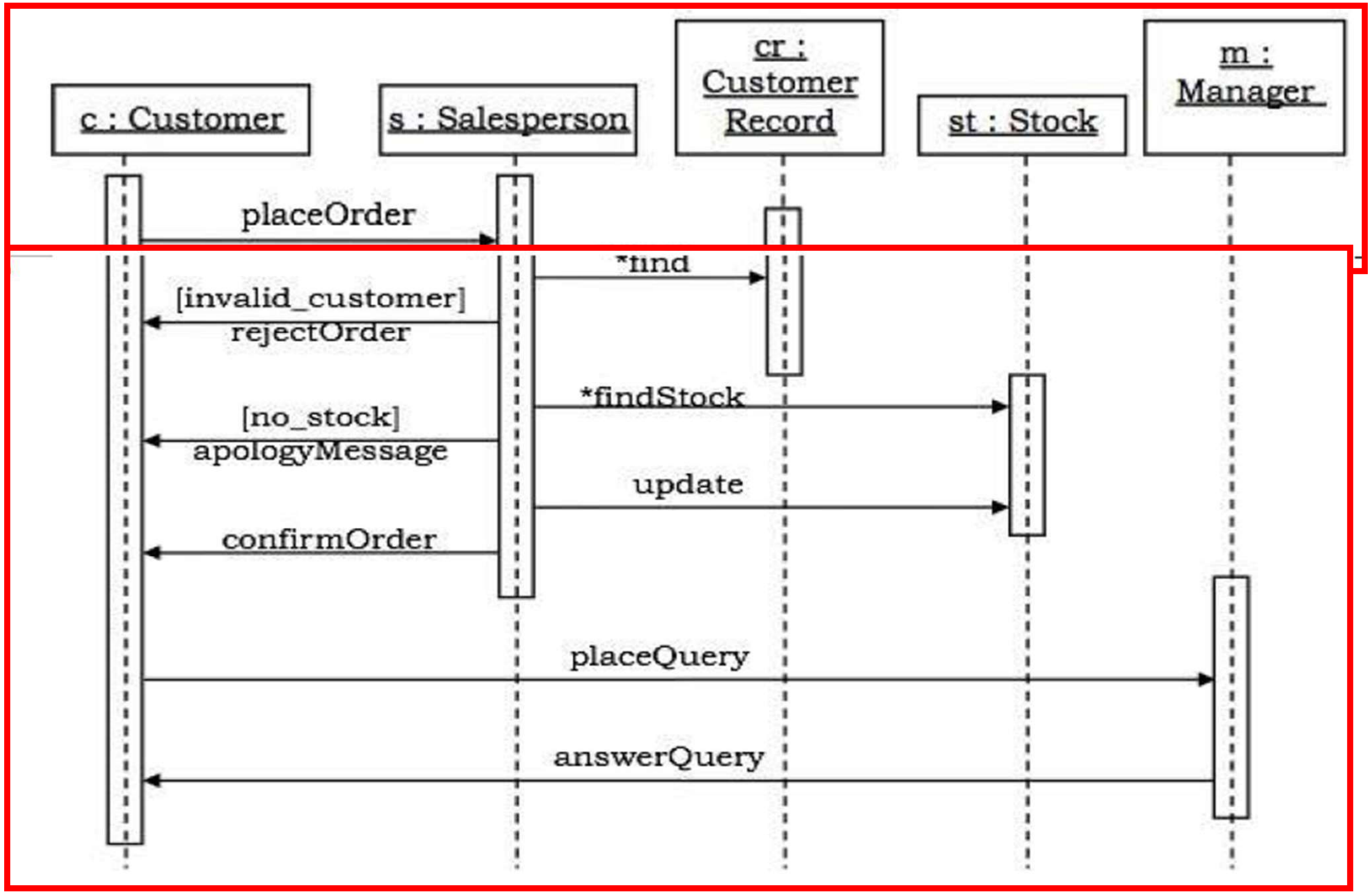
Example


1:placeOrder
cr : Customer Record
2: *find
3. [invalid_customer] rejectOrder
c : Customer
s : Salesperson
5 : [no_stock] apologyMessage
4 : *findStock
6 : update
st : Stock
9 : answer Query
7 : confirmOrder
m : Manager

Sequence Diagrams

- Sequence diagrams are interaction diagrams that illustrate the ordering of messages according to time.
- **Notations :**
  - These diagrams are in the form of two-dimensional charts.
  - The objects that initiate the interaction are placed on the x–axis.
  - The messages that these objects send and receive are placed along the y–axis, in the order of increasing time from top to bottom.

Example


c : Customer
s : Salesperson
cr : Customer Record
st : Stock
m : Manager
placeOrder
*find
[invalid_customer] rejectOrder
*findStock
[no_stock] apologyMessage
update
confirmOrder
placeQuery
answerQuery

State–Chart Diagrams

State–Chart Diagrams comprise of:

- States: Simple or Composite
- Transitions between states
- Events causing transitions
- Actions due to the events

- State-chart diagrams are used for modeling objects which are reactive in nature.

Introducing Object State

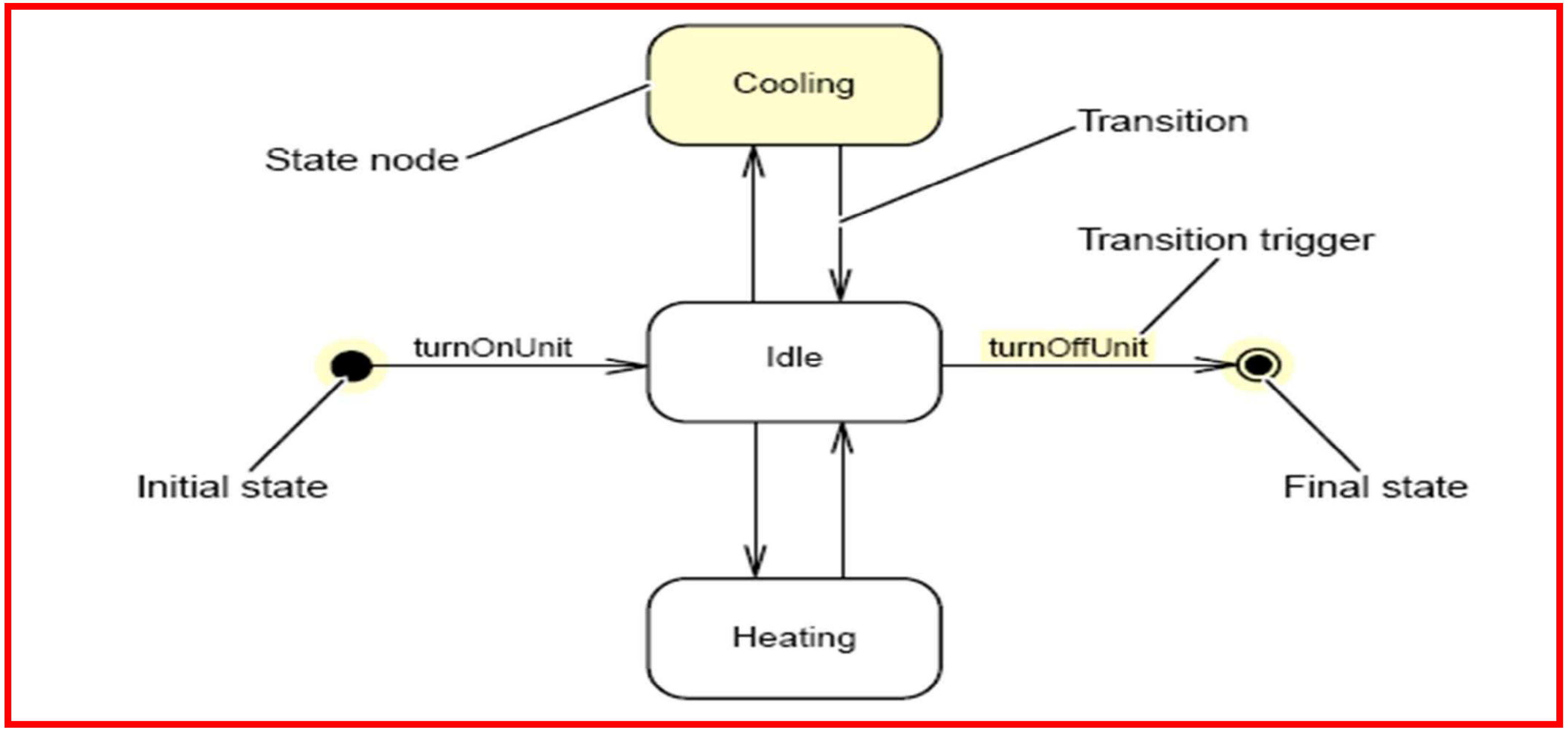
State is “1a: mode or condition of being”

There are two ways to think about object state:

- The state of an object is specific collection of attribute values for the object.
- The state of an object describes the behavior of the object relative to external stimuli.

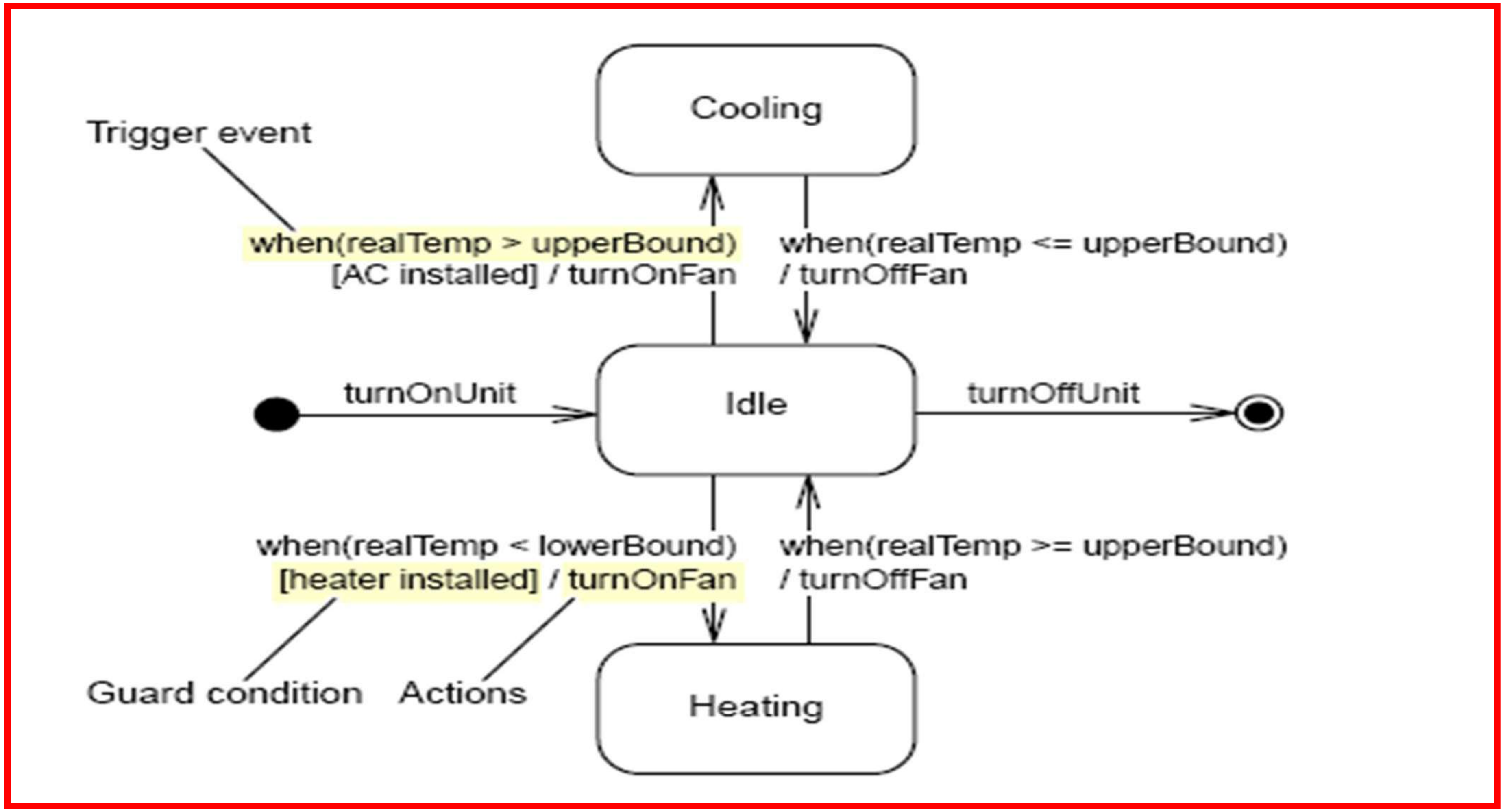
- This module considers the second definition.

Identifying the Elements of a State Machine



State Transitions

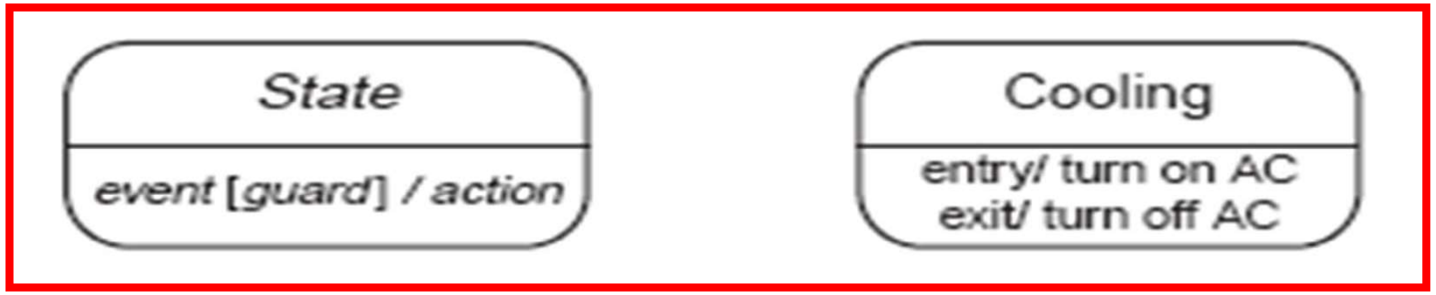
- A state transition represents a change of state at runtime.



Cooling
Trigger event
when(realTemp > upperBound)
[AC installed] / turnOnFan
when(realTemp <= upperBound)
/ turnOffFan
turnOnUnit
Idle
turnOffUnit
when(realTemp < lowerBound)
[heater installed] / turnOnFan
when(realTemp >= upperBound)
/ turnOffFan
Guard condition
Actions
Heating

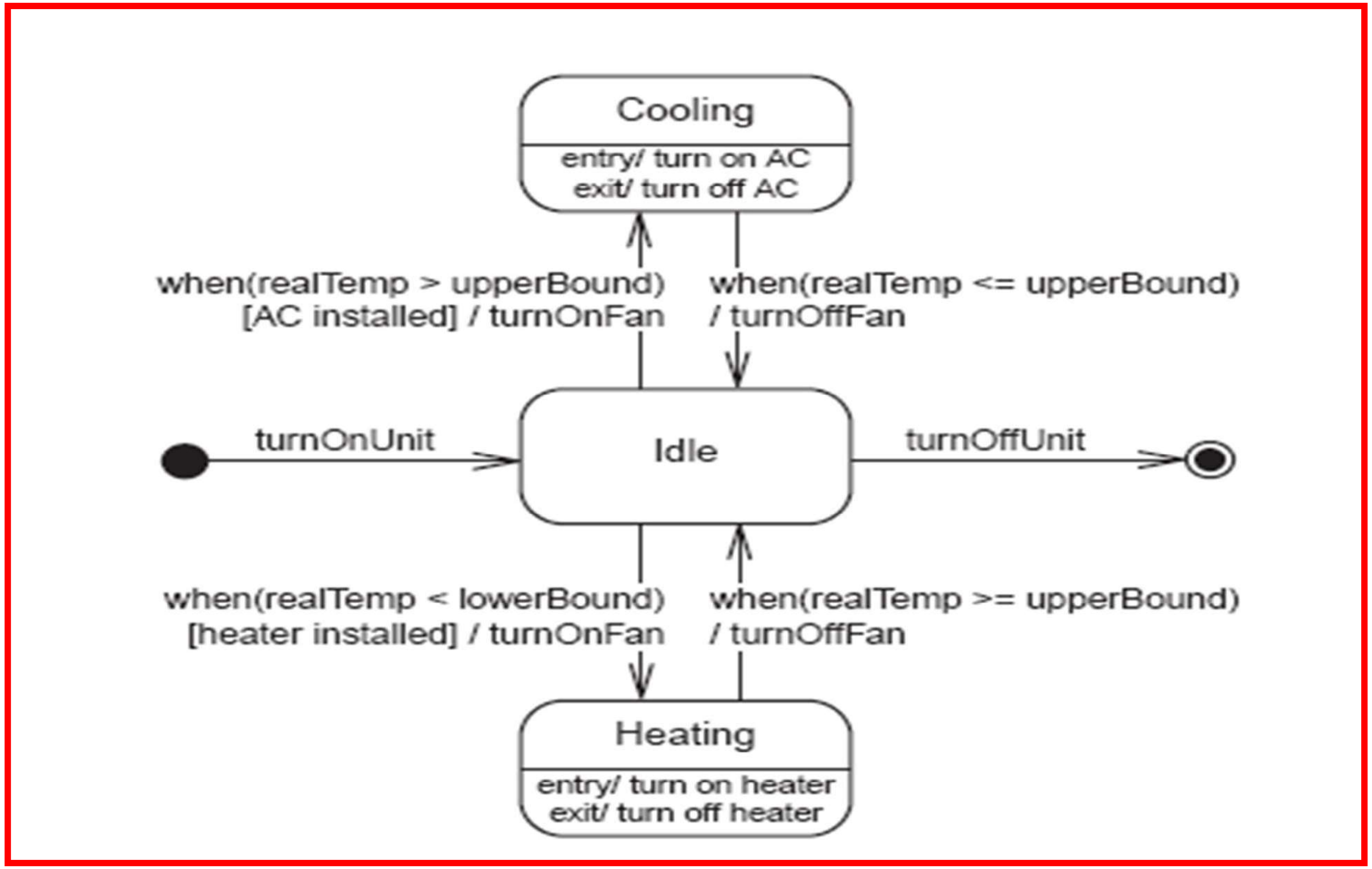
Internal Structure of State Nodes

State nodes represents a state of a single object at runtime.



- “Entry” event specifies actions upon entry into the state.
- “Exit” event specifies actions upon exit from the state.
- “Do” event specifies ongoing actions.
- You can also specify specific events with corresponding actions.

Complete HVAC State Machine Diagram

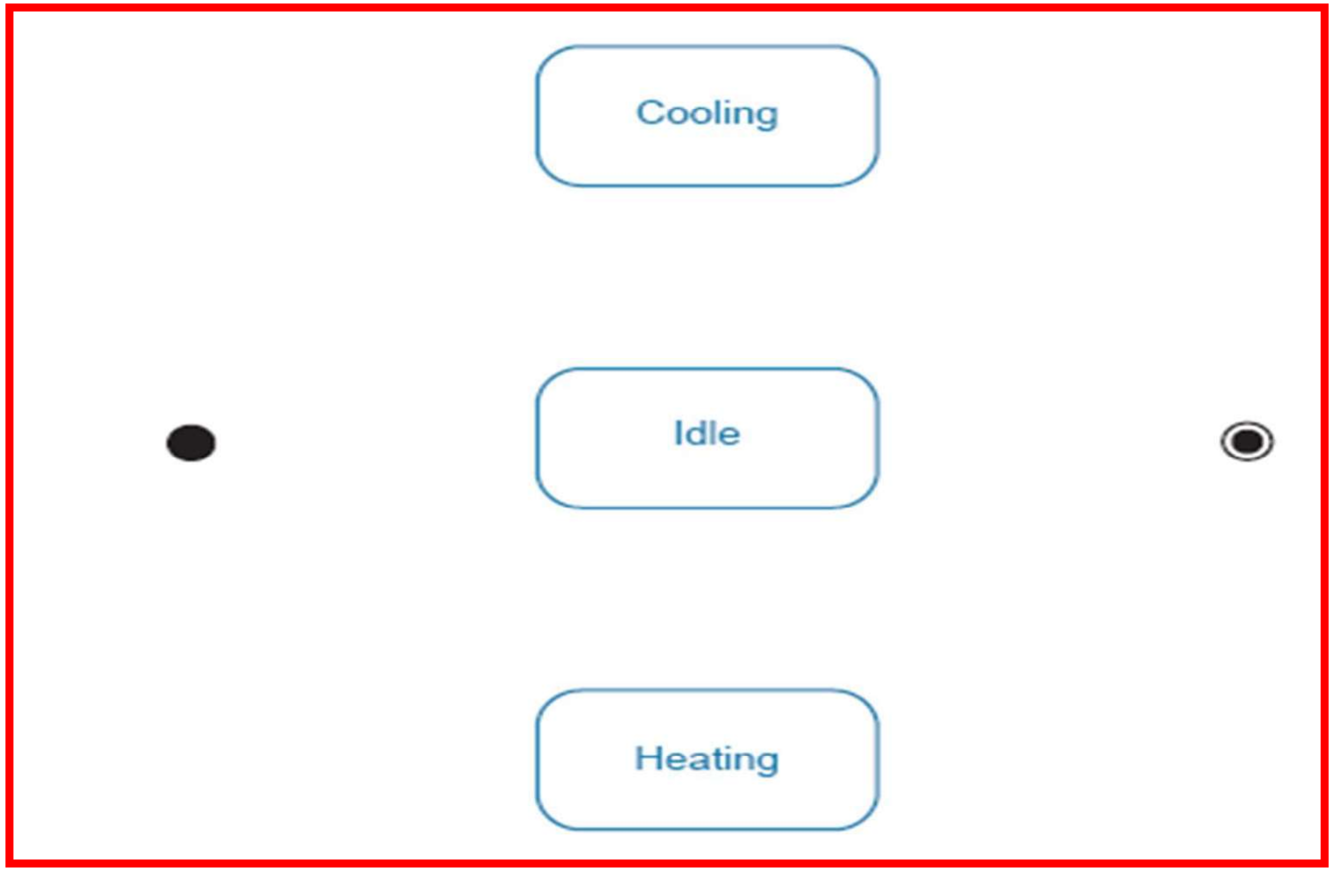

Cooling
entry/ turn on AC
exit/ turn off AC
when(realTemp > upperBound)
[AC installed] / turnOnFan
when(realTemp <= upperBound)
/ turnOffFan
turnOnUnit
Idle
turnOffUnit
when(realTemp < lowerBound)
[heater installed] / turnOnFan
when(realTemp >= upperBound)
/ turnOffFan
Heating
entry/ turn on heater
exit/ turn off heater

Creating a State Machine Diagram for a

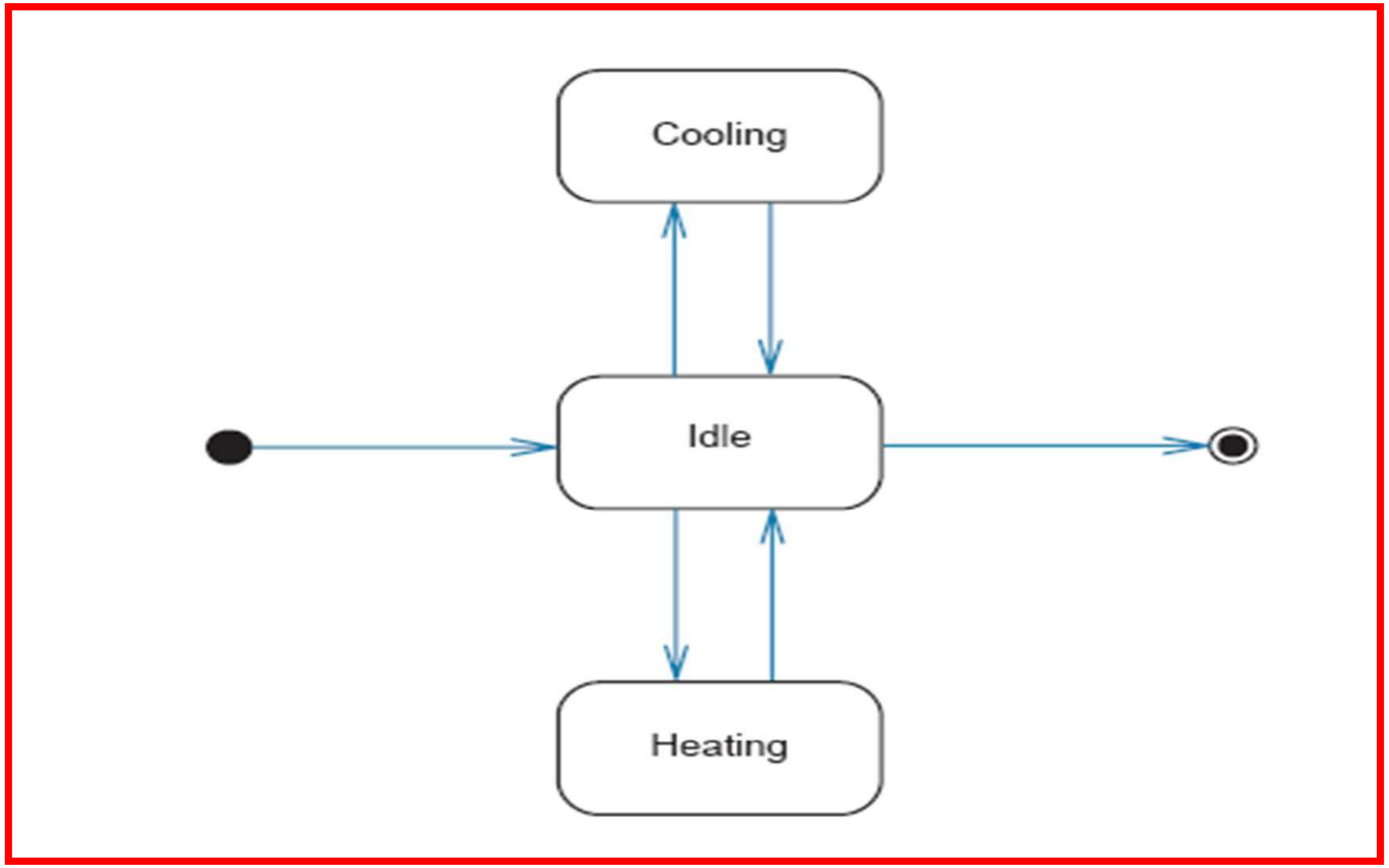
1. Draw the initial and final state for the object.
2. Draw the stable states of the object.
3. Specify the partial ordering of stable states over the lifetime of the object.
4. Specify the events that trigger the transitions between the states of the object. Specify transition actions (if any).
5. Specify the actions within a state (if any).

Step 1 – Start With the Initial and Final

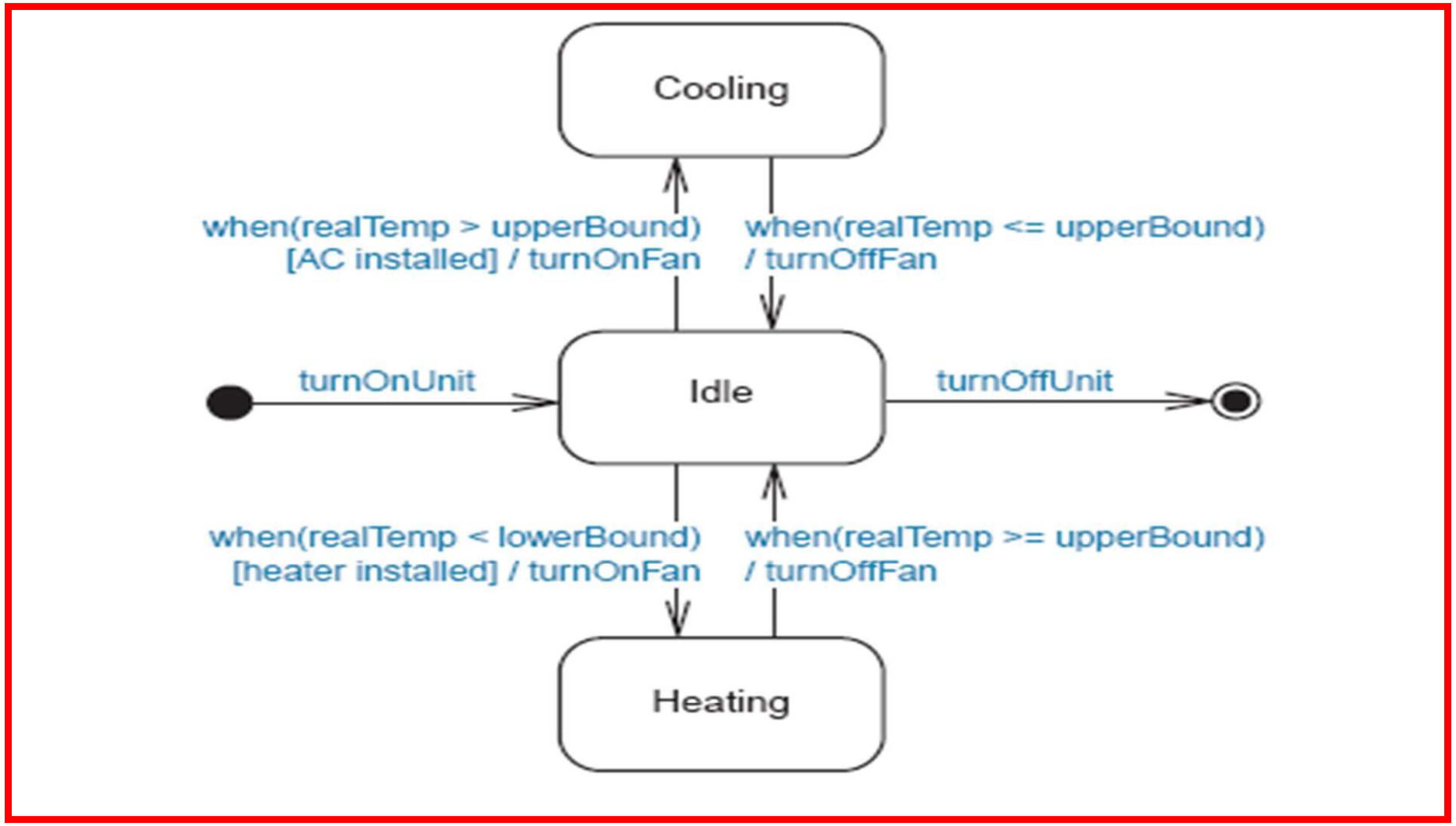
Step 2 – Determine Stable Object States



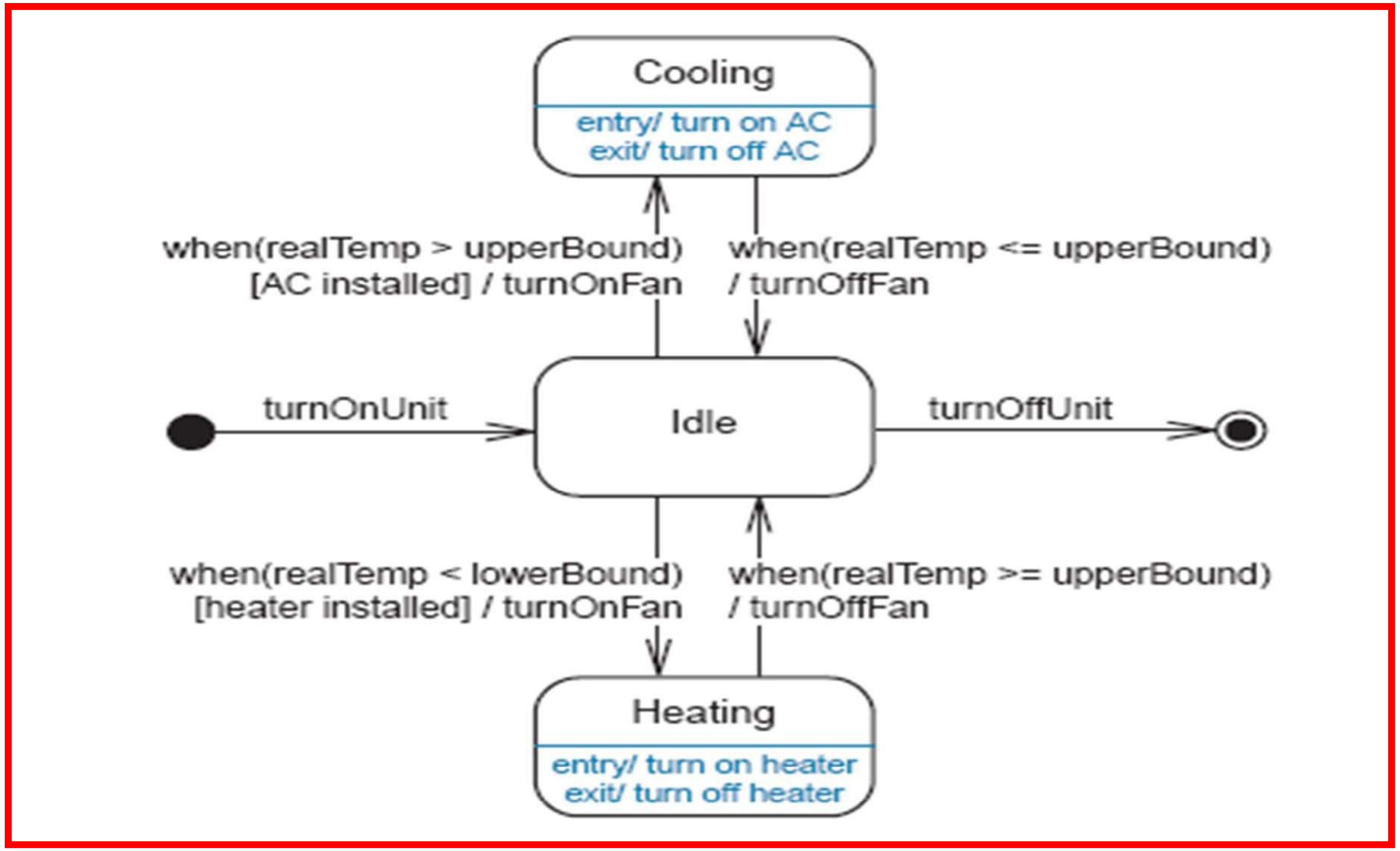
Step 3 – Specify the Partial Ordering of



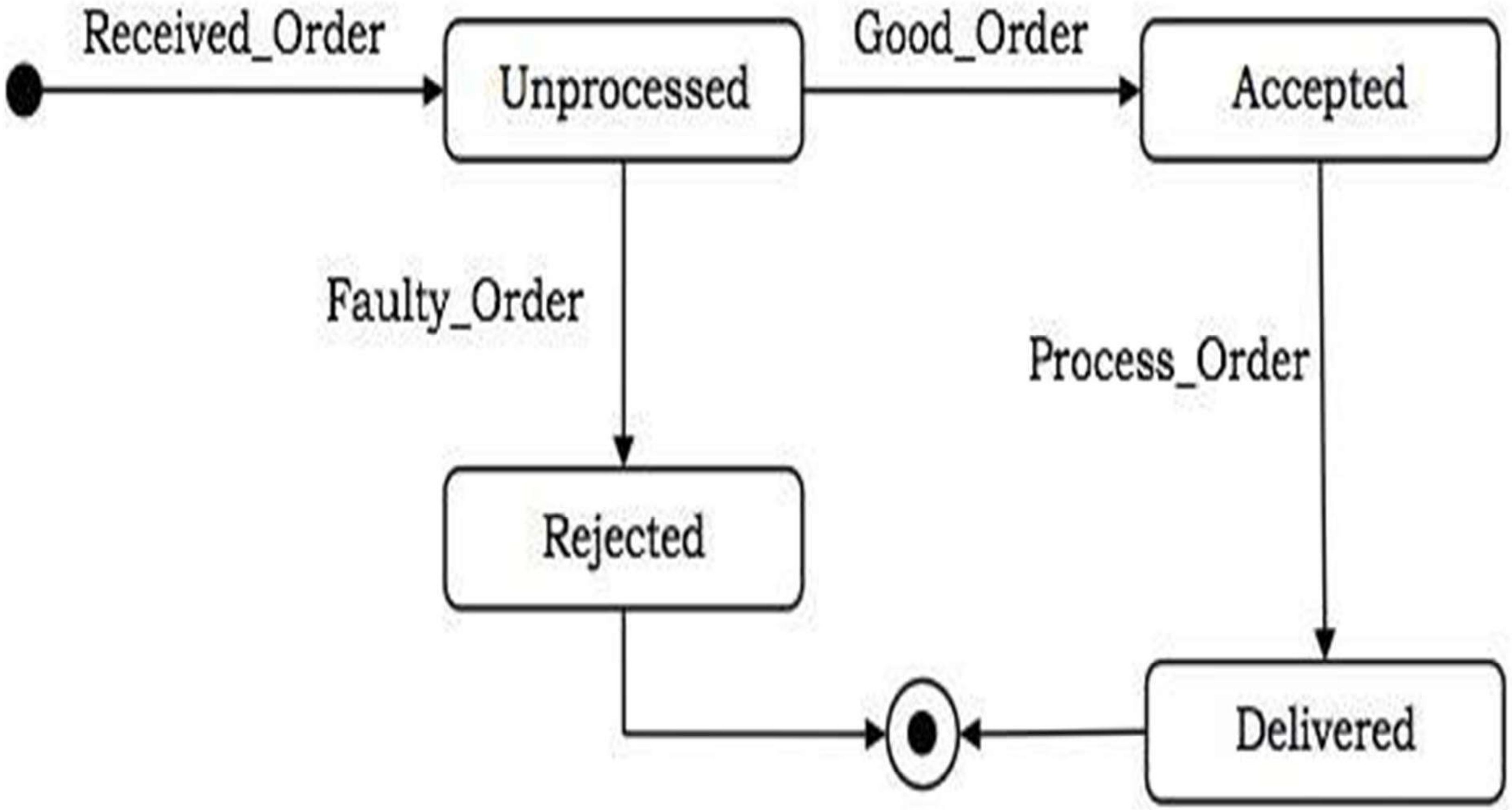
Step 4 – Specify the Transition Events and


Cooling
when(realTemp > upperBound)
[AC installed] / turnOnFan
when(realTemp <= upperBound)
/ turnOffFan
turnOnUnit
Idle
turnOffUnit
when(realTemp < lowerBound)
[heater installed] / turnOnFan
when(realTemp >= upperBound)
/ turnOffFan
Heating

Step 5 – Specify the Actions Within a State


Cooling
entry/ turn on AC
exit/ turn off AC
when(realTemp > upperBound)
[AC installed] / turnOnFan
when(realTemp <= upperBound)
/ turnOffFan
turnOnUnit
Idle
turnOffUnit
when(realTemp < lowerBound)
[heater installed] / turnOnFan
when(realTemp >= upperBound)
/ turnOffFan
Heating
entry/ turn on heater
exit/ turn off heater

Example


Received_Order
Unprocessed
Good_Order
Accepted
Faulty_Order
Process_Order
Rejected
Delivered

Summary

In this lesson, you should have learned the following:

- Explain the purpose and elements of the Design model
- Identify the essential elements of a UML Communication diagram
- Create a Communication diagram view of the Design model
- Identify the essential elements of a UML Sequence diagram
- Create a Sequence diagram view of the Design model

Practice : Overview

This practice covers the following topics:

- Working with Use Case Diagrams