ACS2913
Software Requirements Analysis and Design

Instructor: Victor Balogun

OBJECT-ORIENTED DESIGN WITH INTERACTION DIAGRAMS

Chapter 13 Outline

Object-Oriented Design with Interaction Diagrams

Use Case Realization with Communication Diagrams

Use Case Realization with Sequence Diagrams

Developing a Multilayer Design

Updating and Packaging the Design Classes

Design Patterns

Learning Objectives

Explain the different types of objects and layers in a design

Develop communication diagrams for use case realization

Develop sequence diagrams for use case realization

Develop updated design class diagrams

Develop multilayer subsystem packages

Explain design patterns and recognize various specific patterns

OOD with Interaction Diagrams

CRC Cards focuses on the business logic, also known as problem domain layer of classes

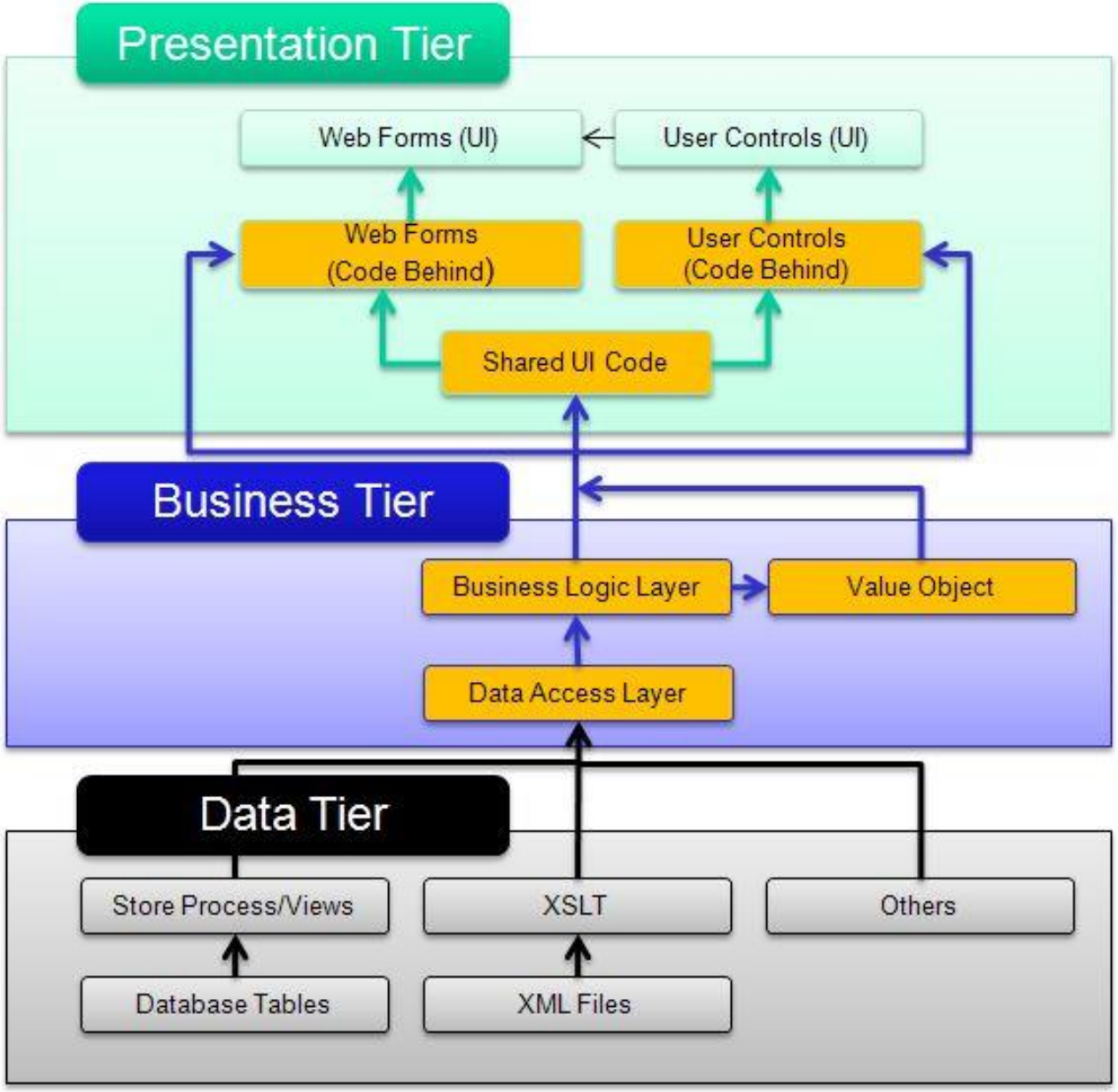
Three layers include

- view layer,
- business logic/problem domain layer, and
- data access layer

Questions that come up include

- How do objects get created in memory?
- How does the user interface interact with other objects?
- How are objects handled by the database?
- Will other objects be necessary?
- What is the lifespan of each object?

3-Layer in C#.Net



OOD with Interaction Diagrams

Use case realization

- The process of elaborating the detailed design for a particular use case using interaction diagrams

Communication diagram

- A type of interaction diagram which emphasizes the set of objects involved in a use case

Sequence diagram

- A type of interaction diagram which emphasizes the sequence of messages involved in a use case

Use Case Controller

Switchboard between user-interface classes and domain layer classes

Reduces coupling between view and domain layer

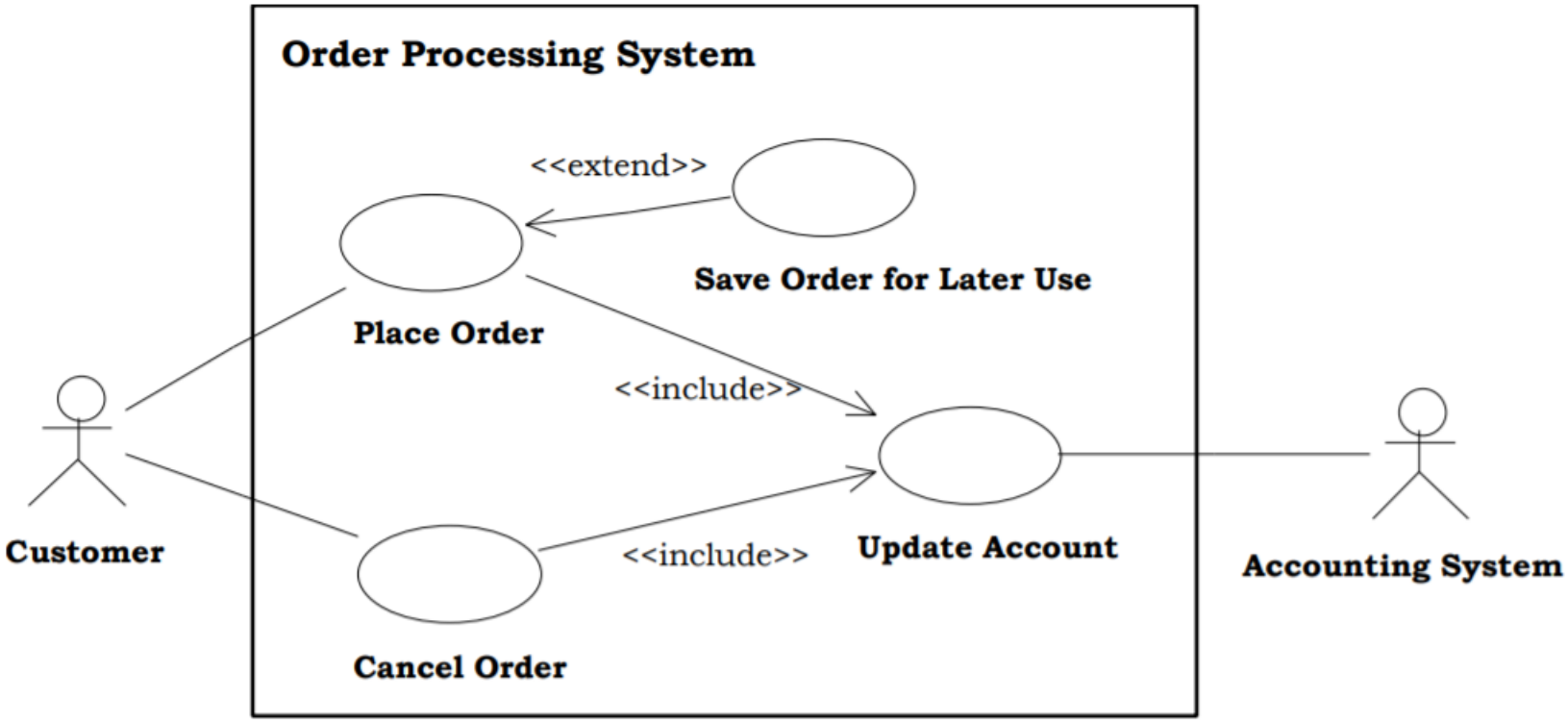
A controller can be created for each use case, however, several controllers can be combined together for a group of related use cases

It is a completely artificial class – an artifact

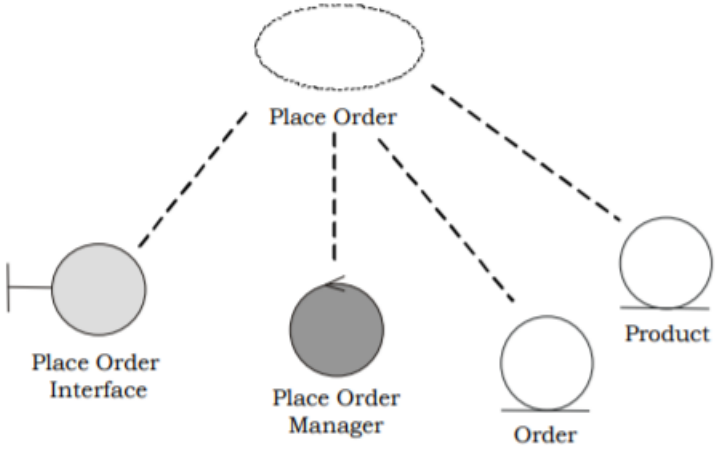
Use-case controllers provide a uniform way of coordinating actor events, user interfaces and system services.

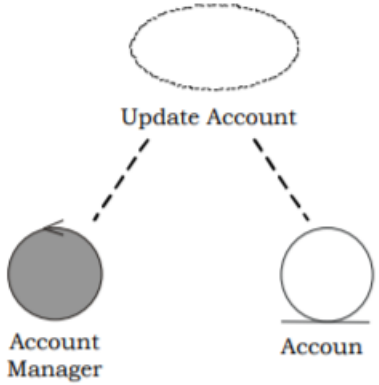
Use Case Controller

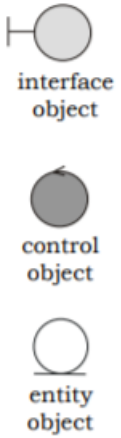
Consider this use case:

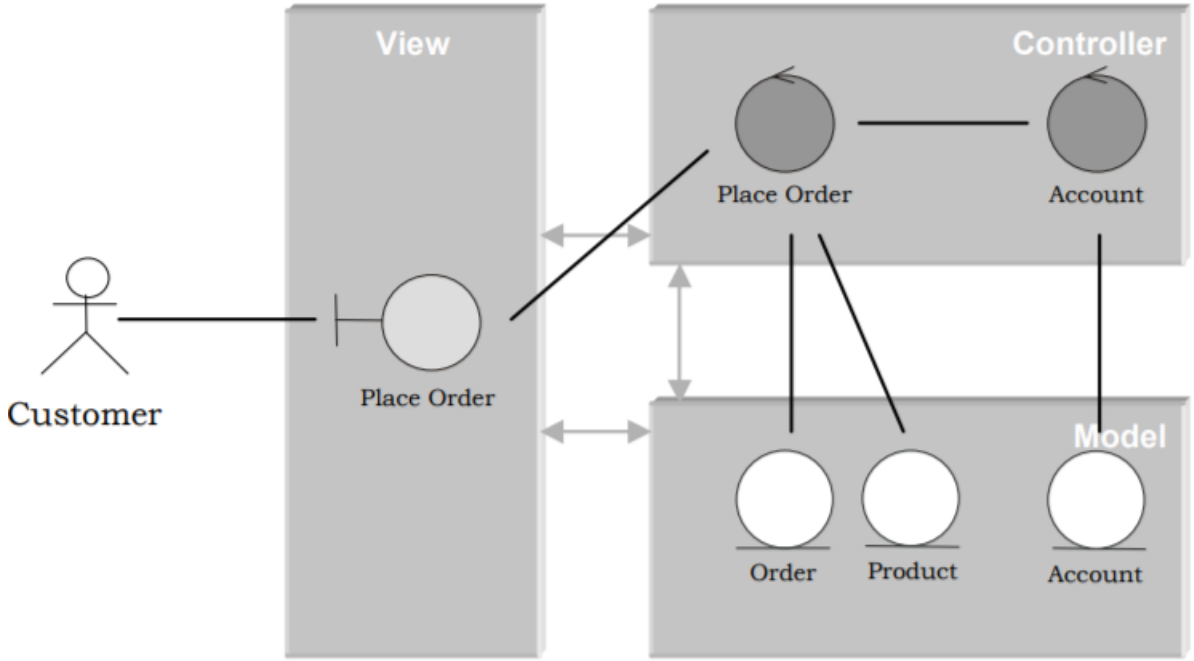


Use-Case Realization









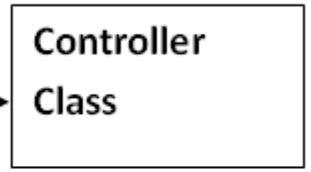
Controller Class in VB.Net

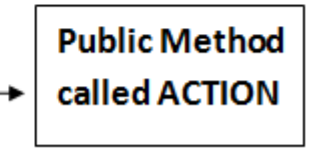
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace MvcApplication1.Controllers
{
    [HandleError]
    public class HomeController : Controller
    {
        public ActionResult Index()
        {
            ViewData["Message"] = "Welcome to ASP.NET MVC!";

            return View();
        }

        public ActionResult About()
        {
            return View();
        }
    }
}
```





Interaction Diagrams

interaction diagrams are used to illustrate how objects interact via messages.

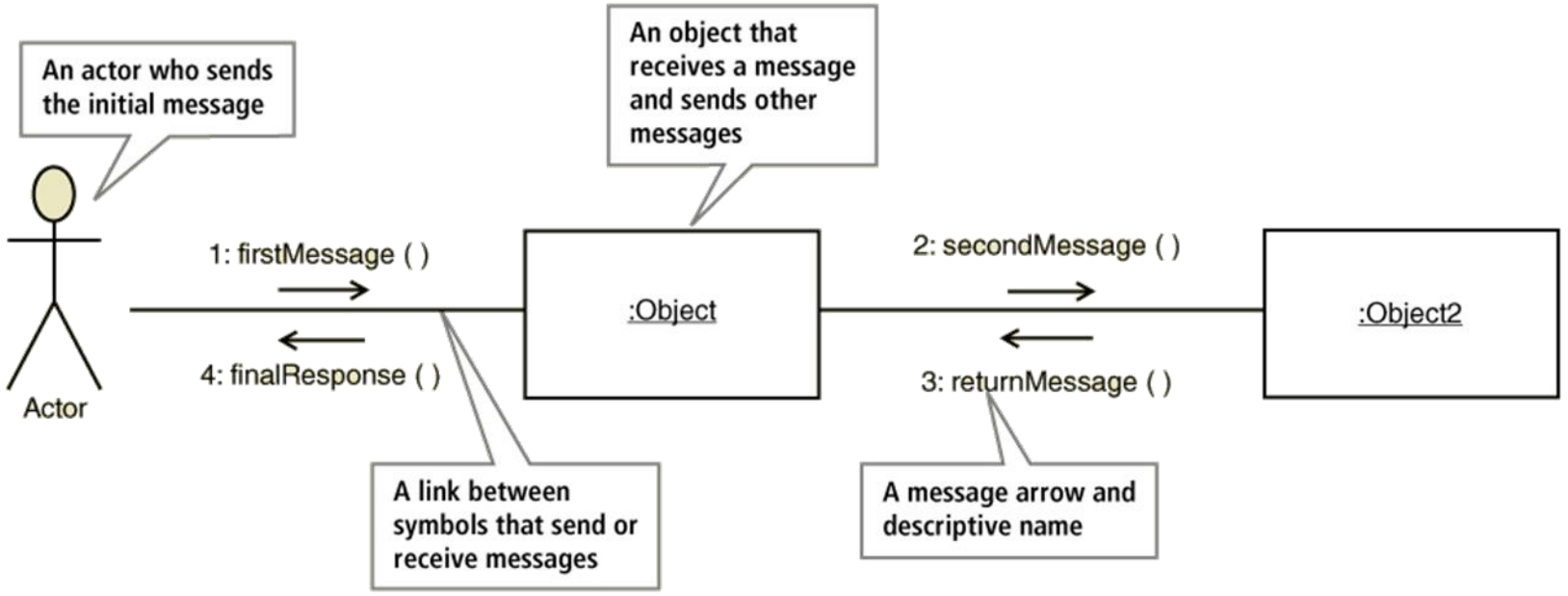
They are used for dynamic object modeling.

There are two common types:

- sequence and
- communication interaction diagrams.

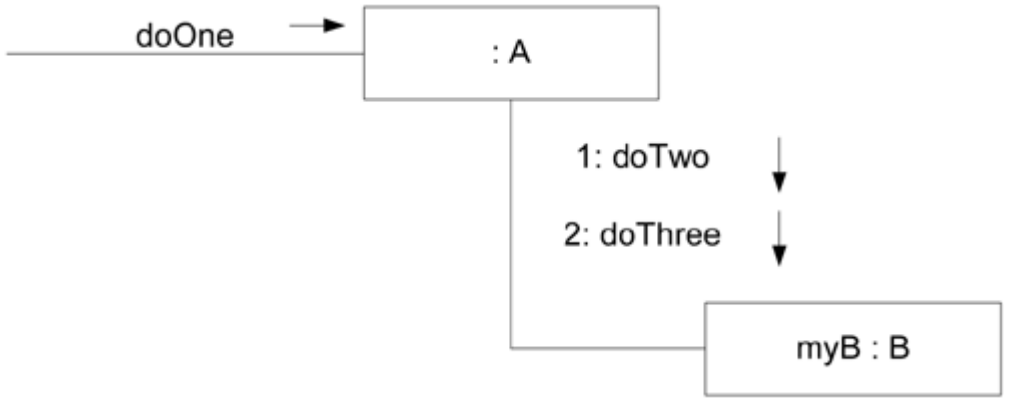
Communication Diagrams

Communication diagrams show the message flow between objects in an OO application and also imply the basic associations (relationships) between classes



Communication Diagrams

They Illustrate object interactions in a graph or network format, in which objects can be placed anywhere on the diagram




```
public class A
  {
  private B myB = new B();

  public void doOne()
  {
  myB.doTwo();
  myB.doThree();
  }
  // . .
  }
```

Understanding Communication Diagrams

Actor – the external role of the person or thing that initiates the use case. Sends messages.

Object – the instantiated class objects that perform the actions (methods) to execute the use case. They receive messages and process messages.

Link – simply connectors between objects to carry the messages.

Message – the requests for service with an originating actor or object and a destination object, which performs the requested service

Message Syntax

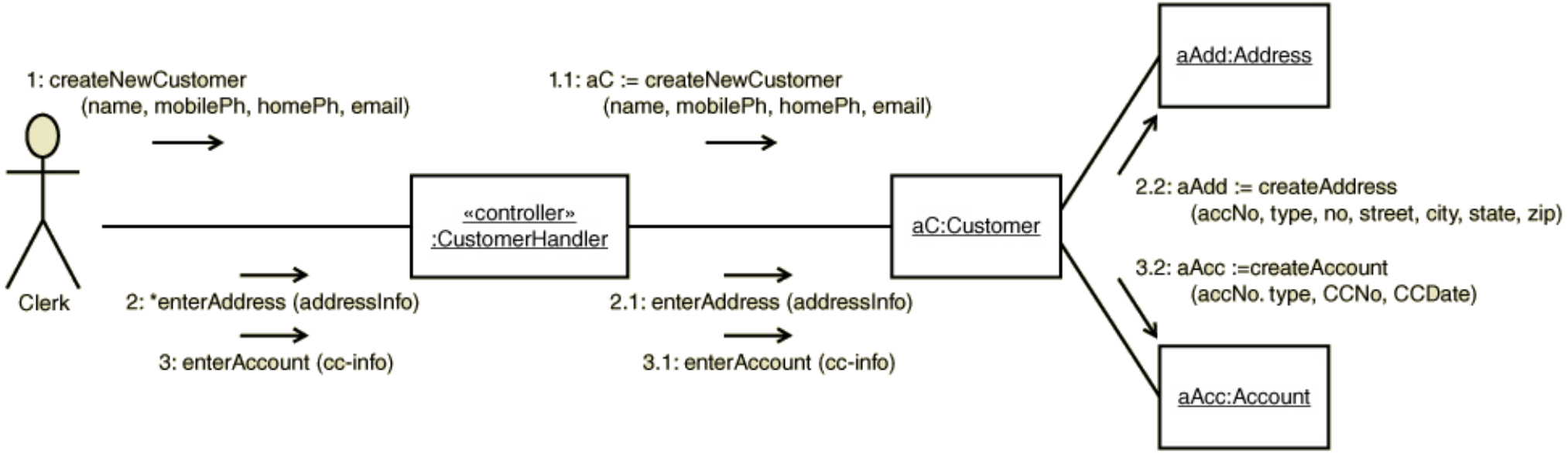
[true/false condition] sequence-number: return-value: = message-name (parameter-list)

- true/false condition – determines if message is sent
- sequence number – notes the order of the messages
- return-value – value coming back to origin object from the destination object
- message-name – camelCase name identifier. Reflects the requested service
- parameter-list – the arguments being passed to the destination object

Every element of a message is optional

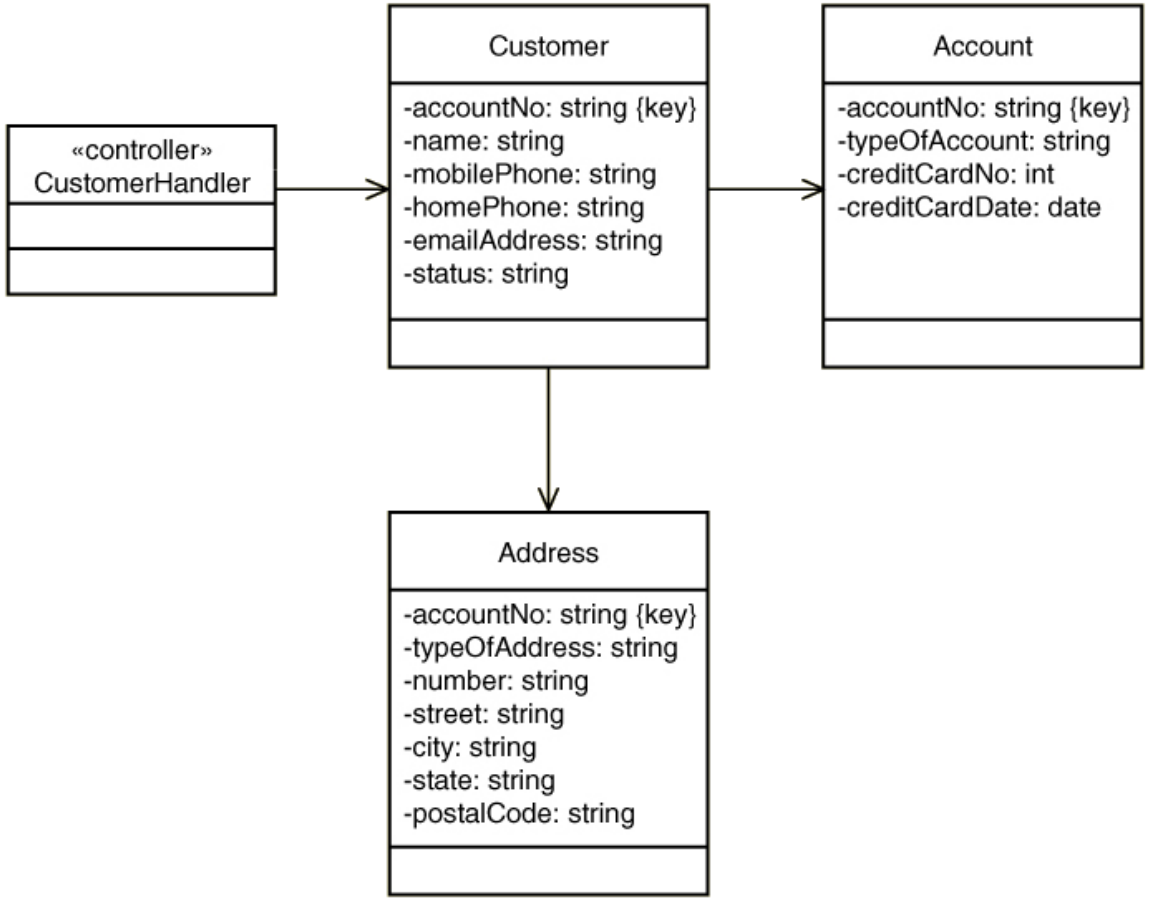
Example Communication Diagram:

Create customer account use case



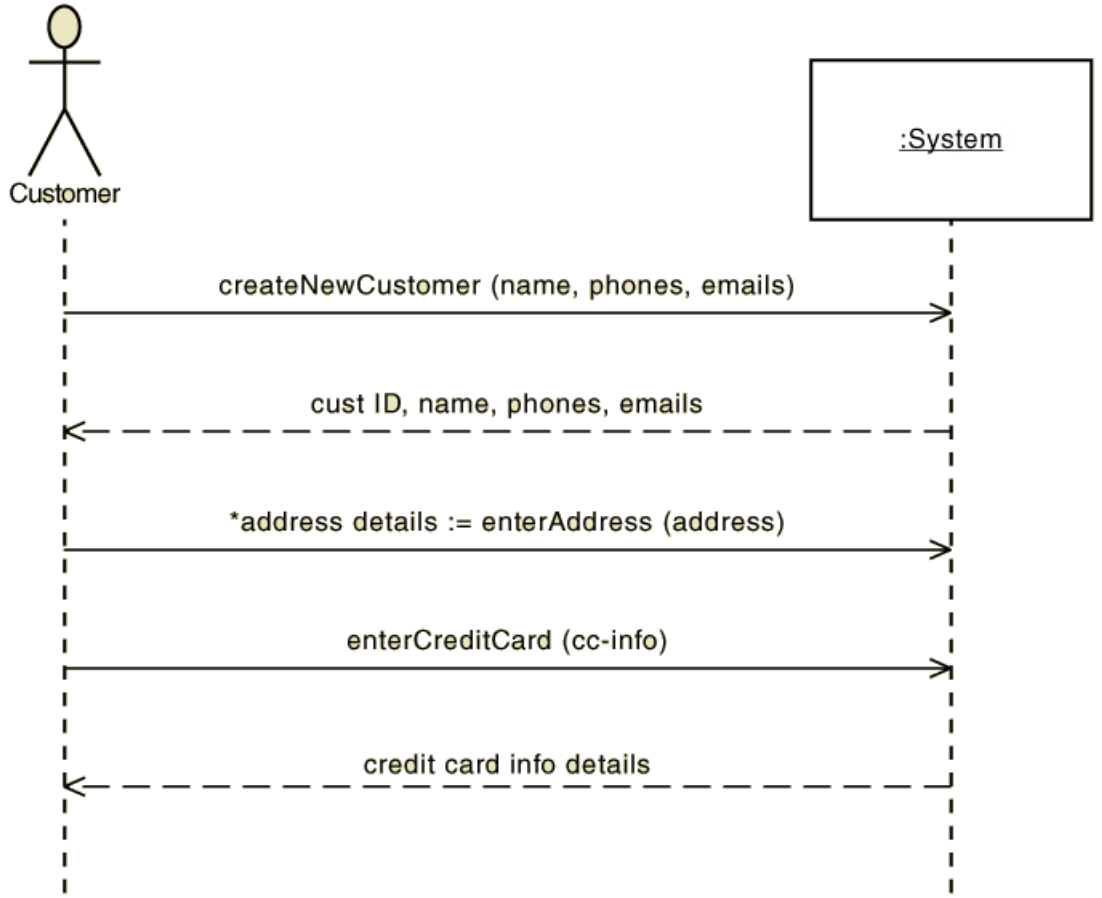
OOD with Communication Diagrams

First-cut DCD for *Create customer account* use case



OOD for *Create customer account*

Input model -- SSD



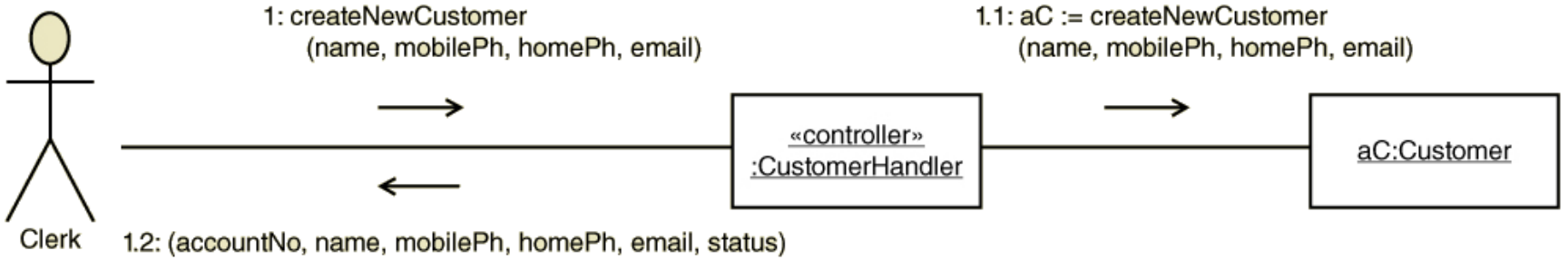
OOD for *Create customer account*

Extend input messages

1. From the DCD put objects on communication diagram
2. For each message, find primary object, ensure visibility, elaborate use case with all messages between objects
3. Name each message and add all required message elements

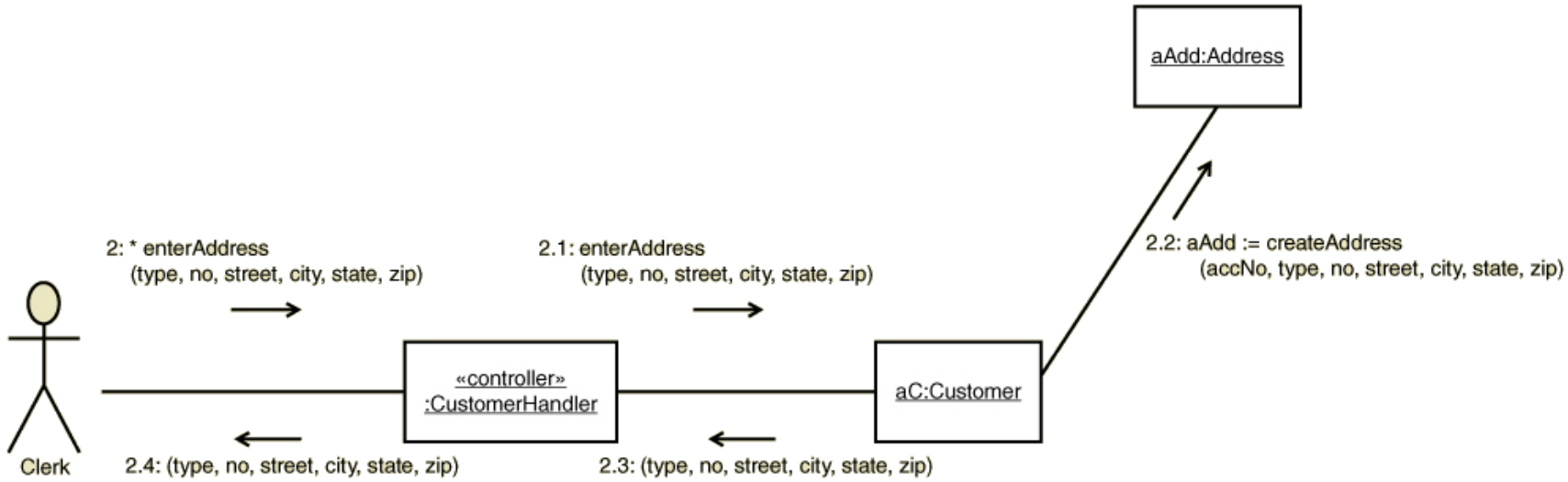
OOD with communication diagrams

createNewCustomer message extended to all objects



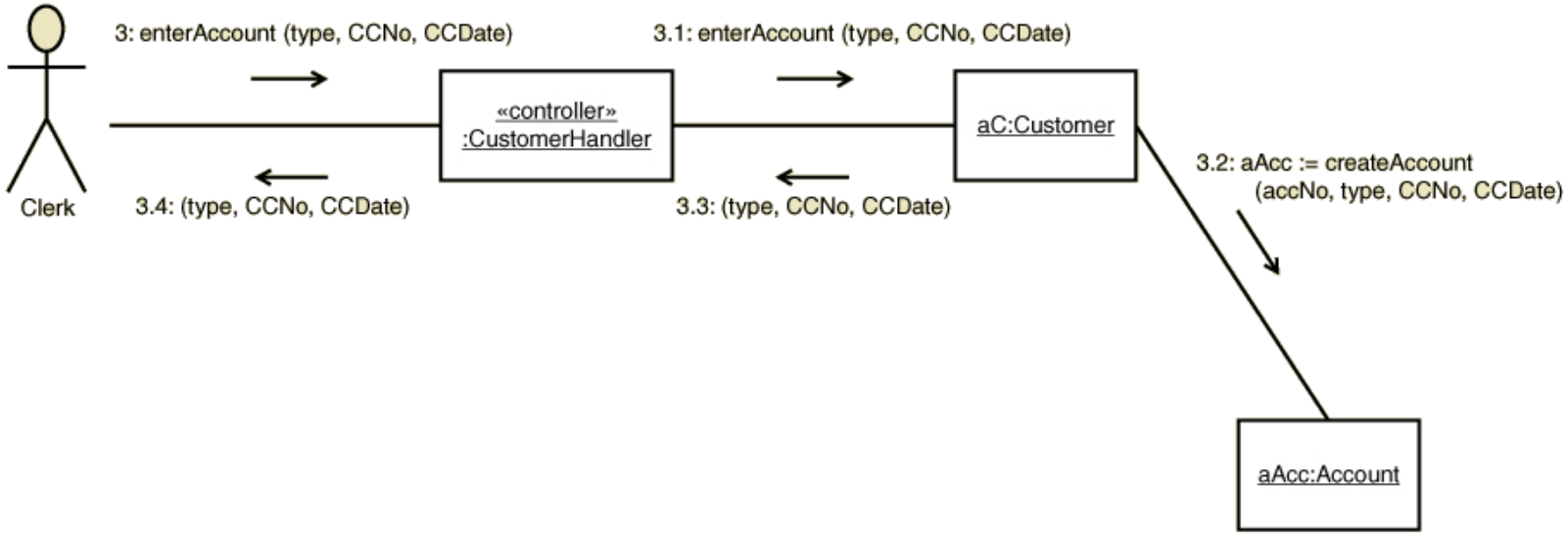
OOD with communication diagrams

enterAddress message extended to all objects



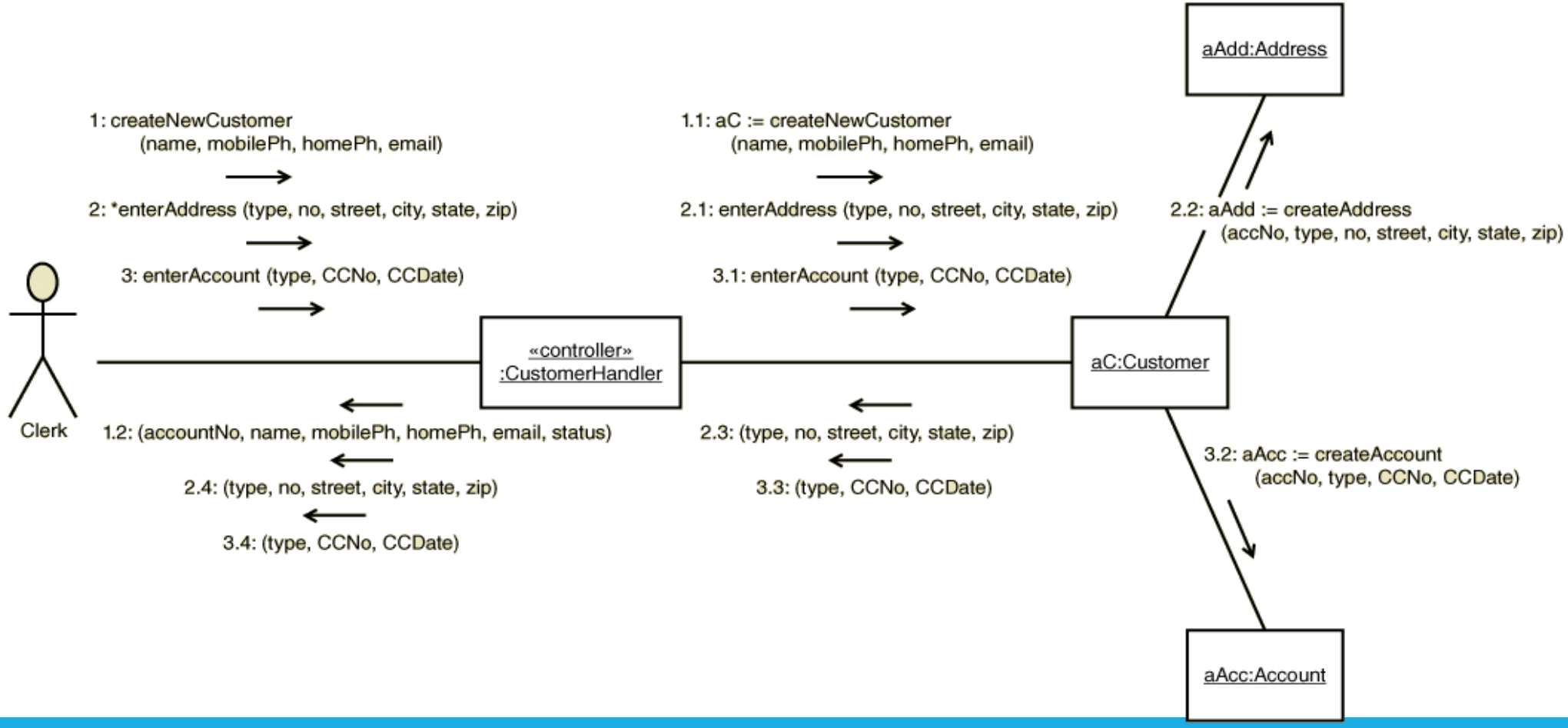
OOD with communication diagrams

enterAccount message extended to all objects



OOD with communication diagrams

Final communication diagram with all messages



OOD with communication diagrams

Updated DCD