CHAPTER 8
Object Oriented Analysis and Design

Management of object-orient software projects

- Greater resilience to change :
- Increased degree of re-use:
- Complex system is divided into different subsystem:
- Improved Quality :

Q.N Explain different features of OOP. Why do you prefer object orient software projects.

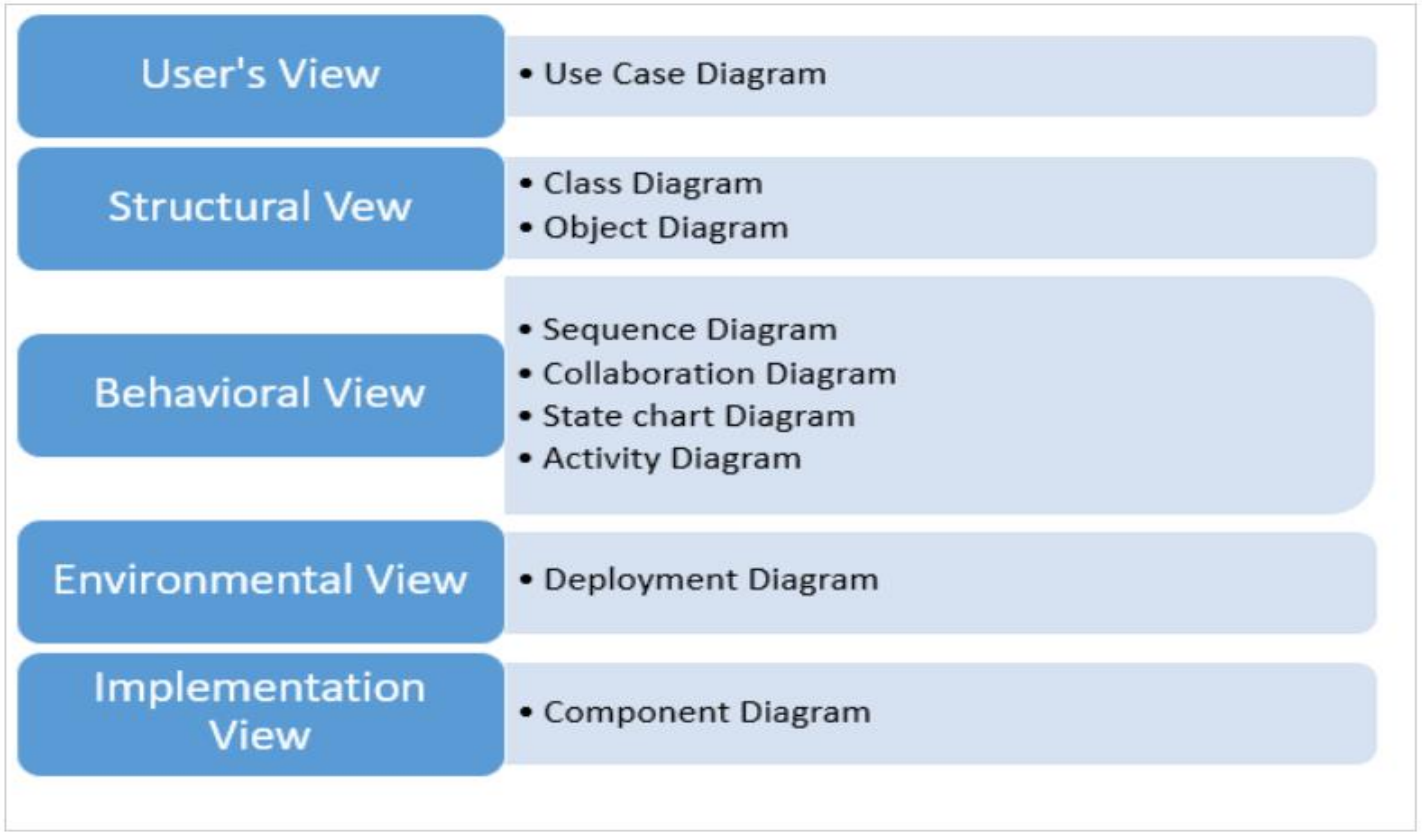
Object-Oriented Analysis

- **The Booch method**: macro & micro development
- **The Rumbagh method**: object modelling technique
- **The Jacobson method**: Object Oriented Software Engineering (Use Case)

Unified Approach to OOA

- Over the past decade, Booch, Jacobson & **Rumbagh** have collaborated to combine the best features of their individual object-oriented analysis and design into a unified method.
- UML allows a software engineers to express an analysis model using a modeling notation that is governed by a set of syntactic, semantic rules.

UML View


User's View
• Use Case Diagram
Structural Vew
• Class Diagram
• Object Diagram
Behavioral View
• Sequence Diagram
• Collaboration Diagram
• State chart Diagram
• Activity Diagram
Environmental View
• Deployment Diagram
Implementation View
• Component Diagram

Domain Analysis

- Analysis for object-oriented systems can occur at many different **levels of abstraction**.
- Domain Analysis falls under the **middle level** of abstraction.
- Domain analysis is performed when an organization wants to create **library of reusable classes**.
- The objective of domain analysis is to define a set of classes which can be reused in many applications

Q.N Why do we need domain analysis. Explain domain analysis process.

Domain Analysis Process

- The goal of domain analysis is to find or create **classes** that are broadly applicable so that they may be **reused.**

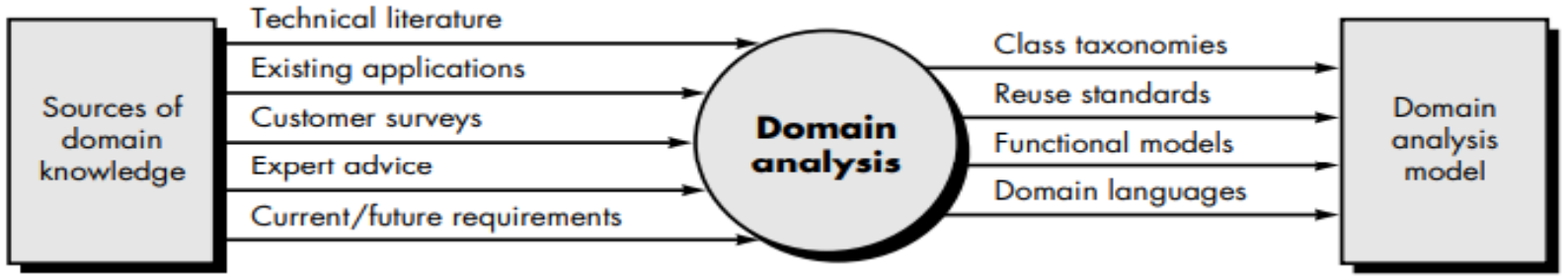


Figure: Input and output process for domain analysis

Domain Analysis Process

- The domain analysis process can be characterized by a series of activities:
    - **Define the domain to be investigated**
    - **Categorize the items extracted from the domain.**
    - **Collect a representative sample of applications in the domain**
    - **Analyze each application in the sample.**
    - **Develop an analysis model for the objects.**

GENERIC COMPONENTS OF THE OO ANALYSIS MODEL

- **Static components** do not change as the application is executed.
- **Dynamic components** are influenced by timing and events
- *Static components are structural in nature*
- *Dynamic components focus on control and are sensitive to timing and event processing*

Key components of OOA model

- **Static view of classes**
- **Static view of attributes**
- **Static view of relationships**
- **Static View of behaviors**
- **Dynamic view of communication**
- **Dynamic view of control and time**

THE OOA PROCESS

- The OOA process does not begin with a concern for **objects**. Rather, it begins with an understanding of the manner in which the **system** will be used
- series of techniques that may be used to gather basic customer requirements and then define an analysis model for an object oriented system are given below
- **Use Cases**
- **Class Responsibility Collaborator Modelling**
- **Defining Structures and Hierarchies**
- **Defining Subjects and Subsystems**

Q.N Explain OOA process

Use-case

- Use-case model the system from the end-user's point of view.
- Created during requirements elicitation,
- use-cases should achieve the following objectives:

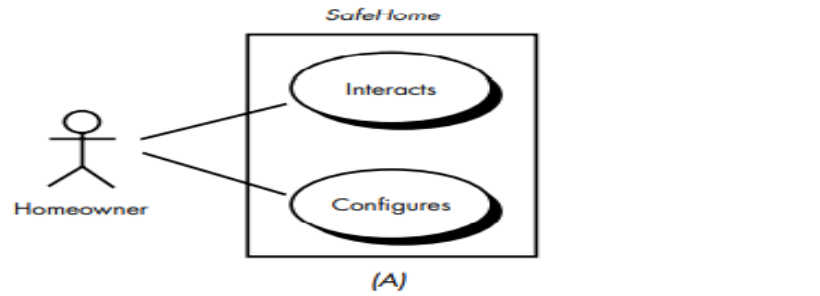
• To define the functional and operational requirements of the system that is agreed upon by the end-user and the software engineering team.

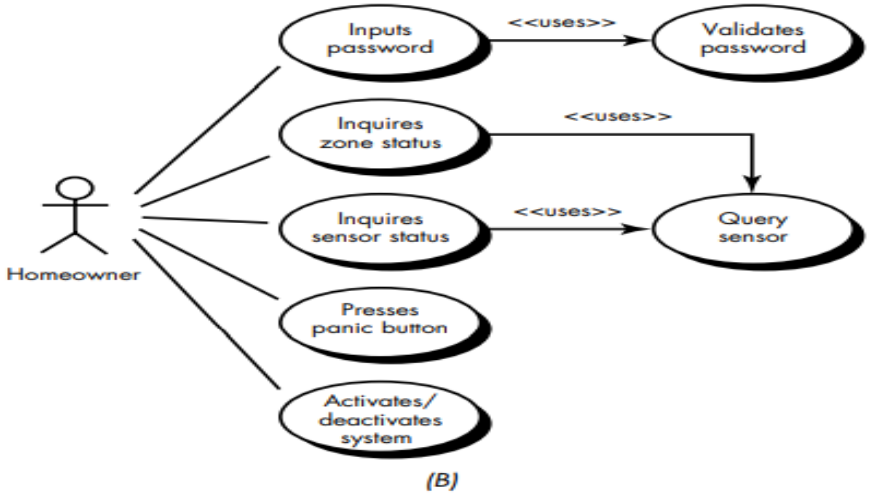
• To provide a clear and unambiguous description of how the end-user and the system interact with one another.

• To provide a basis for validation testing

Q.N Explain need of use case with an example. Draw use case diagram for library management system.

Use-Cases


SafeHome
Interacts
Configures
Homeowner
(A)


Inputs password
<<uses>>
Validates password
Inquires zone status
<<uses>>
Inquires sensor status
<<uses>>
Query sensor
Presses panic button
Activates/ deactivates system
Homeowner
(B)

Class-Responsibility-Collaborator Modeling

- Once basic usage scenarios have been developed for the system, it is time to identify candidate classes and indicate their responsibilities and collaborations.
- Class responsibility-collaborator (CRC) modeling provides a simple means for identifying and organizing the classes which are relevant to system or product requirements.

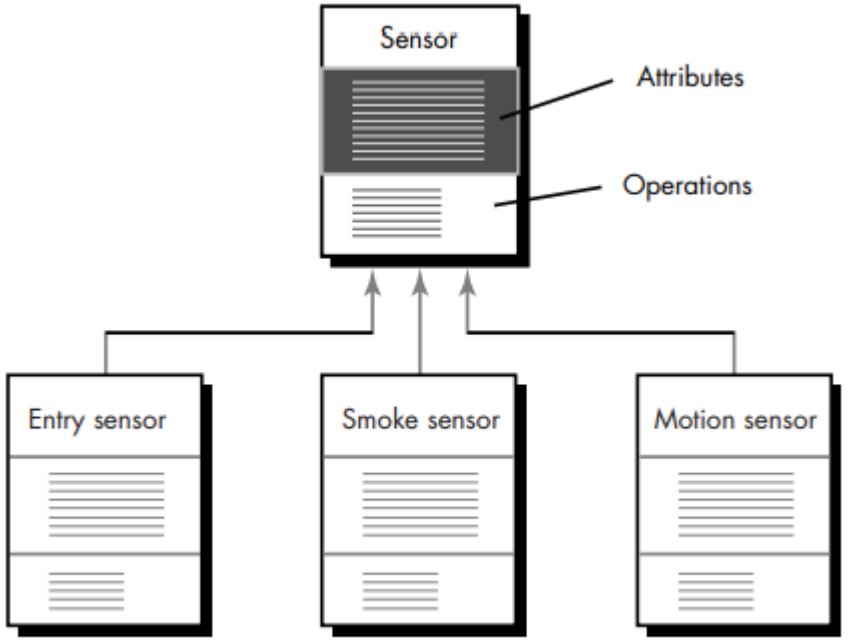
Q.N Explain the need of CRC card with an example. Draw CRC card for Library system

Class-Responsibility-Collaborator Modeling

Class name:	
Class type: (e.g., device, property, role, event)	
Class characteristic: (e.g., tangible, atomic, concurrent)	
responsibilities:	collaborations:

Defining Structure and Hierarchies

Q.n How will you define structure of class and hierarchies in oo system



Class diagram for generalization

Defining Structure and Hierarchies

Q.n Draw a class diagram for Library.

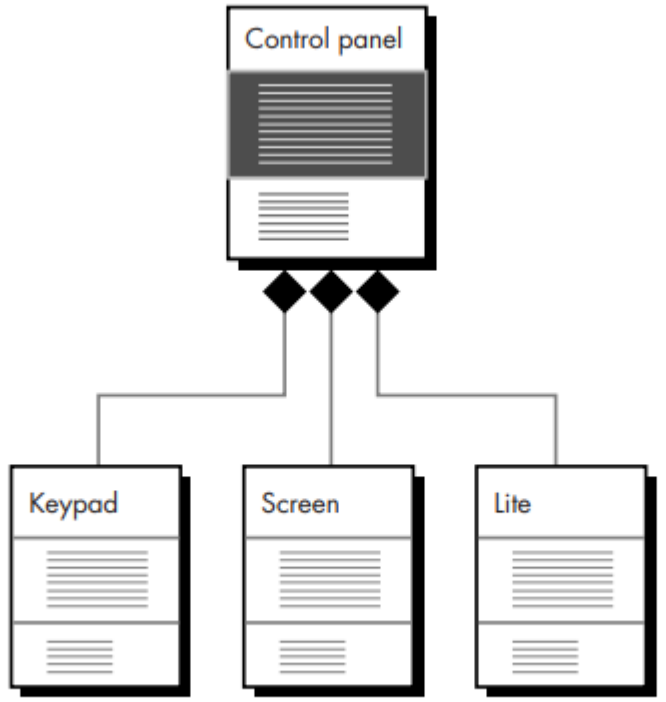
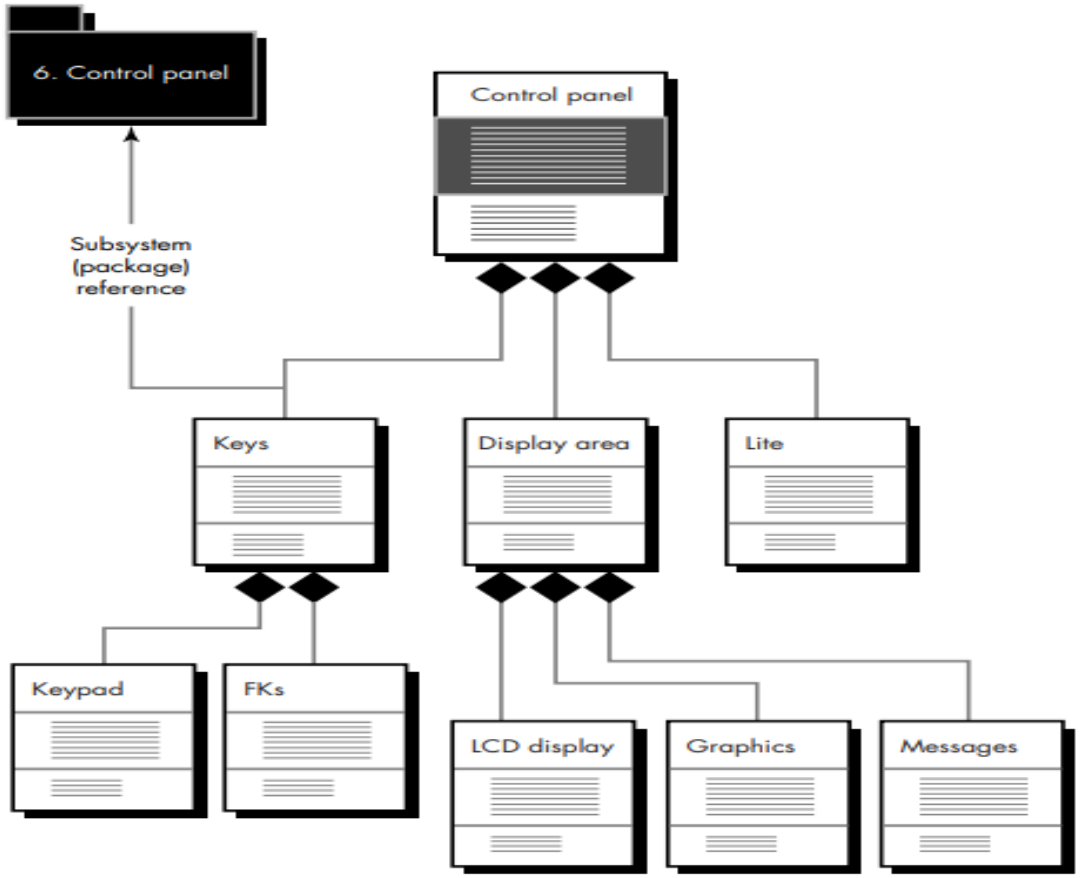


FIGURE 21.5
Class diagram for composite aggregates

Defining Subjects and Subsystems

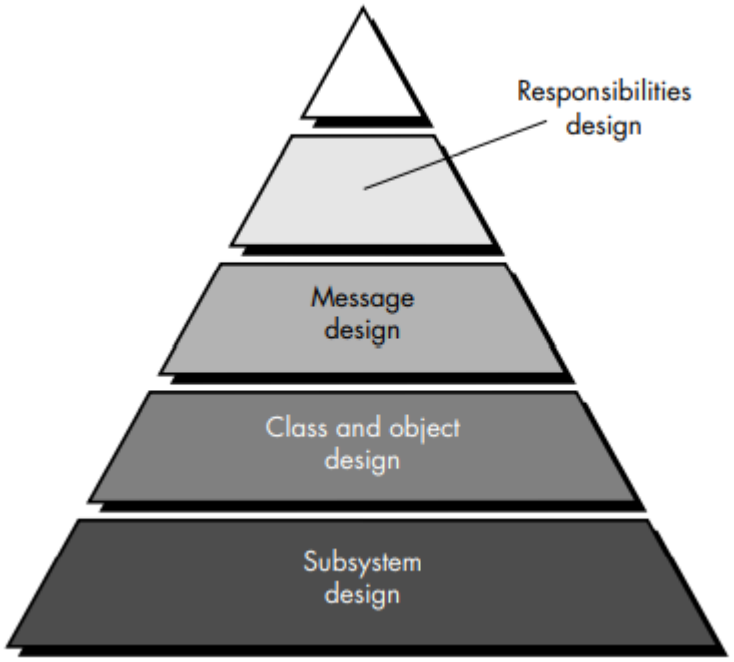


OBJECT ORIENTED DESIGN

- During the object oriented design there is an emphasis on defining **software objects** and **how** they collaborate to fulfill the requirements.
- Object-oriented design transform the analysis model created using **object-oriented analysis** into a design model that serves as a **blueprint** for software construction.

***Q.N Explain OOD process* with its layers.**

Layers of OO design Pyramid



Layers of OO design Pyramid

- **The subsystem layer:** contains the representation of subsystems helps to achieve its customer-defined requirements
- **The class and object layer** contains the class hierarchies. This layer also contains representations of each object
- **The message layer** contains the design details that enable each object to communicate with its collaborators. This layer establishes the external and internal interfaces for the system

Layers of OO design Pyramid

- **The responsibilities layer** contains the data structure and algorithmic design for all attributes and operations for each object

Translating OOA model to OOD model

Q.N How can you translate OOA to OOD.

Q.N Differentiate between OOA and OOD process.

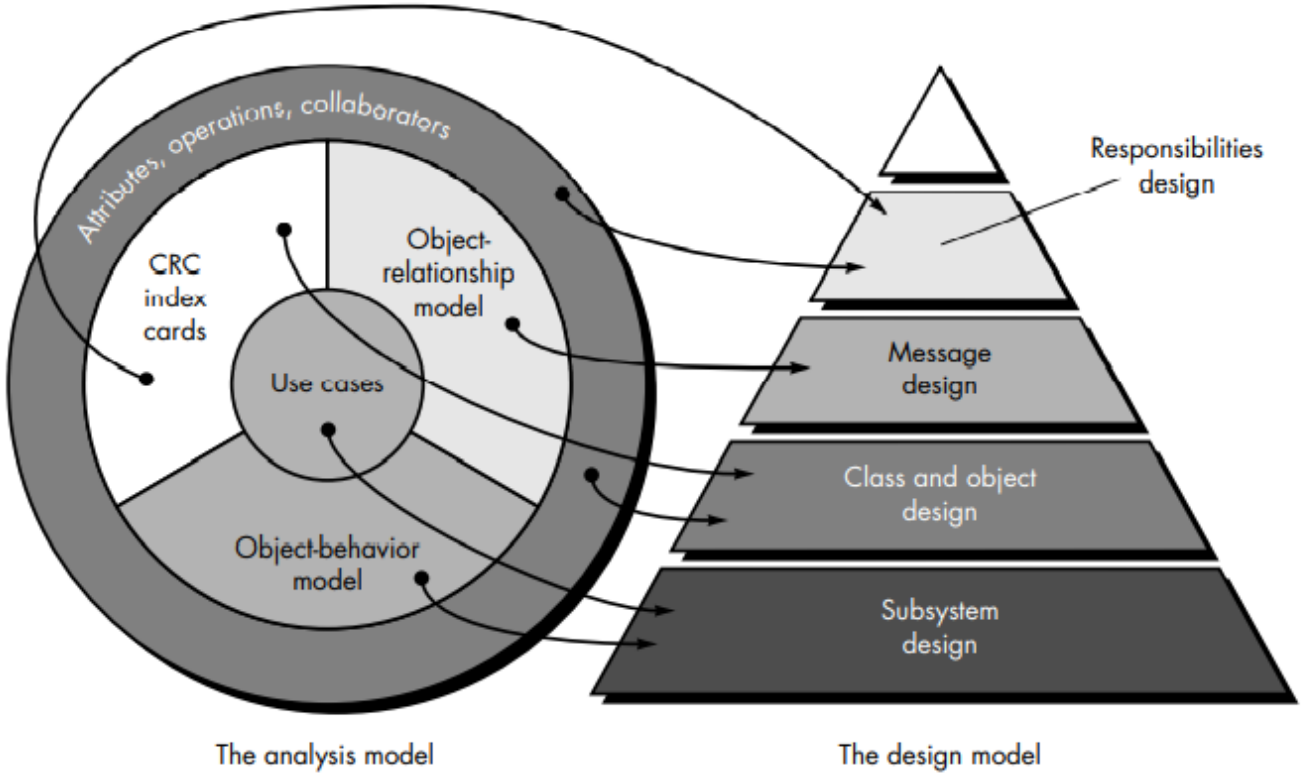


FIGURE 22.2 Translating an OOA model into an OOD model

Translating OOA model to OOD model

- **The subsystem design** is derived by considering overall customer requirements (represented with use-cases) and the events and states that are externally observable (the object-behavior model).
- **Class and object design** is mapped from the description of attributes, operations, and collaborations contained in the CRC model.
- **Message design** is driven by the object-relationship model,
- **Responsibilities design** is derived using the attributes, operations, and collaborations described in the CRC model

DESIGN PATTERNS

Q.N Explain design pattern with an example.

- Design patterns allow the designer to create the system architecture by integrating **reusable** components.
- There are many recurring patterns of **classes** and **communicating objects** in many object oriented systems.
- Throughout the OOD process, a software engineer should look for every opportunity to **reuse existing** design patterns rather than creating new ones.

Describing a Design Pattern

- All design patterns can be described by specifying the following information
- the **name** of the pattern
- the **intent** of the pattern
- the "**design forces**" that motivate the pattern
- the **solution** that mitigates these forces
- the **classes** that are required to implement the solution
- the **responsibilities** and **collaboration** among solution classes
- **guidance** that leads to effective implementation
- **example** source code or source code templates
- **cross-references** to related design pattern

Using Patterns in Design

- In an object-oriented system, design patterns can be used by applying two different mechanisms**:**

a. Inheritance

b. composition

Using Patterns in Design

Composition

- Composition is a concept that leads to **aggregate objects.**
- That is, a problem may require objects that have **complex functionality**

Using Patterns in Design

Inheritance

Using **inheritance**, an existing design pattern becomes a template for a new **subclass**.

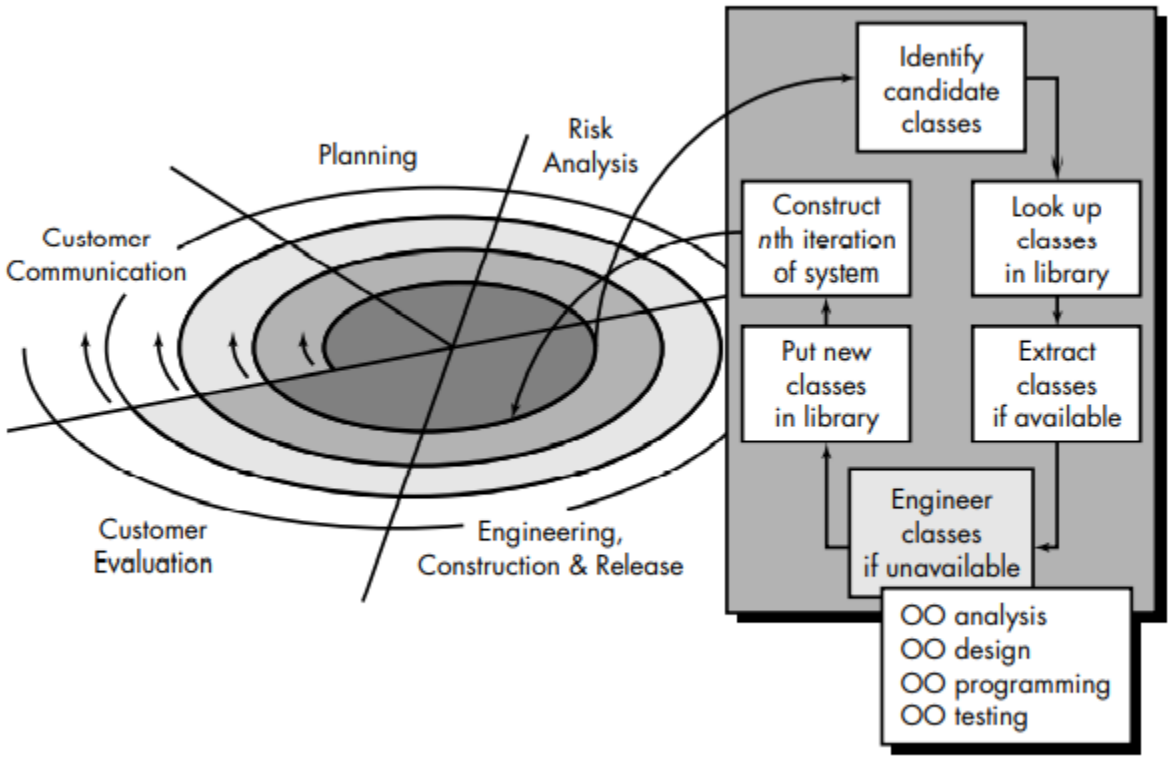
The **attributes and operations** that exist in the pattern become part of the **subclass**

CHAPTER 8

OBJECT-ORIENTED CONCEPTS AND PRINCIPLES

The OO Process model

FIGURE 20.1
The OO process model



OO Process model

- The OO process moves through an **evolutionary spiral** that starts with customer communication.
- .Planning and risk analysis establish a foundation for the OO project plan
- The technical work associated with OO software engineering follows the **iterative path** shown in the shaded box.
- OO software engineering emphasizes reuse.

OO Process model

- Therefore, classes are “looked up” in a library (of existing OO classes) before they are built.
- When a class cannot be found in the library, the software engineer applies object-oriented analysis (OOA), object-oriented design (OOD), object-oriented programming (OOP), and object-oriented testing (OOT) to create the class and the objects derived from the class.
- The new class is then put into the library so that it may be reused in the future

Definition of Object and Class

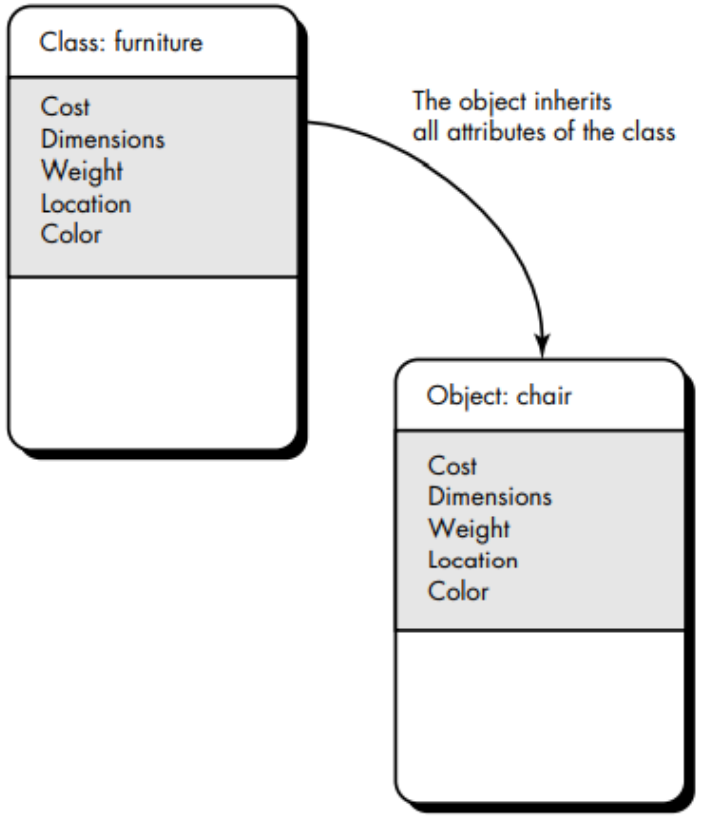
- **Class** : A class is a generalized description that describes a collection of a similar objects.
- **Object** : Instance of Class is called object

- One widely used approach to problem solving takes an **object-oriented viewpoint**

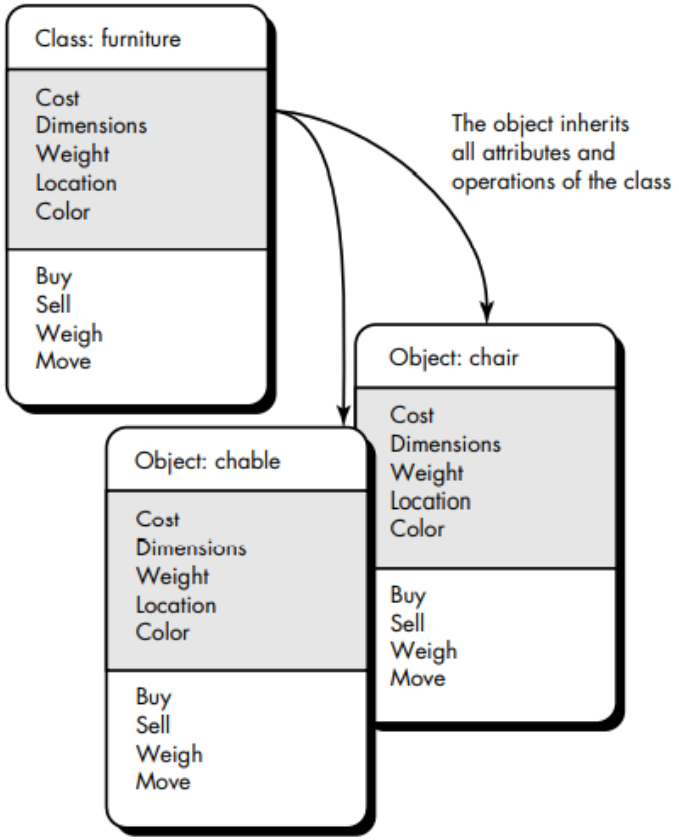
The Object Oriented Paradigm

- For many years, the term object oriented (OO) was used to denote a **software development approach** that used one of a number of object-oriented programming languages Java, C++,
- Today, the OO paradigm encompasses a complete view of software engineering.

Inheritance

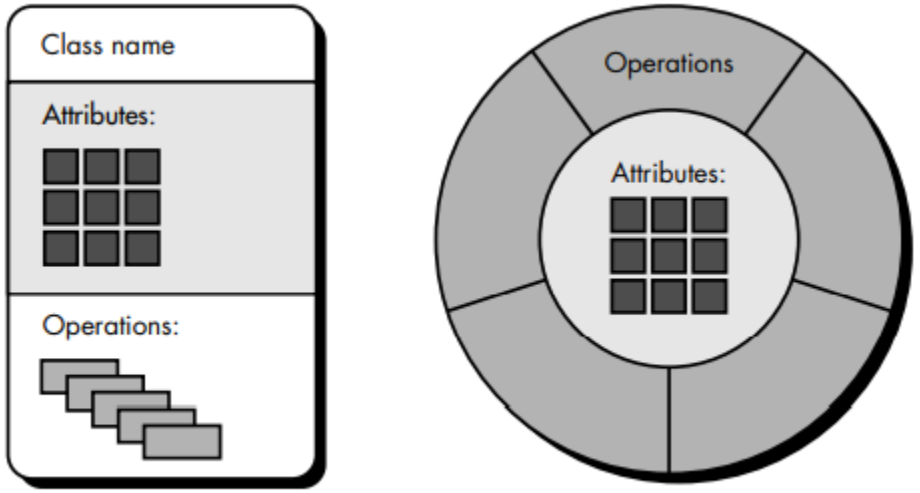


Inheritance

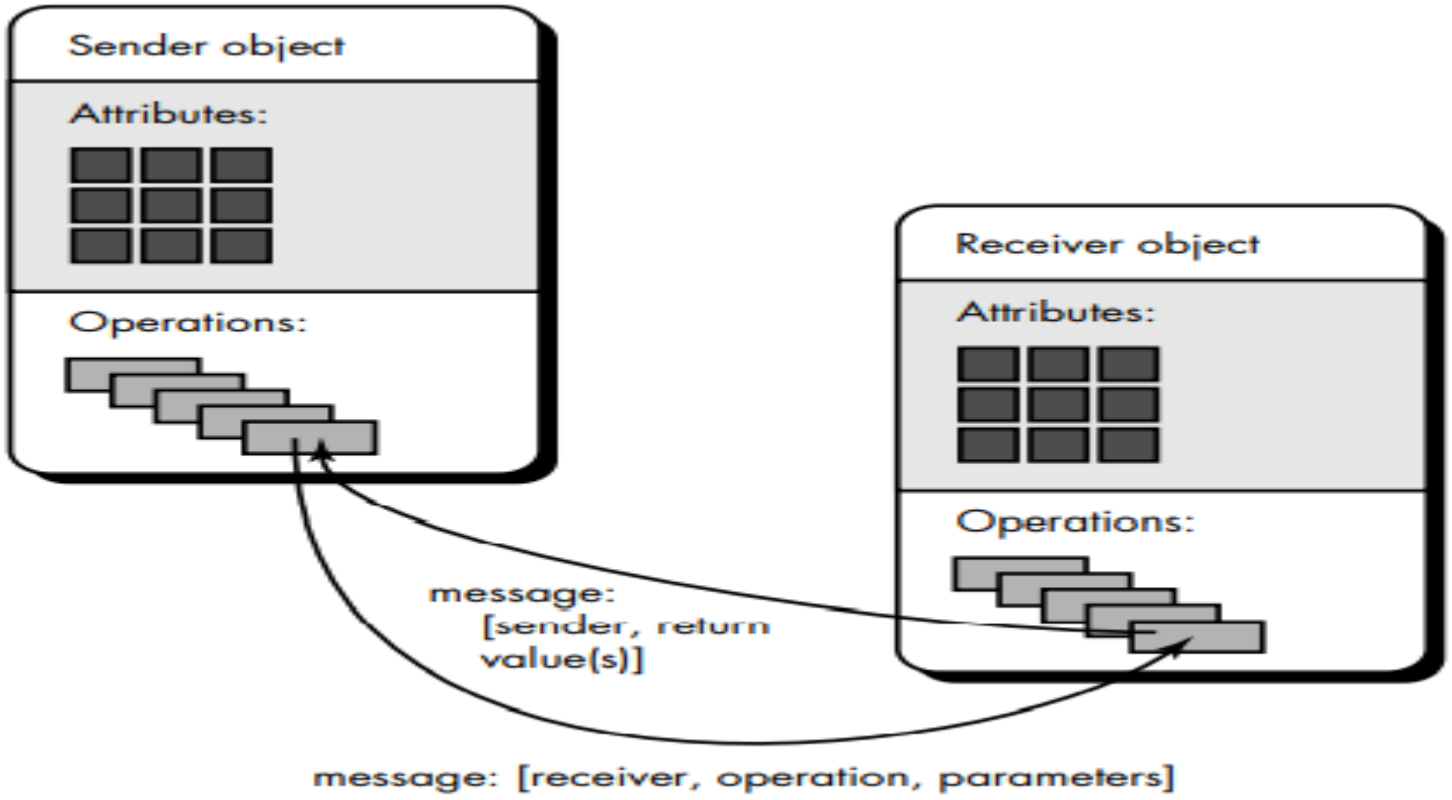


Class and Object

An Alternative way to represent class and object



Message Passing Between Class



IDENTIFYING THE ELEMENTS OF AN OBJECT MODEL

1. **Identifying Classes and Objects**

Objects can be:-

- **External Entities**
- **Things**
- **Occurrences or events**
- **Roles**
- **Organizational units**
- **Places**
- **Structure**

IDENTIFYING THE ELEMENTS OF AN OBJECT MODE

2. Specifying Attributes

- Attributes are chosen by examining the problem statement, looking for things that fully define an object and make it unique.

IDENTIFYING THE ELEMENTS OF AN OBJECT MODE

3. **Defining the operation**

Although many different types of operations exist, they can generally be divided into three broad categories:

(1) operations that manipulate data in some way (e.g., adding, deleting, reformatting, selecting),

(2) operations that perform a computation,

(3) operations that monitor an object for the occurrence of a controlling event

IDENTIFYING THE ELEMENTS OF AN OBJECT MODE

3. Finalizing the Object Definition

The definition of operations is the last step in completing the specification of an object

OO Project Metrics and Estimation

- Conventional software project estimation -- (LOC) and function points (FP)

OOP Project Metrics is based on:

1. **Number of scenario scripts** (initiator, action and participator)
2. **Number of key classes**
3. **Number of support classes**
4. **Average number of support classes per key class**
5. **Number of subsystems.**