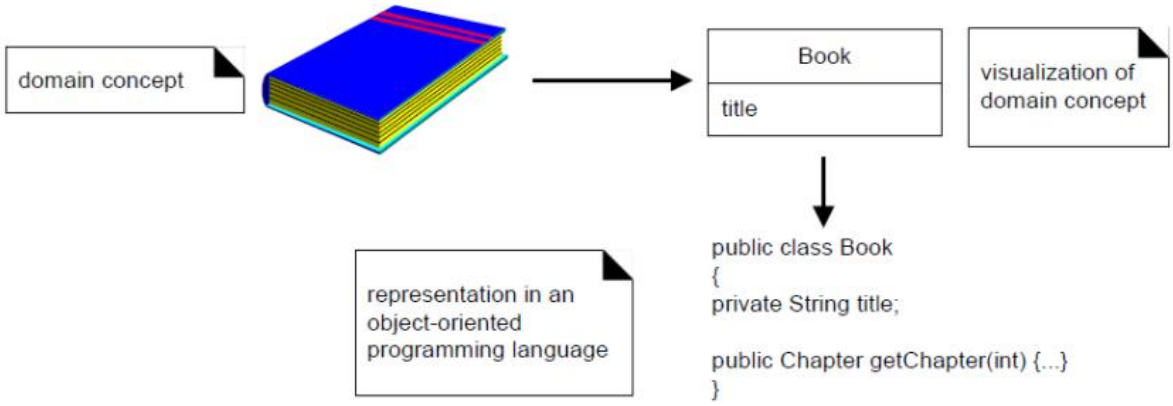
Chapter 5: Object Oriented Design

- Components of 00 Design model
- System Design process
- Partitioning the analysis model
- Concurrency and subsystem allocation
- Task Management component
- Object DBMS
- Data Management components
- Resource Management components
- Inter sub-system communication
- Object Design process

Object Oriented Design

- During object-oriented analysis, the focus is on finding and describing objects or concepts in the problem domain. For example, in the case of library management system, some of the concepts are Book, Library and Librarian.
- During object-oriented design, the focus is on defining software objects and how they collaborate to fulfill the requirements. For example, in the library system, a Book software object may have a title attribute and getChapter method.
- Object Oriented Design serves as part of OOP process of lifestyle.
- It is mainly the process of using an object methodology to design a computing system or application.
- This technique enables the implementation of software based on the concepts of objects.



Characteristics of OOD

- Objects are abstractions of the real-world or system entities and manage themselves.
- The objects are independent and in an encapsulated state and representation information.
- System functionality is expressed in terms of object services.
- Communication between objects is through message passing.
- The objects may be distributed and may execute sequentially or in parallel.

The Booch Method: Object Oriented Design (OOD)

- The Booch method is a software engineering technique.
- The Booch method is a method for object-oriented software development.
- The method was authored by Grady Booch when he was working for Rational Software (acquired by IBM), published in 1992 and revised in 1994.
- It was widely used in software engineering for object-oriented analysis and design and benefited from ample documentation and support tools.
- The notation aspect of the Booch method was superseded by the Unified Modeling Language (UML), which features graphical elements from the Booch method along with elements from the object-modeling technique (OMT) and object-oriented software engineering (OOSE).
- Methodological aspects of the Booch method have been incorporated into several methodologies and processes, the primary such methodology being the Rational Unified Process (RUP).
- Booch uses many diagrams in his OOD at two different levels:
    i. Design level
    ii. Implementation level

S.N	Diagram Type	Level Define
1	Class diagram	Design Level
2	Object Diagram	
3	State transition Diagram	
4	Interaction Diagram	
5	Module Diagram	Implementation Level
6	Process Diagram	

Booch Methodology provides micro and macro development process for the software production.

Micro Development Process:

- The micro process represent the daily activity of the individual developer, or of a small team of developers. Here the analysis and design phases are intentionally blurred.
- Steps in Micro development phase:
    - Identify the classes and objects at a given level of abstraction.
    - Identify the semantics of these classes and objects.
    - Identify the relationships among these classes and objects.
    - Specify the interface and then the implementation of these classes and objects.

Macro Development Process:

- Each macro development process has its own micro development process.
- This process focuses on the two manageable elements: Risk and Architectural vision.
- The macro process identifies the following activities cycle:
    - **Conceptualization:** establish core requirements
    - **Analysis:** develop a model of the desired behavior
    - **Design:** create an architecture

- **Evolution:** for the implementation
- **Maintenance:** for evolution after the delivery

The Coad- Yourdon Method

- This model describes static characteristics.
- It addresses not only the application but also the infrastructure for the application.
- Coad-Yourdon methodology has its primary strength in system analysis.
- Their methodology is based on a technique called "SOSAS", which stands for the five steps that help make up the analysis part of their methodology.
- The first step in system analysis is called "Subjects", which arc basically data flow diagrams for objects.
- The second step is called "Objects", where they identify the object classes and the class hierarchies. The third step is called "Structures", where they decompose structures into two types, classification structures and composition structures.
- Classification structures handle the inheritance connection between related classes, while composition structures handle all of the other connections among classes.
- The next step in analysis is called "Attributes".
- The final step is called "Services", where all of the behaviors or methods for each class are identified.
- focuses on the representation of four major system components:
  1. Problem domain component
  2. Human interaction component
  3. Task management component
  4. Data management component.

The Jacobson Method

- Design model emphasizes traceability to the OOSE analysis model
- Idealized analysis model is adapted to fit the real world environment.
- Primary design objects, called blocks are created and categorized as interface blocks, entity blocks, and control blocks.
- Communication between blocks during execution is defined.
- Blocks are organized into subsystems.

The Rumbaugh method

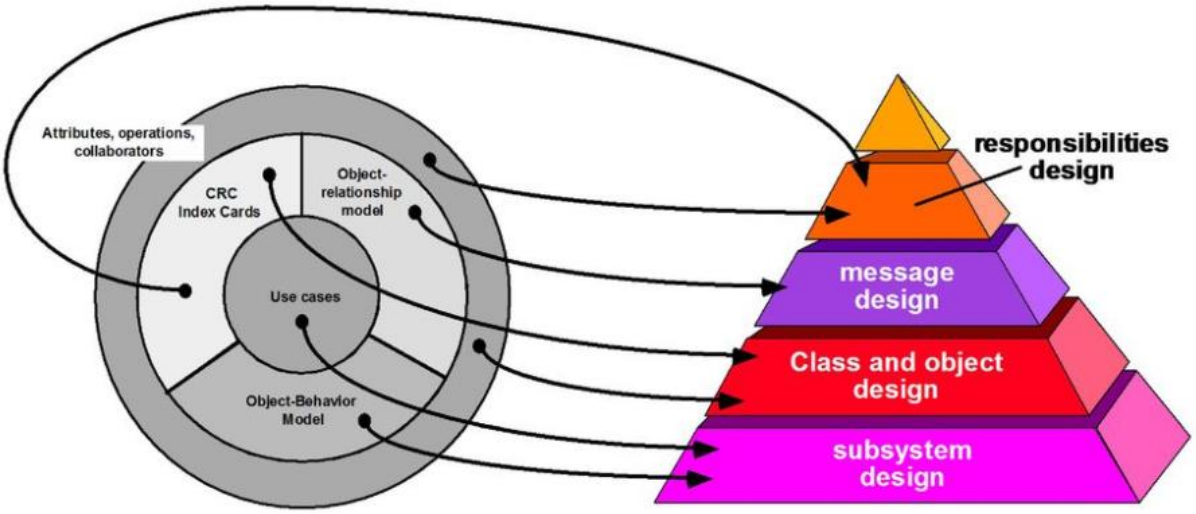
- encompasses a design activity that encourages design to be conducted at two different levels of abstraction:
- **System design:**
  - focuses on the layout of components that are needed to construct a complete product or system
  - Analysis model is partitioned into subsystems -> allocated to processors and tasks.
  - strategy for implementing data management is defined
  - global resources and the control mechanisms required to access them are identified
- **Object design**
  - Emphasizes the detailed layout of an individual object.
  - Operations are selected from the analysis model and algorithms are defined for each operation.

- Data structures appropriate to attributes and algorithms are represented.
- Classes and class attributes are designed so they optimize data access and improve computational efficiency.
- Messaging model is created to implement the object relationships (associations).

To perform object-oriented design, a software engineer should perform the following generic steps:

1. Describe each subsystem and allocate it to processors and tasks.
2. Choose a design strategy for implementing data management, interface support, and task management.
3. Design an appropriate control mechanism for the system.
4. Perform object design by creating a procedural representation for each operation and data structures for class attributes.
5. Perform message design using collaborations between objects and object relationships.
6. Create the messaging model.
7. Review the design model and iterate as required.

Relationship between OOA and OOD



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

1. The Subsystem Layer: This layer contains a representation of each of the subsystems that enable the software to achieve its customer defined requirements and to implement the technical infrastructure that supports customer requirements.
2. The Class and Object Layer: This layer contains a hierarchy of classes, which enable the system to be created using generalizations and increasingly more targeted specializations. This layer also represents each object.
3. The Message Layer: This layer contains the design details, which enables each object to communicate with its collaborators. This laver establishes the internal and external interfaces for the system.
4. The Responsibilities Layer: This layer contains the data structure and algorithmic design for all operations and attributes for each object.

Generic Component for OOD

1. Problem domain component

- It includes the problem domain classes that were identified during OOA.
- The subsystems that are responsible for implementing customer requirements directly.

2. Human interaction component

- It is a very important component of OO design model since using this component, a model interacts with Human.
- The subsystems that implement the user interface (this included reusable GUI subsystems).
- For example: MS Excel has various cells in which we put entries and manipulate them. Each different entry is classified into different manner.

ID	Name	Salary
1	Nilesh	8000
2	Atul	5700
3	Sourabh	7000
4	Sunil	10000
		30700

Fig: Example of database

- In example, we take three columns ID, Name, Salary which are according to the class design. Here all three attributes, operations (we make the sum of third column salary) we use sum method for that.

3. Task Management Component

- The subsystems that are responsible for controlling and coordinating concurrent tasks that may be packaged within a subsystem or among different subsystems.

4. Data management component

- The subsystem that is responsible for the storage and retrieval of objects.

System Design Process:

- System design develops the architectural detail required to build a system or product.
- It is the process of defining elements of a system like modules, architecture, components and their interfaces and data for a system based on the specified requirements. The designs can be defined in graphical or textual modelling language.
- System design encompasses the following activities:
    1. Partition the analysis model into subsystems.
    2. Identify concurrency that is dictated by the problem.
    3. Allocate subsystems to processors and tasks.
    4. Develop a design for the user interface.
    5. Choose a basic strategy for implementing data management.
    6. Identify global resources and the control mechanisms required to access them.
    7. Design an appropriate control mechanism for the system, including task management.
    8. Consider how boundary conditions should be handled.
    9. Review and consider trade-offs.

1. Partitioning the analysis model

- Partition the analysis model to define cohesive collections of classes, relationships, and behavior namely subsystems.
- These design elements are packaged as a subsystem.
- All of the elements of a subsystem share some property in common
- All may be involved in accomplishing the same function
- They may reside within the same product hardware, or they may manage the same class of resources
- Subsystems are characterized by their responsibilities -> identified by the services that they provide.
- A service is a collection of operations that perform a specific function (e.g., managing word-processor files, producing a three-dimensional rendering, translating an analog video signal into a compressed digital image).
- Each subsystem should have a well-defined interface through which all communication with the rest of the system occurs.
- With the exception of a small number of communication classes, the classes within a subsystem should collaborate only with other classes within the subsystem.
- The number of subsystems should be kept small.
- Subsystems can be partitioned internally to help reduce complexity.
- Communication between subsystems is either peer-to-peer or client-server.

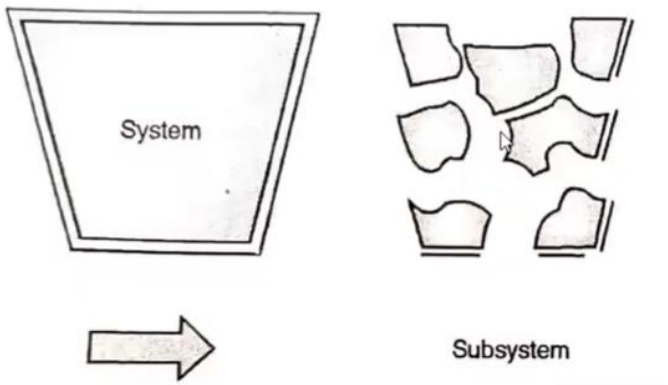


Fig: Partioning the Analysis Model

2. Concurrency and Subsystem Allocation

- Dynamic aspects of object-behavior model provide the indication of concurrency among objects or subsystems.
- If objects or subsystems must act on events asynchronously and at the same time, they are viewed as concurrent.
- When concurrent, there are two options:
  - Allocate to independent processors;
  - Allocate to the same processor and provide concurrency support through OS features.
- To determine which of the processor allocation options is appropriate, the designer must consider performance requirements, costs, and the overhead imposed by interprocessor communication.
- In real world objects are concurrent. For Example: we work with printer, speaker, keyboard as well as many hardware objects work together. This is all about hardware object but this cannot follow with all software objects. It is not always concurrent. The reason is that one process supports many objects. And hence single object can't be utilized at the same time.
- **Identifying inherent Concurrency:** Two objects are inherently concurrent if they react on the events at the same time without interacting. Example: CPU and speaker on a computer must operate concurrently.

3. Task Management Process

- Strategy for the design of the objects that manage concurrent tasks:
  - The characteristics of the task are determined
    - by understanding how the task is initiated
    - Event-driven and clock-driven tasks are most common
    - Both activated by an interrupts
  - A coordinator task and associated objects are defined.
  - The coordinator and other tasks are integrated.
- Priority and criticality of the task must also be determined.
  - High-priority tasks must have immediate access to system resources.
  - High-criticality tasks must continue to operate even if resource availability is reduced or the system is operating in a degraded state.
- Once the characteristics of the task have been determined, object attributes and operations required to achieve coordination and communication with other tasks are defined.

- The basic task template (for a task object) takes the form:
  Task name—the name of the object
  Description—a narrative describing the purpose of the object
  Priority—task priority (e.g., low, medium, high)
  Services—a list of operations that are responsibilities of the object
  Coordinates by—the manner in which object behavior is invoked
  Communicates via—input and output data values relevant to the task

4. Data Management Component

- Data management encompasses two distinct areas of concern:
  - The management of data that are critical to the application itself
  - The creation of an infrastructure for storage and retrieval of objects.
- Data management is designed in a layered fashion
  - Isolate the low-level requirements for manipulating data structures from the higher-level requirements for handling system attributes.
- A Database management system is often used as a common data store for all subsystems
- The design of the data management component includes the design of the attributes and operations required to manage objects.

5. Resource Management Component

- Different resources are available to an OO system or product
- Subsystems compete for these resources at the same time.
- Global system resources can be external entities (e.g., a disk drive, processor, or communication line) or abstractions (e.g., a database, an object).
- Design a control mechanism regardless of the nature of the resource.

6. Intersubsystem Communication

- Interactive Interface means communication protocol between one subsystem with another subsystem.
- Once each subsystem is specified, one needs to define collaborations between subsystems. A collaboration can be viewed as a high-level transaction type or request.
- First identify transaction types and participating subsystems. For each type, identify one or more contracts between pairs of subsystems. A contract provides an indication of the ways in which one subsystem can interact with another.
- If modes of interaction are complex, a high-level event-flow graph should be defined (called a collaboration graph).
- For each contract, identify the following:
  - Type (i.e., client server or peer to peer)
  - Collaborators
  - Components of subsystems supporting services implied by the contract and
  - Message format

Object Design Process:

- Object design is concerned with the detailed design of the objects and their interactions.
- It is completed within the overall architecture defined during system design and according to agreed design guidelines and protocols.
- Object design is particularly concerned with the specification of attribute types, how operations function, and how objects are linked to other objects.
- Purpose of object design:
  • Prepare for the implementation of the analysis model based on system design decisions
  • Transform analysis and system design models
- Investigate alternative ways to implement the analysis model
  • Use design goals: minimize execution time, memory and other measures of cost.
- Object Design serves as the basis of implementation
- A design description of an object (an instance of a class or subclass) can take one of two forms:
  1. **protocol description**:
     - It establishes the interface of an object by defining each message that the object can receive and the related operation that the object performs when it receives the message.
     - It is a set of messages and a corresponding comment for each message.
  2. **Implementation description:**
     - It shows implementation details for each operation implied by a message that is passed to an object.
     - Implementation details include information about the object's private part; that is, internal details about the data structures that describe the object's attributes and procedural details that describe operations.
     - Implementation description of an object provides the internal ("hidden") details that are required for implementation but are not necessary for invocation.
     - Composed of the following information:
       (1) Specification of the object's name and reference to class
       (2) Specification of private data structure with indication of data items and types
       (3) Procedural description of each operation or pointers to such procedural descriptions.
  - Must contain sufficient information to provide for proper handling of all messages described in the protocol description.

Object Design Process

It includes following steps:

1. Develop a problem statement
2. Identify objects and classes
3. Filter out required classes
4. Identify Associations, attributes and links
5. Prepare data dictionary
6. Discard unnecessary associations and attributes
7. Apply inheritance if possible

8. Check the accessibility for possible queries
9. Iterate and Refine model
10. Group Classes into models