CHAPTER 10

OBJECT ORIENTED ANALYSIS AND DESIGN

❖ DOMAIN ANALYSIS

Analysis for OO system can occur at many levels of abstraction. OOA is a middle level of abstraction called as domain domain analysis. Domain analysis is performed when an organization create a library of reusable classes that will be applicable for applications.

Main objectives of domain analysis are to define a set of classes that are encountered throughout an application domain. These can be reused in many applications.

1. Reuse and domain analysis:
   OO technique is reusable technique. Benefit of reuse means that they have consistency and familiarity.
2. Domain analysis process:
   Software domain analysis is the identification, analysis and specification of common requirements from a specific application domain.

OO domain analysis is the identification, analysis and specification of common, reusable, capabilities within a specific application domain, in terms of common objects, classes.

The goal of domain analysis is to find or create classes that are more used so that they may be reused.

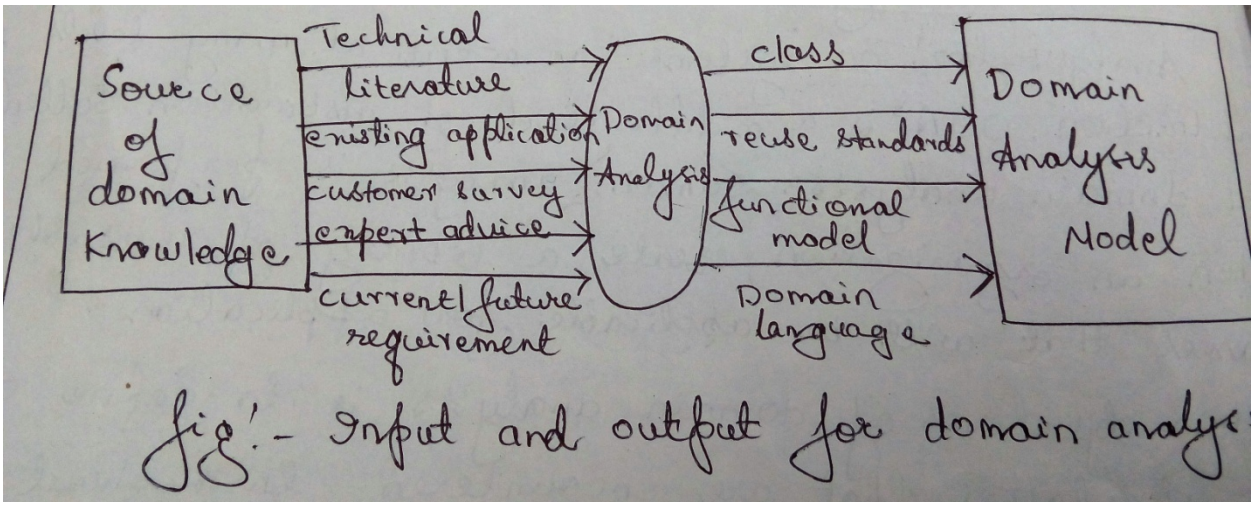

fig:- Input and output for domain analysis

Domain analysis may be viewed as an umbrella activity for the s/w process that means domain analysis is an s/w engineering activity that is not connected to any one s/w project. The role of domain analysis is to design and build reusable components that may be used by many people working on similar but not necessary the same application.

The domain analysis process can be characterized by a series of activity:

a. Define the domain to investigated:
   The analyst first defines the business area, system type or product category of interest. Next both OO and non OO items must be extracted. OO items include specifications, designs and code for existing OO application classes. Non OO items include policies, procedures, plan, standards and guidelines.
b. Categories the items extracted from the domain. The items are organized into categories and the general defining characteristics of the category are defined.
c. Collect a sample of applications in the domain. The domain analyst must identify the conceptual objects in each application.
d. Analyze each application in the sample. The following steps are as followed:
   1. Identify reusable objects.
   2. Indicate the reasons that the object has been identified for reuse.
   3. Estimate the percent of applications in the domain that might make reuse of the object.
   4. Identify the objects by name and use configuration management technique to control them.
   5. Develop an analyst model for the objects. The analysis model will serve as the basic for design and construction of the domain objects.

❖ COMPONENTS OF THE OO ANALYSIS MODEL:

Analysis model is concerned with designing a precise, concise, understandable and correct model of the real world.
OO Analysis is the first technical activity that is performed as part of OO s/w engineering and is based upon the basic concept of objects and attributes and classes.
Generally with real world, object oriented analysis components may be either static or dynamic.
Static components are structural in nature and indicate characteristics that hold throughout the operational life of an application. Static components are:

a. Static view of semantic classes
b. Static view of attribute
c. Static view of relationship
d. Static view of behaviors

Dynamic components focus on control and are sensitive to timing and event processing. These help the objects to interact with each other. For e.g. Dynamic view of communication or message passing and dynamic view of control and time.

❖ THE OOA PROCESS

OOA process doesn't begin with a concern for objects. Rather, it begins with an understanding of manner in which the system is used by user. It uses use case model that represents the system in the above mentioned way. Use case diagram is a diagrammatic UML notation that represents the use case scenario of the system. Use case diagram contains actors that interact with the system and use cases that describe how interactions are done. Once use case is developed, classes and objects are identified.
Class responsibility collaborator (CRC) modeling provides a simple means for identifying and organizing the classes that are relevant to system or product requirements. CRC models are a collection of index cards that represents class. CRC cards have three sections:

- On the top of card we write the name of the identified class.
- In the body of the card, class responsibilities are listed on the left side.
- In the right side the collaborator classes associated with the candidate class are listed.

Once the classes are identified and CRC cards are developed, the structural hierarchies of these classes are created. These are often called generalization/ specialized class structures. The system wholly can be seen as a subsystem of structures of class. Inversely, structural hierarchies of classes can be grouped together in order to form subsystems.

❖ OO DESIGN

OOD transforms the analysis model using OOA into a design model. OOD is divided into two major activities:

1. System design: Consider the three components, the user interface, data management functions and task management.
2. Object design focuses on the internal details of classes, defining attributes, operations and message details.

Design for OO System

For OO systems, we can define a design pyramid, the four layers of OOD pyramids are:

a. The subsystem layer representations of customer defined requirements and to implement the technical infrastructure.
b. The class and object layers contain the class hierarchies and representations of each objects.
c. The message layer contains the external and internal design details for the system
d. The responsibility layer contains the data structure and algorithm design for all attributes and operation for each object.

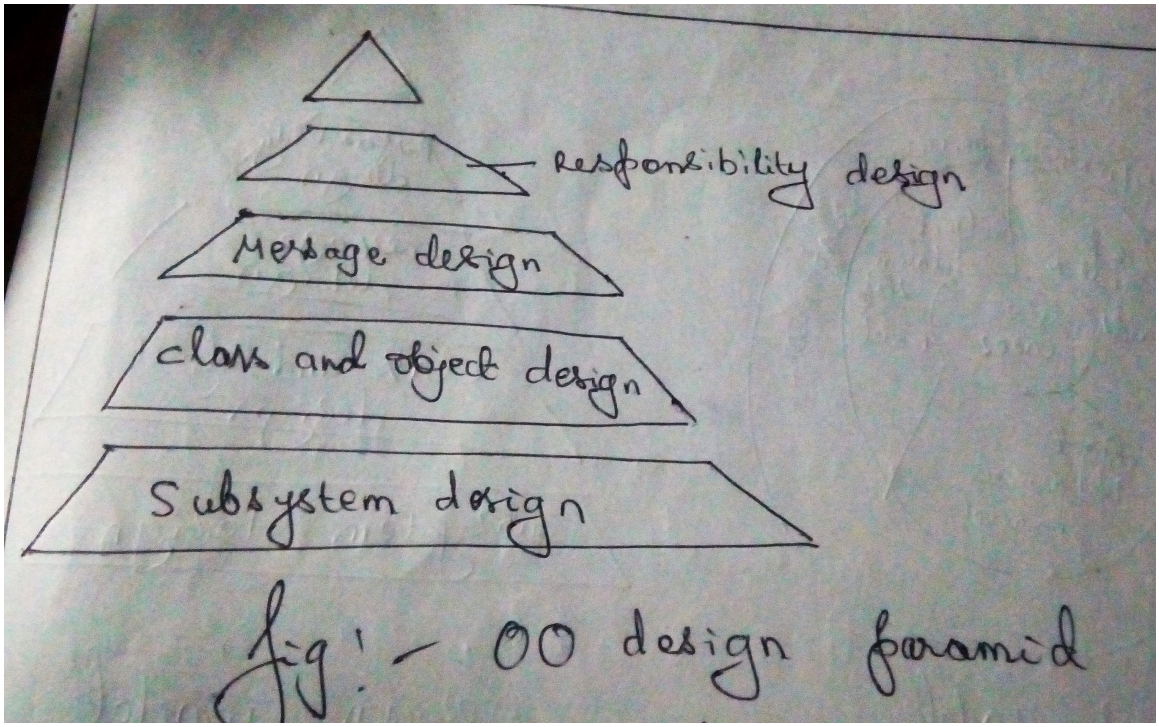

fig:- OO design paramid

The design pyramid focuses on the design of a specific product or systems.

Translating an OOA model into an OOD model figure below illustrate the relationship between the OOA model and OOD model. The subsystem design is derived by considering overall customer requirements and the object behavior model. Class and object design is derived from the description of attributes, operations and collaborations contained in the CRC model. Message design is driven by the object relationship model and responsibilities design is derived using the attributes, operation and collaborations described in the CRC model.

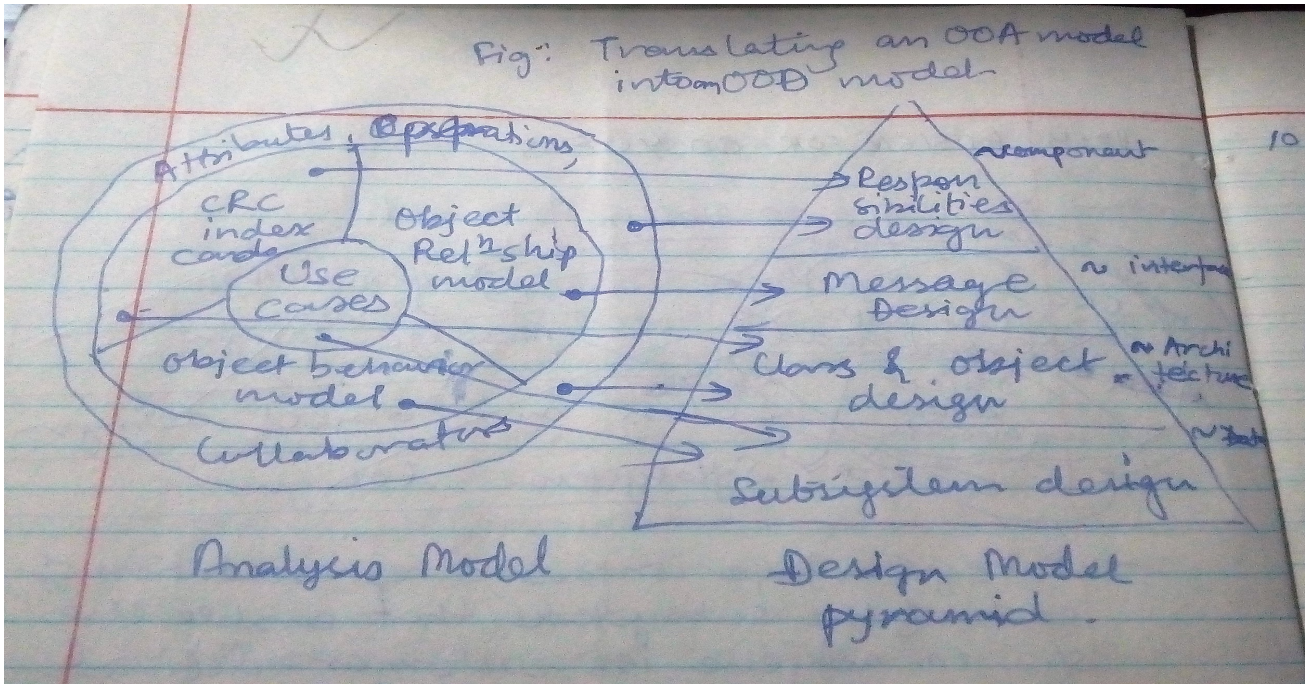

Fig: Translating an OOA model into an OOD model

pyramid.

The design process: It defines the detail of object and their interactions.

1. Object description
2. Designing Algorithm and Data structures
3. Program components and interfaces.

An important aspect of s/w design quality is modularity that is the specification of program components that are combined to form a complete program. The OO approach defines the objects as a program component that is itself links to other components. During design, we must also identify the interface between objects and all the overall structure.

DESIGN PATTERNS:

Design patterns helps designer to create the system architecture of components. There are many patterns of classes and communicating objects in many object- oriented systems. These patterns solve specific design problems and make objects oriented design more flexible and reusable. The OOD process a s/w designer should follow or reuse existing design patterns rather than creating new ones.

All design patterns can be described by specifying the following information

- Name
- Intent
- "design forces" that motivate pattern
- The solution that mitigates these forces
- The classes required to implement the solution
- The responsibilities and collaboration among solution classes
- Guidance that leads to effective implementations.
- Cross- references to related design patterns

Generally, design patterns are implemented using two mechanisms:

- Inheritance: use of existing design pattern as templates for new subclasses, that use attributes and operations from the existing patterns.
- Compositions: It leads to aggregate objects. The formation of complex objects is done by assembling sets of design patterns.