OOAD (OBJECT ORIENTED ANALYSIS & DESIGN)

Unit-1

❖ A Brief History

The object-oriented paradigm took its shape from the initial concept of a new programming approach, while the interest in design and analysis methods came much later.

- The first object–oriented language was Simula (Simulation of real systems) that was developed in 1960 by researchers at the Norwegian Computing Center.
- In 1970, Alan Kay and his research group at Xerox PARK created a personal computer named Dynabook and the first pure object-oriented programming language (OOPL) - Smalltalk, for programming the Dynabook.
- In the 1980s, Grady Booch published a paper titled Object Oriented Design that mainly presented a design for the programming language, Ada. In the ensuing editions, he extended his ideas to a complete object–oriented design method.
- In the 1990s, Coad incorporated behavioral ideas to object-oriented methods.

The other significant innovations were Object Modelling Techniques (OMT) by James Rumbaugh and Object-Oriented Software Engineering (OOSE) by Ivar Jacobson.

The Concept Of Object-Orientation

Object-orientation is what's referred to as a programming paradigm. It's not a language itself but a set of concepts that is supported by many languages

Analysis and Design

For a given problem, **Analysis** focus on investigation of the problem and requirements, rather than searching for a solution. For example, if a new online shopping system is desired, how will the system be used? "Analysis" is a broad term, best qualified, as in *requirements analysis* (an investigation of the requirements) or *object analysis* (an investigation of the domain objects).

Design focuses on a conceptual solution that fulfills the requirements, rather than its implementation. As with analysis, the term is best qualified, as in *object design* or *database design*.

❖ Object-Oriented Analysis And Design (OOAD)

It's a structured method for analyzing, designing a system by applying the object-orientated concepts, and develop a set of graphical system models during the development life cycle of the software.

OR

Object-oriented analysis and design (OOAD) is a technical approach used in the analysis and design of an application or system through the application of the object-oriented paradigm and concepts including visual modeling. This is

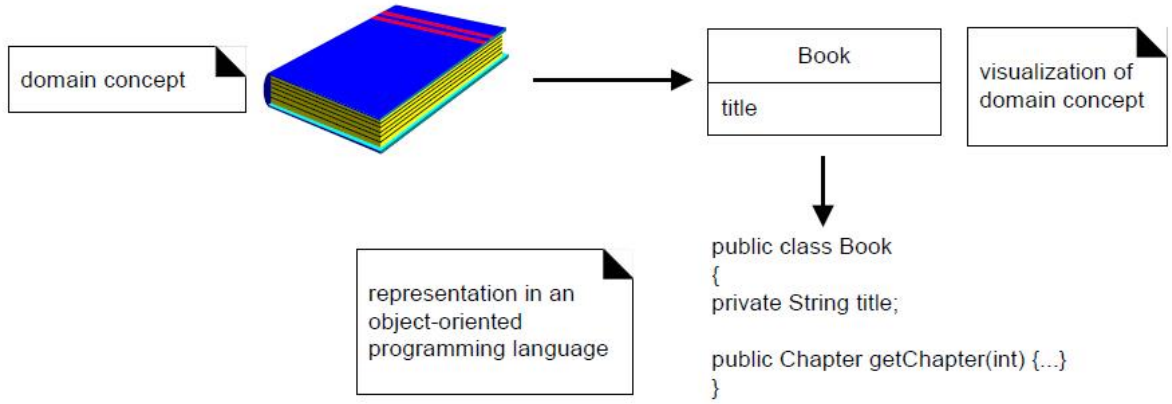
applied throughout the development life cycle of the application or system, fostering better product quality and even encouraging stakeholder participation and communication.

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*.

In the object-oriented analysis, we …

1. **Elicit** *requirements:* Define what does the software need to do, and what's the problem the software trying to solve.
2. **Specify** *requirements*: Describe the requirements, usually, using use cases (and scenarios) or user stories.
3. **Conceptual** *model*: Identify the important objects, refine them, and define their relationships and behavior and draw them in a simple diagram.

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.
Finally, in Implementation or object-oriented programming, the design objects are implemented as classes like *Book* class in Java.



❖ Object Oriented Development

Object Oriented Development (OOD) has been touted as the next great advance in software engineering. It promises to reduce development time, reduce the time and resources required to maintain existing applications, increase code reuse, and provide a competitive advantage to organizations that use it. While the potential benefits and advantages of OOD are real, excessive hype has lead to unrealistic expectations among executives and managers. Even software developers often miss the subtle but profound differences between OOD and classic software development.

Expected Benefits of OOD

Many benefits are cited for OOD, often to an unrealistic degree. Some of these potential benefits are

- ***Faster Development****:* OOD has long been touted as leading to faster development. Many of the claims of potentially reduced development time are correct in principle, if a bit overstated.
- ***Reuse of Previous work:*** This is the benefit cited most commonly in literature, particularly in business periodicals. OOD produces software modules that can be plugged into one another, which allows creation of new programs. However, such reuse does not come easily. It takes planning and investment.
- ***Increased Quality:*** Increases in quality are largely a by-product of this program reuse. If 90% of a new application consists of proven, existing components, then only the remaining 10% of the code has to be tested from scratch. That observation implies an order-of-magnitude reduction in defects.
- ***Modular Architecture:*** Object-oriented systems have a natural structure for modular design: objects, subsystems, framework, and so on. Thus, OOD systems are easier to modify. OOD systems can be altered in fundamental ways without ever breaking up since changes are neatly encapsulated. However, nothing in OOD guarantees or requires that the code produced will be modular. The same level of care in design and implementation is required to produce a modular structure in OOD, as it is for any form of software development.
- ***Client/Server Applications:*** By their very nature, client/server applications involve transmission of messages back and forth over a network, and the object-message paradigm of OOD meshes well with the physical and conceptual architecture of client/server applications.
- ***Better Mapping to the Problem Domain:*** This is a clear winner for OOD, particularly when the project maps to the real world. Whether objects represent customers, machinery, banks, sensors or pieces of paper, they can provide a clean, self-contained implication which fits naturally into human thought processes.

❖ Object oriented themes:-

- Abstraction
- Encapsulation
- Combining Data& behaviour
- Sharing
- Emphasis on the essence of object
- Synergy

1) Abstraction:

- It means focusing on the essential aspects of an entity while ignoring its details.
- This means focusing on what an object is and does, before deciding how it should be implemented.
- Use of abstraction during analysis means dealing only with application domain concepts, not making decisions before problem is understood.

2) Encapsulation:

- It means information hiding. It consists of separating the external aspects of an object, which are accessible to other objects, from internal implementation details of the object, which are hidden from other objects.
- It prevents program from becoming so interdependent that a small change has massive ripple effect. It gives the ability to combine data structure and behaviour in a single entity.

3) Combining Data and Behavior:

- The burden of calling code for data execution and operations separately can be minimized by combining data properties and behavioural properties of an entity together.
- In object oriented program data structure and procedure is defined in single class definition.

4) Sharing:

- Object Oriented technologies promote sharing at different levels. Inheritance of both data structure and behavior lets subclasses share common code.
- This sharing via inheritance is one of the main advantages of Object Oriented languages.
- Object Oriented Development not only lets you share information within an application but also offers the prospect of reusing designs and code on future projects.

5) Emphasis on the essence of object:

- Object Oriented Technology stresses what an object is, rather than how it is used.
- The uses of an object depend on the details of the application and often change during development.

6. Synergy:

- Identity, classification, polymorphism and inheritance characterize Object Oriented languages.
- Each of these concepts can be used in isolation but together they complement each other synergistically.

❖ Usefulness of OOP

Object-oriented programming (OOP) is a programming paradigm based upon objects (having both data and methods) that aims to incorporate the advantages of modularity and reusability. Objects, which are usually instances of classes, are used to interact with one another to design applications and computer programs.

The important features of object–oriented programming are −

- Bottom–up approach in program design
- Programs organized around objects, grouped in classes
- Focus on data with methods to operate upon object's data
- Interaction between objects through functions

- Reusability of design through creation of new classes by adding features to existing classes

❖ OOAD Object Modeling Techniques

The object modeling techniques is an methodology of object oriented analysis, design and implementation that focuses on creating a model of objects from the real world and then to use this model to develop object–oriented software. object modeling technique, OMT was developed by James Rambaugh. Now-a-days, OMT is one of the most popular object oriented development techniques. It is primarily used by system and software developers to support full life cycle development while targeting object oriented implementations.

OMT has proven itself easy to understand, to draw and to use. It is very successful in many application domains: telecommunication, transportation, compilers etc. The popular object modeling technique are used in many real world problems. The object-oriented paradigm using the OMT spans the entire development cycle, so there is no need to transform one type of model to another.

Phase of OMT

The OMT methodology covers the full software development life cycle. The methodology has the following phase.

1. **Analysis** - Analysis is the first phase of OMT methodology. The aim of analysis phase is to build a model of the real world situation to show its important properties and domain. This phase is concerned with preparation of precise and correct modelling of the real world. The analysis phase starts with defining a problem statement which includes a set of goals. This problem statement is then expanded into three models; an object model, a dynamic model and a functional model. The object model shows the static data structure or skeleton of the real world system and divides the whole application into objects. In others words, this model represents the artifacts of the system. The dynamic model represents the interaction between artifacts above designed represented as events, states and transitions. The functional model represents the methods of the system from the data flow perspective. The analysis phase generates object model diagrams, state diagrams, event flow diagrams and data flow diagrams.
2. **System design** - The system design phase comes after the analysis phase. System design phase determines the overall system architecture using subsystems, concurrent tasks and data storage. During system design, the high level structure of the system is designed. The decisions made during system design are:
    - The system is organized in to sub-systems which are then allocated to processes and tasks, taking into account concurrency and collaboration.
    - Persistent data storage is established along with a strategy to manage shared or global information.

- Boundary situations are checked to help guide trade off priorities.

3. **Object design** - The object design phase comes after the system design phase is over. Here the implementation plan is developed. Object design is concerned with fully classifying the existing and remaining classes, associations, attributes and operations necessary for implementing a solution to the problem. In object design:
    - Operations and data structures are fully defined along with any internal objects needed for implementation.
    - Class level associations are determined.
    - Issues of inheritance, aggregation, association and default values are checked.
4. **Implementation** - Implementation pahse of the OMT is a matter of translating the design in to a programming language constructs. It is important to have good software engineering practice so that the design phase is smoothly translated in to the implementation phase. Thus while selecting programming language all constructs should be kept in mind for following noteworthy points.
    - To increase flexibility.
    - To make amendments easily.
    - For the design traceability.
    - To increase efficiency.