**Chapter 3: Object Oriented Analysis (OOA)**

**3.1 Introduction to model building**

* A model is a simplification of reality.
* A model provides the blueprints of a system. A good model includes those elements that have broad effort and omits those minor elements that are not relevant to the given level of abstraction. Every system that are not relevant to the given level of abstraction. Every system may be described from different aspects using different models.

**Three types of model used:**

**1. Object Modelling**

Object modelling develops the static structure of the software system in terms of objects. It identifies the objects, the classes into which the objects can be grouped into and the relationships between the objects. It also identifies the main attributes and operations that characterize each class.

The process of object modelling can be visualized in the following steps −

1. Identify objects and group into classes
2. Identify the relationships among classes
3. Create user object model diagram
4. Define user object attributes
5. Define the operations that should be performed on the classes
6. Review glossary

## 2. Dynamic Modelling

After the static behavior of the system is analyzed, its behavior with respect to time and external changes needs to be examined. This is the purpose of dynamic modelling.

Dynamic Modelling can be defined as “a way of describing how an individual object responds to events, either internal events triggered by other objects, or external events triggered by the outside world”.

The process of dynamic modelling can be visualized in the following steps −

* Identify states of each object
* Identify events and analyze the applicability of actions
* Construct dynamic model diagram, comprising of state transition diagrams
* Express each state in terms of object attributes
* Validate the state–transition diagrams drawn

**3. Functional Modelling**

Functional Modelling is the final component of object-oriented analysis. The functional model shows the processes that are performed within an object and how the data changes as it moves between methods. It specifies the meaning of the operations of object modelling and the actions of dynamic modelling. The functional model corresponds to the data flow diagram of traditional structured analysis.

The process of functional modelling can be visualized in the following steps −

* Identify all the inputs and outputs
* Construct data flow diagrams showing functional dependencies
* State the purpose of each function
* Identify constraints
* Specify optimization criteria

**Importance**

* We build models to better understand the system we are developing.
* Modeling achieves four aims.
* Hepls us to visualize a system as we want it to be.
* Permits us to specify the structure of behavior of a system.
* Gives us a template that guides us in constructing a system.
* Documents the decisions we have made.

## Structured Analysis vs. Object Oriented Analysis

The Structured Analysis/Structured Design (SASD) approach is the traditional approach of software development based upon the waterfall model. The phases of development of a system using SASD are −

* Feasibility Study
* Requirement Analysis and Specification
* System Design
* Implementation
* Post-implementation Review

Now, we will look at the relative advantages and disadvantages of structured analysis approach and object-oriented analysis approach.

### **Advantages/Disadvantages of Object Oriented Analysis**

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Focuses on data rather than the procedures as in Structured Analysis. | Functionality is restricted within objects. This may pose a problem for systems which are intrinsically procedural or computational in nature. |
| The principles of encapsulation and data hiding help the developer to develop systems that cannot be tampered by other parts of the system. | It cannot identify which objects would generate an optimal system design. |
| The principles of encapsulation and data hiding help the developer to develop systems that cannot be tampered by other parts of the system. | The object-oriented models do not easily show the communications between the objects in the system. |
| It allows effective management of software complexity by the virtue of modularity. | All the interfaces between the objects cannot be represented in a single diagram. |
| It can be upgraded from small to large systems at a greater ease than in systems following structured analysis. |  |

### **Advantages/Disadvantages of Structured Analysis**

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| As it follows a top-down approach in contrast to bottom-up approach of object-oriented analysis, it can be more easily comprehended than OOA. | In traditional structured analysis models, one phase should be completed before the next phase. This poses a problem in design, particularly if errors crop up or requirements change. |
| It is based upon functionality. The overall purpose is identified and then functional decomposition is done for developing the software. The emphasis not only gives a better understanding of the system but also generates more complete systems. | The initial cost of constructing the system is high, since the whole system needs to be designed at once leaving very little option to add functionality later. |
| The specifications in it are written in simple English language, and hence can be more easily analyzed by non-technical personnel. | It does not support reusability of code. So, the time and cost of development is inherently high. |

**3.2 Domain Modeling**

- A domain model is a visual representation of conceptual classes or real world objects in a domain of interest.

- They have also been called conceptual models, domain object models and analysis object models.

**-** A domain model is a package containing class and activity diagrams.

**Domain Model representation**

* Identifying a rich set of objects or conceptual classes is at the heart of object-oriented analysis, and well worth the effort in terms of payoff during the design and implementation work.
* Using UML notation, a domain model is illustrated with a set of class diagrams in which no operations are defined.
* Domain objects or conceptual classes.
* Associations between conceptual classes
* Attributes or conceptual classes

**Syntax in domain modeling**

****

**Conceptual Classes**

The domain model illustrates conceptual classes or vocabulary in the domain. Informally, a conceptual class is an idea, thing or object. More formally a conceptual class may be considered in terms of its symbol, intension and extension.

**Symbol:** words or images representing a conceptual class.

**Intension:** the definition of a conceptual class.

**Extension:** the set of examples to which the conceptual class applies.

****

Fig: A conceptual class has a symbol, intension and extension.

**Finding conceptual class**

Strategies for identifying conceptual classes:

* Reuse for modify existing models
* There exist domain models and date models for many common domains.
* Use a category list
* Identify noun phase (linguistic analysis)
* Noun ro concept mapping

**Identifying conceptual classes by category list conceptual class category (example)**

|  |  |  |
| --- | --- | --- |
|  | Store | Airline Reservations |
| Physical or tangible objects | Register | Airplane |
| Places | Store | Airport |
| Transactions | Sale | Payment |
| Role of people | Cashier | Pilot |
| Organization | Sales department | Airline |
| Specifications, design | Product | Flight description |
| Description of things | specification |  |
| Rules and policies | Refund policy | Cancellation policy |
| Contains of other things | Store, bin | Airplane |
| Things in a container | Item | Passengers |

**Identifying conceptual classes by Noun Phrase**

* Another useful technique is linguistic analysis: identify the noun and noun phrases in textual description of a domain and consider them as candidate conceptual classes or attributes.
* Identify Nouns and noun phrases in textual description of the domain.
* Candidate conceptual classes for the sales domain from the conceptual class category list and noun phrase analysis. A list is generated of candidate conceptual classes for the domain. The list is constrained to the requirements and simplification currently under consideration the simplified scenario of process sale.
* Register, product specification, Item, SalesLineItem, store cashier, sale customer, payment, manager, product catalog.

**3.3 Domain Model Refinement**

**a) Finding association**

- An association is a relationship between instances of classes that indicates some interesting and meaningful connection.

- An association is represented as a line between classes with an association name. The association is inherently bidirectional meaning that form instances of their class, logical traversal to there is possible.

**b) Roles**

- Each end of an association is called a role.

- Named to enhance understanding of the relationship.

- Multiplicity: what number of instances can be associated

- Direction arrow: Just helps the reads

- No meaning for the model, often omitted.

**c) Attributes**

- An attribute is a logical data value of an object.

-Use to identify those attributes of conceptual classes that are needed to satisfy information requirements of the current scenario under development.

- Most attributes can be represented by simple data types.

- An attributes by part of the state of an object.

- Complex concepts should be related through associations, not through attributes.

- Attributes are shows in the second component of the class box.



**d) Class Relationship**

1. **Aggregation**

It is a specialized from of association where all objects have their own life cycle, but there is ownership and child object cannot belong to another parent object.



In figure department and teacher, a single teacher cannot belong to multiple departments but we delete the department, the teacher object will not be restored. We can think about it as a “has a” relationship.

1. **Composition:**

I’s again specialized for of aggregation and we can call this as a “death” relationship. It’s a strong type aggression child object doesn’t have it’s; life cycle and if parent object is deleted, all child objects will also be deleted.

Let’s example of relation between house and rooms. House can contain multiple rooms. There is no independent life of room and any room cannot belong to two different houses. If we delete the house room will be automatically deleted.



1. **Class diagram**

* A class diagram is a diagram describing the structure of a system.
* Shows the system’s
  + Classes
  + Attributes
  + Operations
  + Relationships among the classes
* A class diagram depicts classes and their interrelationships.
* Used for describing structure and behavior in the use cases.
* Used for requirements capture, end user interaction.
* Detailed class diagram is used for developers.

1. **Essential elements of a UML class diagram**
   * Class
   * Attributes
   * Operations
   * Relationships
     + Association
     + Generalization
     + Realization
     + Dependency
2. **Constraint Rules and Notes**

* Describes a set of objects having similar attributes, operations and relationships with other classes.
* Graphically, a class is rendered as a rectangular, usually including its name, attributes, and operations in separate designed compartments.
* Class Name: The name of the class is the only required tag in the graphical representation of a class. It always appears in the top-most compartment.
* Class attributes and operations: An attribute is a named property of a class that describes the object being modeled.
* Attributes can be public, private or protected.
* Operations describe the class behavior and appears in the third compartment.

1. **Difference Between aggregation and composition**

* **Aggregation** implies a relationship where the child can exist independently of the parent. Example: Class (parent) and Student (child). Delete the Class and the Students still exist.
* **Composition** implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child). Rooms don't exist separate to a House.
* **Aggregation** is a special form of association. It is also a relationship between two classes like association, however it’s a directional association, which means it is strictly a one-way association. It represents a Has-A relationship
* **Composition i**n an object-oriented design of a **Java** program, the way in which you model objects that contain other objects is with **composition**, the act of composing a class out of references to other objects. With **composition**, references to the constituent objects become fields of the containing object.

**3.4 UML activity diagram and modeling**

* Activity diagram is another important diagram in UML to describe dynamic aspects of the systems. It’s basically flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. This flow can be sequential branched or concurrent.
* Activity diagram consists of activities, links, relationships etc.
* It models