**Lab # 6**

**Object Oriented Analysis:**

**Discovering Classes**

**Objectives:**

Learn the object-oriented analysis phase by understanding the methods of class elicitation and finding the classes in an object-oriented system.

**1. Outline**

* Object-Oriented concepts
* Discovering classes’ approaches: noun phrase approach, common class patterns, use case driven method, CRC (Class-Responsibility-Collaboration) and mixed approach.
* Examples.

**2. Background**

Classes: a description of a group of objects with common properties (attributes), common behavior (operations), common relationships to other objects and common semantics.

**2.1 Object-Oriented Concepts**

* Attribute: the basic data of the class.
* Method (operation): an executable procedure that is encapsulated in a class and is designed to operate on one or more data attributes that are defined as part of the class.
* Object: when specific values are assigned to all the resources defined in a class, the result is an instance of that class. Any instance of any class is called an object.

**2.2 Discovering Classes**

Discovering and defining classes to describe the structure of a computerized system is not an easy task. When the problem domain is new or unfamiliar to the software developers it can be difficult to discover classes; a cookbook for finding classes does not exist.

**2.3 Classes Categories**

Classes are divided into three categories:

* Entity: models information and associated behavior that is long-lived, independent of the surrounding, application independent, and accomplishes some responsibility
* Boundary: handles the communication between the system surroundings and the inside of the system, provides interface, and facilitates communication with other systems
* Control: model sequencing behavior specific to one or more use cases. Control classes coordinate the events needed to realize the behavior specified in the use case, and they are responsible for the flow of events in the use case.

**2.4 Discovering Classes Approaches**

Methods of discovering classes:

**2.4.1 Noun Phrase Approach**

Examine the requirements and underline each noun. Each noun is a candidate class; divide the list of candidate classes into:

* Relevant classes: part of the application domain; occur frequently in requirements.
* Irrelevant classes: outside of application domain
* Fuzzy classes: unable to be declared relevant with confidence; require additional analysis

**2.4.2 Common Class Patterns**

Derives candidate classes from the classification theory of objects; candidate classes and objects come from one of the following sources:

* Tangible things: e.g. buildings, cars.
* Roles: e.g. teachers, students.
* Events: things that happen at a given date and time, or as steps in an ordered sequence: e.g. landing, request, interrupt.
* Interactions: e.g. meeting, discussion.
* Sources, facilities: e.g. departments.
* Other systems: external systems with which the application interacts.
* Concept class: a notion shared by a large community.
* Organization class: a collection or group within the domain.
* People class: roles people can play.
* Places class: a physical location relevant to the system.

**2.4.3 Use Case Driven Method**

The scenarios - use cases that are fundamental to the system operation are enumerated. Going over each scenario leads to the identification of the objects, the responsibilities of each object, and how these objects collaborate with other objects.

**2.4.4 CRC (Class-Responsibility-Collaboration)**

Used primarily as a brainstorming tool for analysis and design. CRC identifies classes by analyzing how objects collaborate to perform business functions (use cases).

A CRC card contains: name of the class, responsibilities of the class and collaborators of the class. Record name of class at the top; record responsibilities down the left-hand side; record other classes (collaborators) that may be required to fulfill each responsibility on the right-hand side.

CRC cards are effective at analyzing scenarios; they force you to be concise and clear; they are cheap, portable and readily available.

**2.4.5 Mixed Approach**

A mix of these approaches can be used, one possible scenario is:

* Use CRC for brainstorming.
* Identify the initial classes by domain knowledge.
* Use common class patterns approach to guide the identification of the classes.
* Use noun phrase approach to add more classes.
* Use the use case approach to verify the identified classes.

**2.5 Class Elicitation Guidelines**

* A class should have a single major role.
* A class should have defined responsibilities (use CRC cards if needed).
* Classes should be of a manageable size: if a class has too many attributes or operations, consider splitting it.
* A class should have a well-defined behavior, preferably by implementing a given requirement or an interface.

**4. In-Class Example**

Now you will learn how to apply the above mentioned methods of finding classes from the problem statement.

**Lab Task:**

1. Draw class diagram for online shopping
2. Draw class diagram for Library Management system