Experiment No: 06

Aim: To draw the class diagram for selected case study with any open source (Dia software).

Objective:

After implementing this experiment you will be able to:

- Learn and draw Use case and Class diagram using rational rose.
- Learn relationships among different classes and use-cases.

Outcome:

- Ability to analyze the system using class and usecase diagram.
- Develop Class and Usecase diagram using Rational-rose.

Theory:

UML:

Unified Modeling Language (**UML**) is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

UML includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems.

The UML is used to specify, visualize, modify, construct and document the of an object-oriented software-intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements such as:

activities
actors
business processes
database schemas
(logical) components
programming language statements
reusable software components.

Usecase Diagram:

Usecase:-A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

Actor

An **actor** in the Unified Modeling Language (UML) "specifies a role played by a user or any other system that interacts with the subject."

"An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject."

"Actors may represent roles played by human users, external hardware, or other subjects. Note that an actor does not necessarily represent a specific physical entity but merely a particular facet (i.e., “role”) of some entity that is relevant to the specification of its associated use cases.

Thus, a single physical instance may play the role of several different actors and, conversely, a given actor may be played by multiple different instances.

Associations

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other.

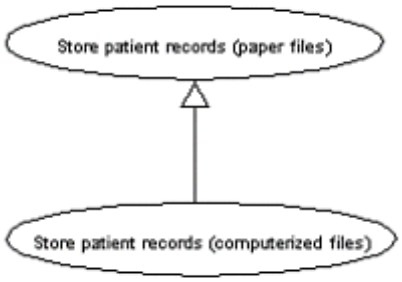
Relationships in UML Use Cases

UML use cases share different kinds of relationships. A relationship between two use cases is basically a dependency between the two use cases. Defining a relationship between two use cases is the decision of the modeler of the use case diagram. This reuse of an existing use case using different types of relationships reduces the overall effort required in defining use cases in a system. A similar reuse established using relationships, will be apparent in the other UML diagrams as well. Use case relationships can be one of the following:

- Include: An include relationship is depicted with a directed arrow having a dotted shaft. The tip of the arrowhead points to the parent use case and the child use case is connected at the base of the arrow. The stereotype "<<include>>" identifies the relationship as an include relationship.

- Extend: In an extend relationship between two use cases, the child use case adds to the existing functionality and characteristics of the parent use case. An extend relationship is depicted with a directed arrow having a dotted shaft, similar to the include relationship.

- Generalizations: A generalization relationship is also a parent-child relationship between use cases. The child use case in the generalization relationship has the underlying business process meaning, but is an enhancement of the parent use case.



Usecase diagram for project:

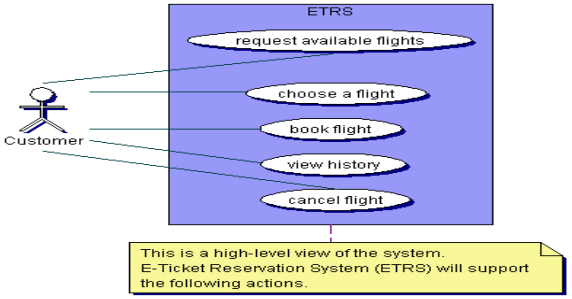
E-Ticket Reservation System –Use Case Diagram

I. Request Flight

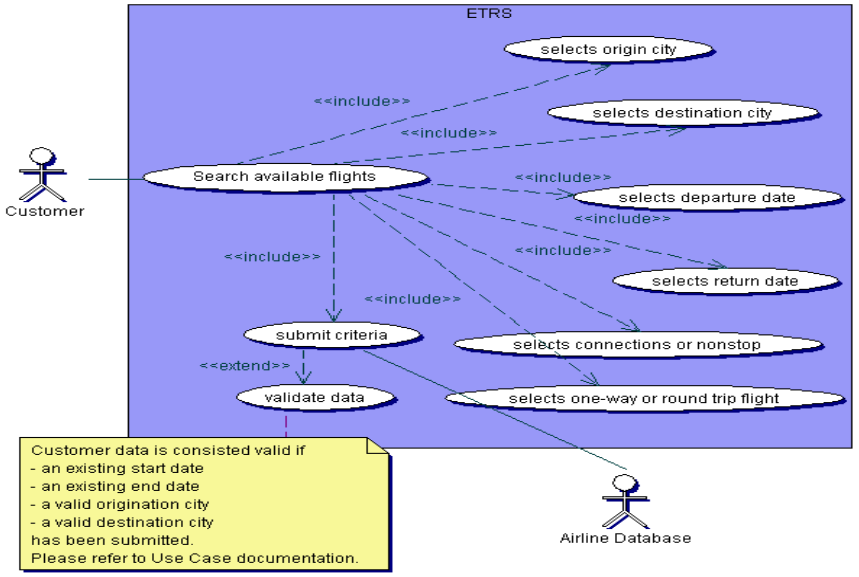
II. Choose A Flight

III. Book A Flight

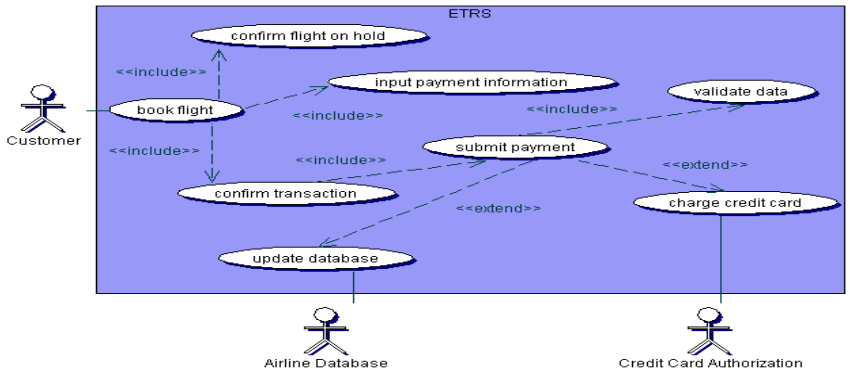
1. Request Flight



2. Choose A Flight



3. Book A Flight



Class Diagram

The class diagram is the main building block in object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code.

In the class diagram these classes are represented with boxes which contain three parts:

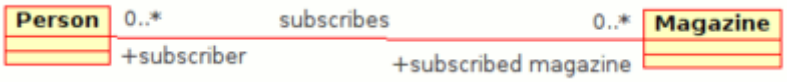
- The upper part holds the name of the class
- The middle part contains the attributes of the class
- The bottom part gives the methods or operations the class can take or undertake

In the system design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects. With detailed modeling, the classes of the conceptual design are often split into a number of subclasses.

Classes

An object is any person, place, thing, concept, event, screen, or report applicable to your system. Objects both know things (they have attributes) and they do things (they have methods). Classes form the main building blocks of an object-oriented application.

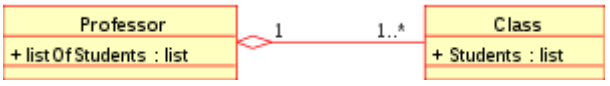
Association



Class diagram example of association between two classes

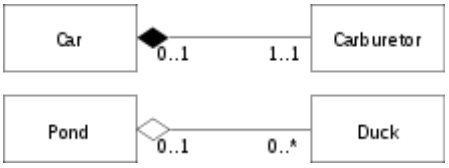
An Association represents a family of links. Binary associations (with two ends) are normally represented as a line, with each end connected to a class box. Higher order associations can be drawn with more than two ends. In such cases, the ends are connected to a central diamond.

Aggregation



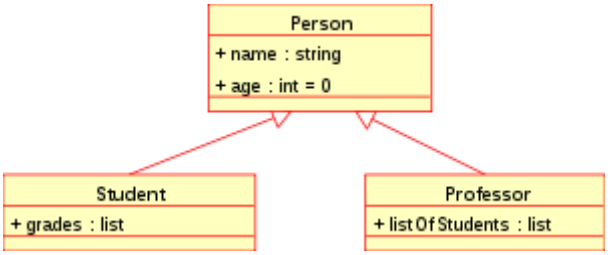
Aggregation is a variant of the "has a" or association relationship; aggregation is more specific than association. It is an association that represents a part-whole or part-of relationship. As a type of association, an aggregation can be named and have the same adornments that an association can.

Composition



Composition is a stronger variant of the "owns a" or association relationship; composition is more specific than aggregation.

Generalization



Class diagram showing generalization between one superclass and two subclasses

The Generalization relationship indicates that one of the two related classes (the subtype) is considered to be a specialized form of the other (the super type) and supertype is considered as

Class diagram for project:

Railway Reservation System Class Diagram