Experiment No. 1. Develop Software Requirement Specification for Library Management System (The requirements specification should include functional and non-functional requirements).

Aim: To understand Software Requirement Specification.

Theory:

Software Requirement Specification (SRS) document usually contains a software vendor's understanding of a customer's software requirements. It is created after the initial requirement elicitation phase in which software vendor interacts with the customer to understand the software needs. Usually SRS documentation is prepared by a business analyst who has some technical background.

Functional Requirements

Functional requirements specify the business requirements of the project in detail. Usually business requirements are specified in terms of the actions that user performs on the software system. This is known as the use case model. But not all requirements need to be specified as use cases. Functional requirements should contain a combination of use cases and plain textual description of system features. System features are specified at a higher level and use cases attempt to translate into user actions. There is no fixed format for use case description, but it usually contains the following information.

- Use case diagram – For a small system, a single diagram can be used to depict all the use cases in the system.
- List of actors and their details – This identifies the various types of users interacting with the software system.
- Use case description – Purpose of the use case and how and when it is invoked by the user. This should also include an identifier for easy reference.
- Preconditions – List of system states/conditions that must be true for the successful execution of the use case. This section is optional and could be easily incorporated into the basic steps section.
- Basic steps – These indicates the various fine grained steps required for the execution of the use case.
- Alternate steps – These indicate alternate events of the use case being described.
- Business validations/rules – These indicates various types of input validations or business rules required in the use case being described.
- Post conditions – Indicates the results of the use case.

Non Functional Requirements

Non functional requirements specify how the software system should operate. In contrast functional requirements specify what a software system should do. Some of the non functional requirements are derived from the functional requirements. Nonfunctional requirements captured include performance requirements, application scalability, application security, maintainability, usability, availability, logging and auditing, data migration requirements.

SRS template

A good SRS template ensures that all important information required in a Software Requirement Specification is captured during requirement elicitation.

Library Management System: A library management system, also known as an automated library system is software that has been developed to handle basic housekeeping functions of a library.

- It is a well-organized software solution for a library.
- It helps to provide information on any book present in library to the user as well as staff member.
- It keeps a track of book issued, returned and added to library.

Viva Question:

1. What is functional and non-functional requirement?
2. What do you mean by data dictionary?
3. What is the requirement of SRS?
4. After which phase SRS should be created?
5. Who prepares the SRS?

Experiment No. 2. Design Use case diagram for Library Management System.

Aim: To understand the concept of Use case diagram.

Software Required: ArgoUML

Theory:

Use case diagram

Use case diagram is the primary form of system/software requirements for a new software program underdeveloped. Use cases specify the expected behavior (what), and not the exact method of making it happen (how). Use cases once specified can be denoted both textual and visual representation (such as UML). A key concept of use case modeling is that it helps us design a system from end user's perspective. It is an effective technique for communicating system behavior in the user's terms by specifying all externally visible system behavior.

A use case diagram contains four components.

- **The boundary**, which defines the system of interest in relation to the world around it.
- **The actors**, usually individuals involved with the system defined according to their roles.
- **The use cases**, which specify roles, are played by the actors within and around the system.
- **The relationships** between and among the actors and the use cases.

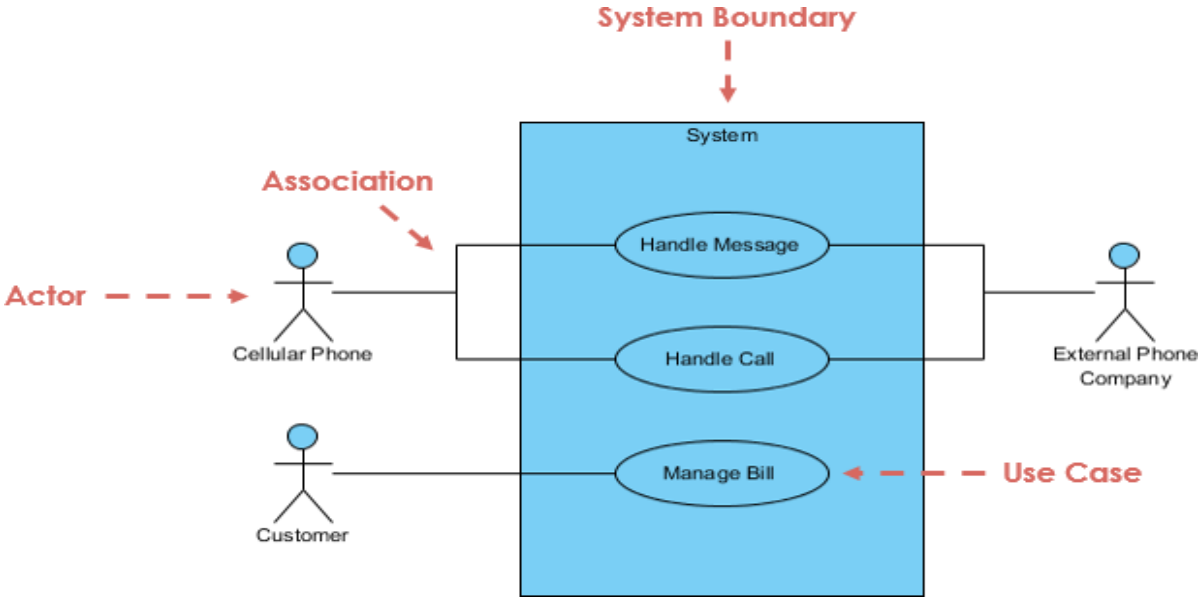

Fig. 2.1 Notations used in Use Case diagram

Viva Question:

1. What is use case?
2. Where use case diagram can be used?
3. What is the difference between use case diagram and use case?
4. What is actor in use case diagrams?
5. Explain “include” and “exclude” in Use case diagram.

Experiment No. 3. Design Class diagram for Library Management System.

Aim: To understand the concept of Class diagram.

Software Required: ArgoUML

Theory:

Class diagrams are the main building block in object-oriented modeling. They are used to show the different objects in a system, their attributes, their operations and the relationships among them.

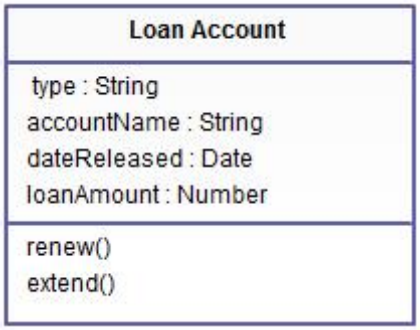

Fig. 3.1 Class Diagram

In the figure 3.1, a class called "loan account" is depicted. Classes in class diagrams are represented by boxes that are partitioned into three:

- The top partition contains the name of the class.
- The middle part contains the class's attributes.
- The bottom partition shows the possible operations that are associated with the class.

Relationships in Class Diagrams

Relationships in class diagrams include different types of logical connections. The following are such types of logical connections that are possible in UML:

Association is a broad term that encompasses just about any logical connection or relationship between classes.

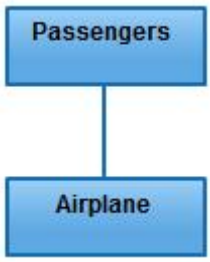

Fig. 3.2 Association

Directed Association refers to a directional relationship represented by a line with an arrowhead. The arrowhead depicts a container-contained directional flow.

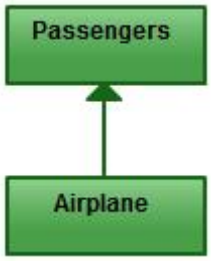

Fig. 3.3 Directed Associations

Reflexive Association: - This occurs when a class may have multiple functions or responsibilities.

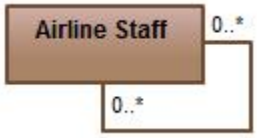

Fig. 3.4 Reflexive Association

Multiplicity is the active logical association when the cardinality of a class in relation to another is being depicted.

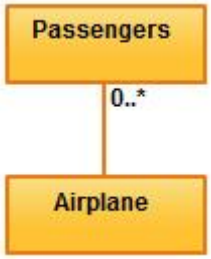

Fig. 3.5 Multiplicity

Aggregation refers to the formation of a particular class as a result of one class being aggregated or built as a collection.

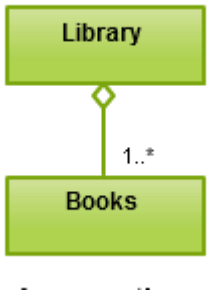

Fig. 3.6 Aggregation

Composition: - The composition relationship is very similar to the aggregation relationship with the only difference being its key purpose of emphasizing the dependence of the contained class to the lifecycle of the container class. That is, the contained class will be obliterated when the container class is destroyed.

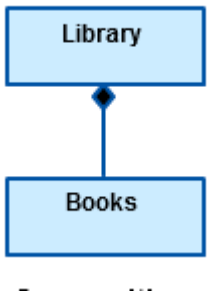

Fig. 3.7 Composition

Inheritance / Generalization refer to a type of relationship wherein one associated class is a child of another by virtue of assuming the same functionalities of the parent class.

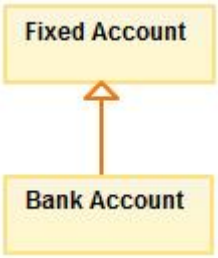

Fig. 3.8 Inheritance / Generalization

Realization denotes the implementation of the functionality defined in one class by another class.

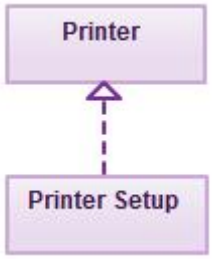

Fig. 3.9 Realization

The following points should be remembered while drawing a class diagram −

- The name of the class diagram should be meaningful to describe the aspect of the system.
- Each element and their relationships should be identified in advance.
- Responsibility (attributes and methods) of each class should be clearly identified
- For each class, minimum number of properties should be specified, as unnecessary properties will make the diagram complicated.

Viva Question:

1. What is class and methods?
2. Where class diagram can be used?
3. What is association relation?
4. Explain generalization and specialization?
5. What are the different components of class diagram?